Software Design Document

Cloud Connect

2/13/2018

Project Sponsor: Tony Pallas

Faculty Mentor: Ana Paula Chaves Steinmacher

Version 2

Team Members:

Parth Patel
Abraham Ramirez
Jacob Serafin
Steven Strickland

Table of Contents

Introduction

Within the hospitality industry, hotels are equipped with hardware on site to help produce an efficient and secure stay for customers. Devices on site transfer data to systems throughout the Internet where it can be used as a way to enhance the customer experience and keep track of various information. One example is the devices that are in place to organize reservation data to make sure customers can reserve rooms and services. Other devices work in similar ways, facilitating many services that are seen as essential for the modern hospitality industry.

Choice Hotels is a company that owns and operates multiple chains of hotels with up to 6,500 hotels deployed globally. They are a billion dollar company that operates, in aggregate, approximately 15.5 million available rooms each night. Each hotel is set up with a wide variety of sophisticated electronic systems. These systems provide security, conveniences, and luxuries for every one of the 15.5 million rooms every day. SkyTouch Technology is a Software as a Service (SaaS) company that provides an operating system for which they can move data from these electronic systems to online databases, allowing all of this information to be collected and used. Team CloudConnect has partnered with both SkyTouch Technology and Choice Hotels to help provide a more modern solution to their current systems.

The goal of our team is to improve the capability of SkyTouch's solution to access and manage the many varied hardware systems installed in today's modern hotels. Many of these hotels rely on old standard RS232 serial port connections to send and receive information from these systems on site at hotels. Figure 1 is a modeled representation of SkyTouch's current solution to proxy devices from any given hotel to and from their own database located within an online AWS database. Within the "on-site" section to the left represents any given hotel. The devices run through their own systems and, currently, send serial data to a PC at the hotel that

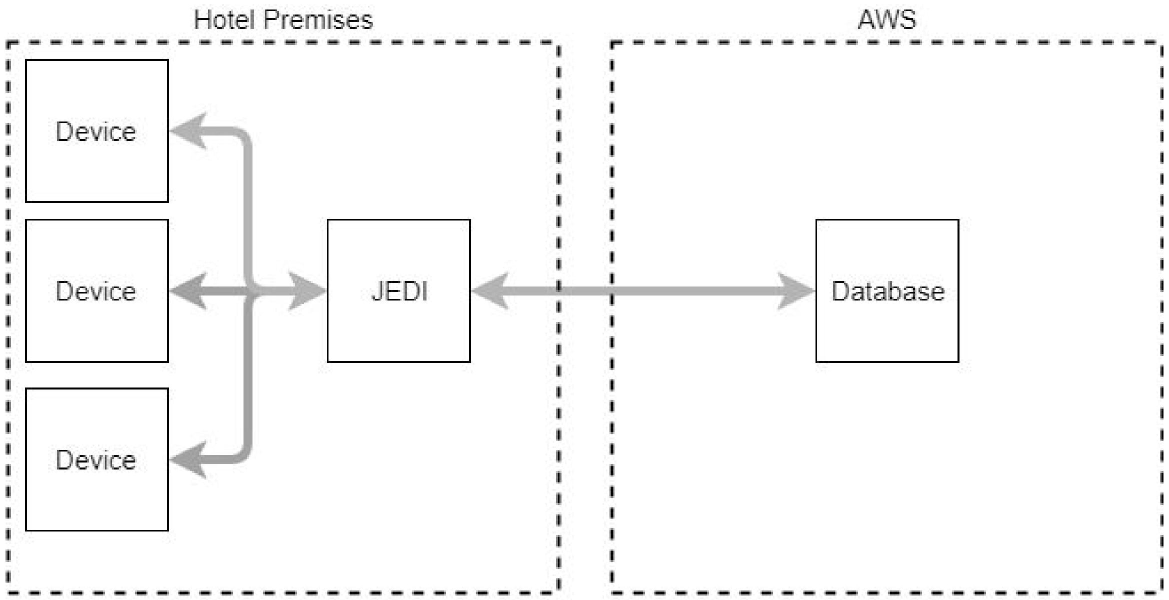


Figure 1

runs custom made software called JEDI (Java Enabled Device Interface). The JEDI will then translate the
incoming device data and from here, the JEDI then runs a proxy from the hotel to an online AWS database which is provided by SkyTouch’s Amazon Web Services account.

Overall, this structure has been kept intact since 1998 when JEDI was first implemented into Choice Hotels’ locations. This leads to several key flaws that cause massive waste in both time and money. Each JEDI must be custom setup and developed, wasting both time and money. In addition, the use of the old standard RS-232 connections cause these connections to be slow, further wasting time. To make this situation even worse, to keep this structure at any given hotel it costs a lot of money to maintain and develop as time continues. SkyTouch has begun to move their systems into a cloud-based solution, but have yet to resolve the issue of how to produce a cloud based system which would not need JEDI at each location.

Our proposed solution is to eliminate the need for JEDI by installing a server at each hotel to replace the current JEDI. This server will communicate serial data through IP/TCP protocols to a custom application hosted inside AWS, that will translate the incoming data and then send it to the current database within AWS. This solution will eliminate the need for a custom JEDI at each hotel, and will allow centralized management and development within AWS.

Implementation Overview

We will select a server which will be connected to the devices at the hotel. This server will also connect to a router that can send data to our custom application hosted in AWS. Our application will translate the incoming data using custom made XML files, and then send the translated data to the current database in AWS. This functionality is shown in Figure 2 with red arrows. In addition, our application will work in the reverse as described above. Our application will receive data from the current AWS database, use custom XML files to translate the data to the destination device's requirements, and then send it the the server that connects to that specified device. his functionality is shown in Figure 3 with red arrows

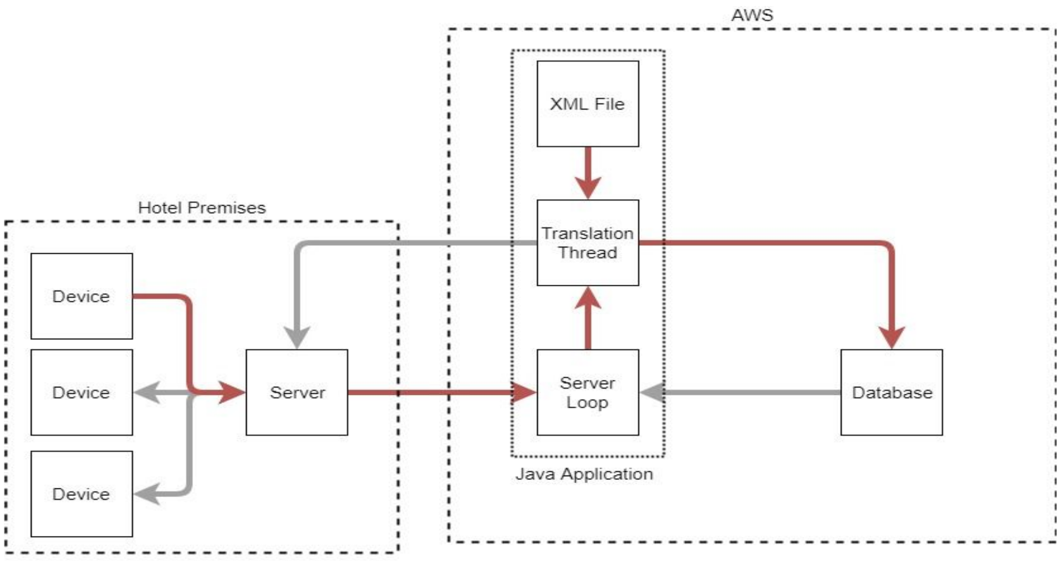


Figure 2

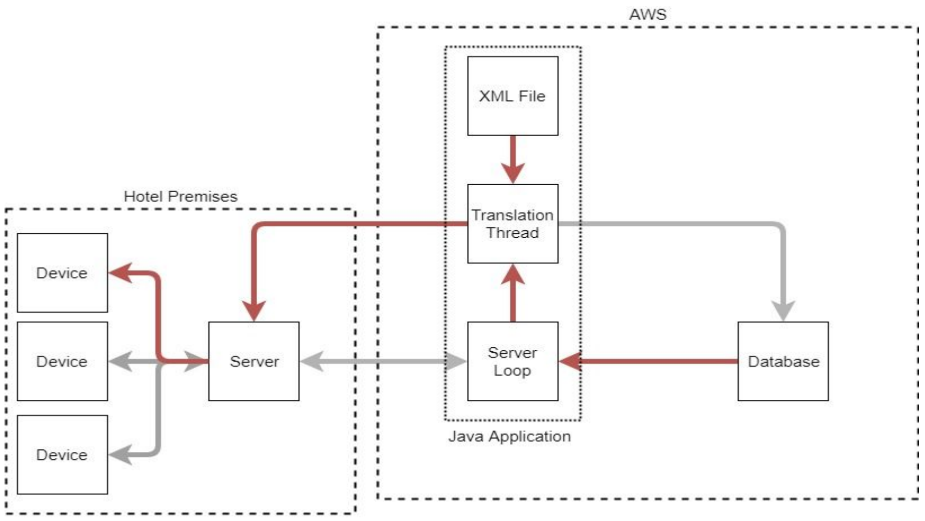


Figure 3

Key Features:

- Communication between the server and custom Java translation server
- Communication between the custom Java translation server and the AWS database
- Custom Java translation server uses XML files to translate data
- Custom Java translation server is multi-threaded, one thread per translation
- Eliminate the need for JEDI configuration within each hotel

Our Application will be written in Java, and will require extensive usage of XML files and parsing of those files. Putting our solution in AWS for demonstration and testing purposes will require knowledge of the web services available in AWS as well as extensive networking knowledge. For demonstration purposes, two GUI applications will be created, one to simulate the server sending data from a PBX device and one simulating the server sending data from a checkout device. These two applications will serve to provide a visual for our demonstration. Creating these GUI applications will require Java knowledge as well.

Our solution will change the way SkyTouch Currently works in its back end. There is not going to be much change on front end of the current system. There won't be a need for any current hotel using the old system to switch to the new system, as those JEDIs will still operate normally. New hotels can start with the new system with no issues, and old hotels can be switched to the new system for relatively low cost at any point.

Architectural Overview

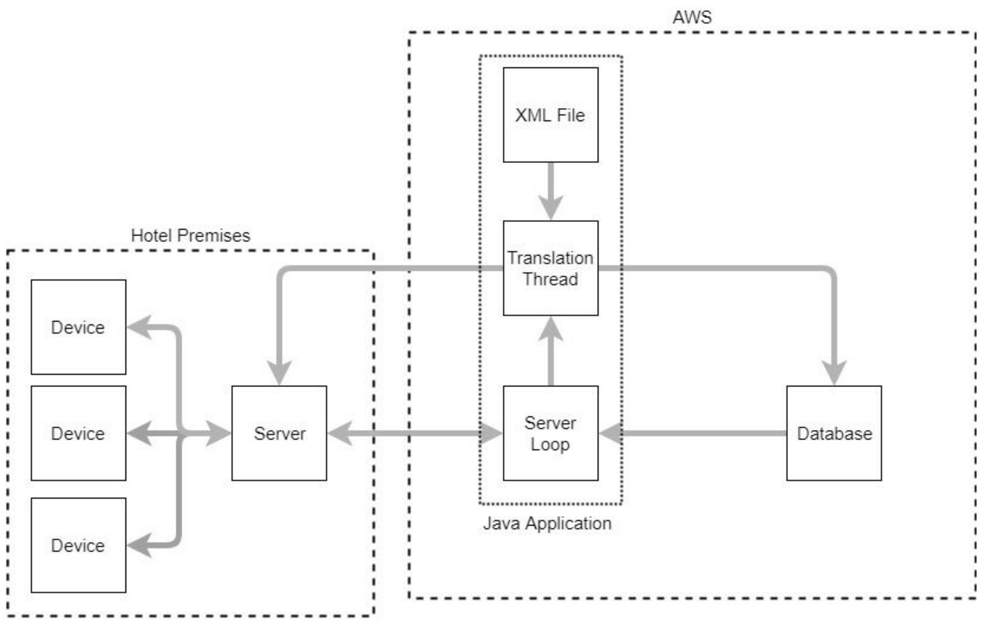


Figure 4

As shown in Figure 4, our solution is a Java application that is hosted inside of AWS (shown as the dotted box labeled “Java Application”). It is made up of three main modules. These modules work together to provide the full functionality of our application. Below are each of the three main modules and a brief description of their purpose.

- Server Loop
    - This module is a loop that constantly cycles, listening for any incoming connections. When an incoming connection is successfully connected, the server loop passes the connection into a new Translation Thread. After passing the connection, the server loop returns to cycling and waiting from new incoming connections.

- Translation Thread
    - This module takes the data from its connection and reads through each XML File until it finds a XML File that contains instructions for the type of data the module is trying to translate. Once it finds the correct XML File, it will then follow the instructions in that file. Once completed, the Translation Thread will then exit, releasing the resources it was using.

- XML File
    - This module is a file, formatted in such a way that the Translation Thread can read it and use it to properly translated according to the instructions contained within this file. The format for this file and its instructions is described in the next section of this document.

Besides the three main modules in our solution, there are a few systems that interact with our solution. These systems are not part of our solution, thus we are not responsible for ensuring their functionality nor are we responsible for installing our solution to interact with these systems for our demonstration. These systems are described in brief below:

- Database
    - The Database is the current SkyTouch Database that is hosted inside AWS and holds much of al the information gathered from the various customers at any hotel that uses the SkyTouch systems.

- Server
    - The Server is a hardware device that is connected to each of the various devices at a hotel . It takes incoming data from the connected devices and sends it to the location of our application. Likewise, when incoming data from our application comes to the Server, it sends this data to the correct connected device.

- Device
    - A Device is a third party piece of hardware that functions according to third party specifications, but will ultimately send and receive data on a predominantly RS-232 connection.

Module and Interface Descriptions

Server Loop

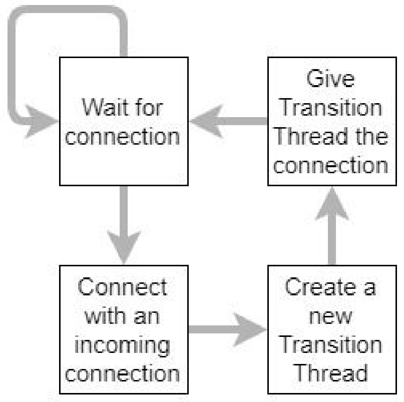
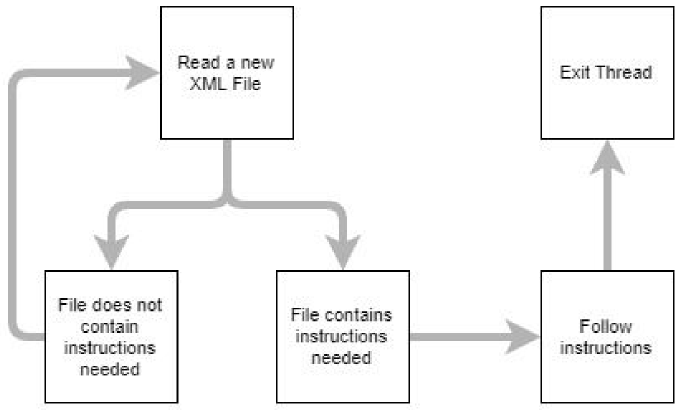

Wait for connection
Give Transition Thread the connection
Connect with an incoming connection
Create a new Transition Thread

Figure 5

The Server Loop module constantly loops, checking for any incoming connections. When an incoming connection is found, it will create a new translation thread, and give that thread the connection. Once the thread has started, the server continues to loops, starting the cycle over again.

Translation Thread


Read a new XML File
Exit Thread
File does not contain instructions needed
File contains instructions needed
Follow instructions

Each translation thread is created specifically to handle one connection. It starts by reading through the current directory, generating a list of xml files. It then selects the first xml file, opening the file and searching for the files “ID” string in the connection's message. If the “ID” string is not found in the message, the translation thread closes the file, removes it from the list, and repeated with the next xml file.

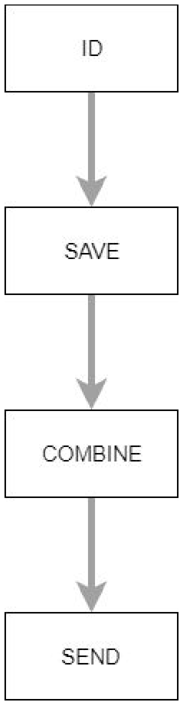
Once the “ID” string is found, the translation thread searches the message for the strings labeled “Key” that are in the xml file’s “SAVE” field. Each time the translation thread finds a “Key”, it saves it for later use. The translation thread saves the “Key” as starting at the end of the found string plus Y number of characters, and going for X amount of characters, where X and Y is specified by the contents of the corresponding “Index” field (Y:X). Both Y and X are allowed to be negative.

Once every Key has been searched for and either found or determined not to be in the connection’s message, the thread then creates the message whose format is specified in the “COMBINE” field. The message is enclosed in double quotes, and adding of saved Keys from the “SAVE” field is done using the plus symbol . (“example “ + Key1 + “more example.”).

Finally, once the translation thread has created the final message, it sends the message to the database and then quits.

XML File

Files will follow a generic structure but will vary depending on what device the file is meant to be for. Each device will have a specific file written for it that the parser will use to understand how to parse the data and what to do with it. Each file will only need to be created once, after which they can be edited to reflect any changes in the device's configuration.

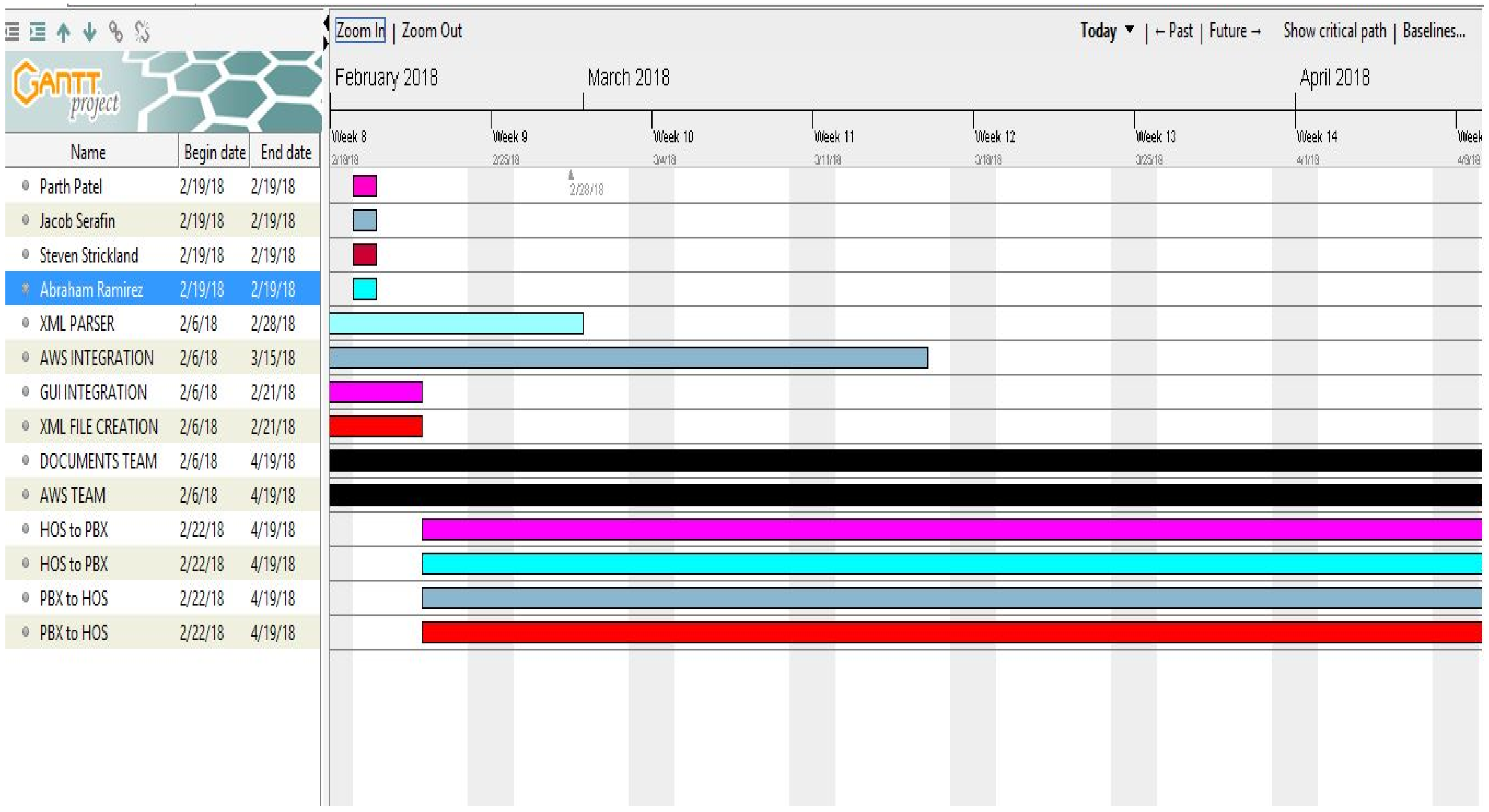


The ID field will contain a string that can be used to identify whether the message being parsed is a valid message type that this file will be able to describe how to parse. The SAVE field will describe what bits of the message need to be saved to be used in the final message. The COMBINE field will show how the final message will use the saved values from the SAVE field. The SEND field will then confirm to send the final message from the COMBINE field

Implementation Plan

Our implementation schedule as of right now is shown below in the gantt chart but as any implementation schedules it can and probably will change in the upcoming weeks. If any changes are made or additions to the gantt chart it will be updated accordingly for everyone to see.

In the chart shown below we begin our schedule with all of our team members working in their individual projects for the past and upcoming weeks starting at week 6. The first few weeks we will be focusing on our individual tasks which include, XML parsing, AWS integration, GUI integration and XML file creation. These tasks will help us move forward with our future projects that come in the upcoming weeks starting about week 9. During week 9 we begin implementing the actual hotel devices in this case we will be working with PBX to Hotel operating system and be doing various tests and implementations during said weeks. The PBX implementation is the beginning of multiple devices for us to implement which will change the gantt chart in the upcoming weeks. We added the black outline in the gantt chart which represents the whole team and covers projects like documents and weekly task reports.



Name	Begin date	End date
Parth Patel	2/19/18	2/19/18
Jacob Serafin	2/19/18	2/19/18
Steven Strickland	2/19/18	2/19/18
Abraham Ramirez	2/19/18	2/19/18
XML PARSER	2/6/18	2/28/18
AWS INTEGRATION	2/6/18	3/15/18
GUI INTEGRATION	2/6/18	2/21/18
XML FILE CREATION	2/6/18	2/21/18
DOCUMENTS TEAM	2/6/18	4/19/18
AWS TEAM	2/6/18	4/19/18
HOS to PBX	2/22/18	4/19/18
HOS to PBX	2/22/18	4/19/18
PBX to HOS	2/22/18	4/19/18
PBX to HOS	2/22/18	4/19/18

Conclusion

Overall, our vision for our design is to seamlessly implement a Cloud Architecture to replace an outdated Java application. Every job that this Java Application (J.E.D.I) does will be done by our web application. The advantages here include less software maintenance, a more cost effective solution that can help integrate newer technologies, and the ability to have all of SkyTouch utilities online. After our implementation, this solution can be increased to a higher scale and can increase the connectivity of each hotel to SkyTouch operating system. The hotel industry has many customers come in and each use case we will implement will show how the effectiveness of the web can help companies save data easier and promote a more seamless style of communication.