Playbook CMS Messaging via ActiveITS Databus

Andres Chavez

Andres.Chavez@dot.ca.gov

DRISI

D3 Central Systems & TMC Manager

Caltrans

Jared Sun

Jared.Sun@dot.ca.gov

District 3

Caltrans

Andres Chavez, P.E.

Senior Transportation Electrical Engineer Specialist

Experience

- 22 Years Network Engineer in Private Sector
- 7 Years Caltrans D3 - DRISI

Fun Facts:

- Goal, hike all 63 national parks, 1/3 done.
- I'm a recovering health-nut.
- Wind therapy is the best therapy.

About Me

- Jared Sun, P.E.
- Caltrans District 3
- Formerly at CA Dept. of Water Resources (DWR)
- Professional Interests: web development, security, data science, AI/ML

The Shredder!

Glossary

- **ActiveITS – Southwest Research Institute's ATMS.**
- API: Application Programming Interface
- ATMS – Advanced Transportation Management System.
- **CMS – Changeable Message Sign.**
- **EMS – Extinguishable Message Sign.**
- **GoldenEye – Caltrans's version of ActiveITS.**
- **HAR – Highway Advisory Radio.**
- ICM – Intelligent Corridor Management.
- JavaScript – Object oriented programming language.
- JSON – JavaScript Object Notation.
- JWT: JSON Web Token
- Sun Guide – Florida DOT's version of ActiveITS.
- XML – Extensible Markup Language.

Agenda

- The problem.
- The working solution.
- Quick demo.
- Nuts and bolts.
- What’s next.

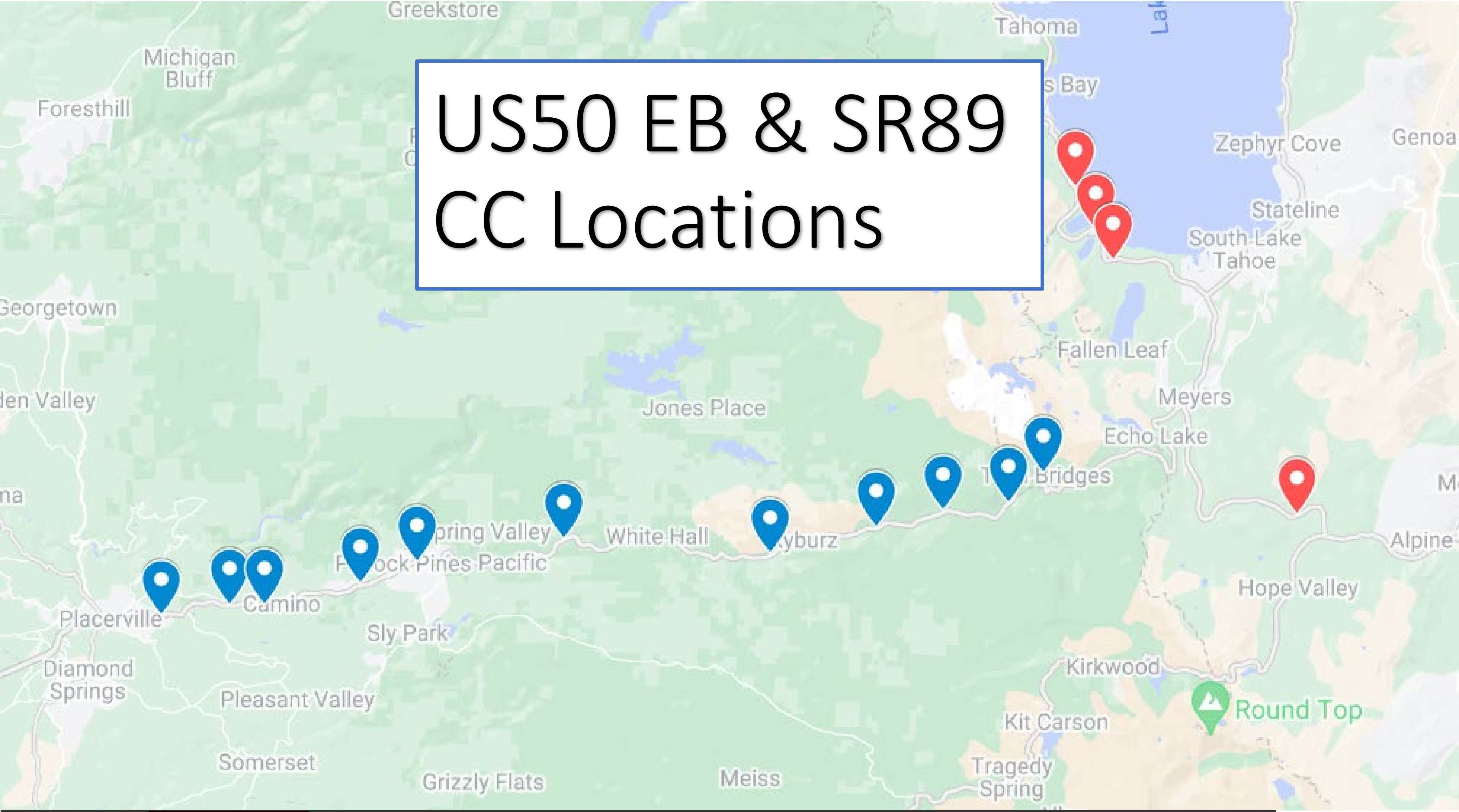
The Problem Statement

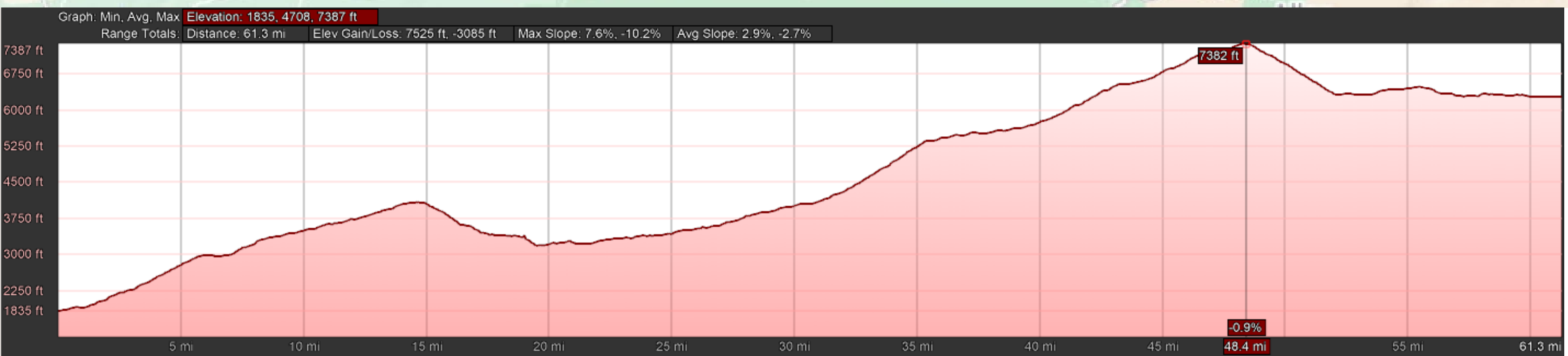
CMS Chain Control Messaging is a manual process that requires constant message updates as weather condition change. These changes can happen faster than the time it takes an operator to update CMS, HAR's and Event Logging System.

CMS Message Requirements

- Correctness!
- Consistent (over time).
- Deployed quickly.
- **Updated/modified rapidly.**
- Removed in a timely manner.

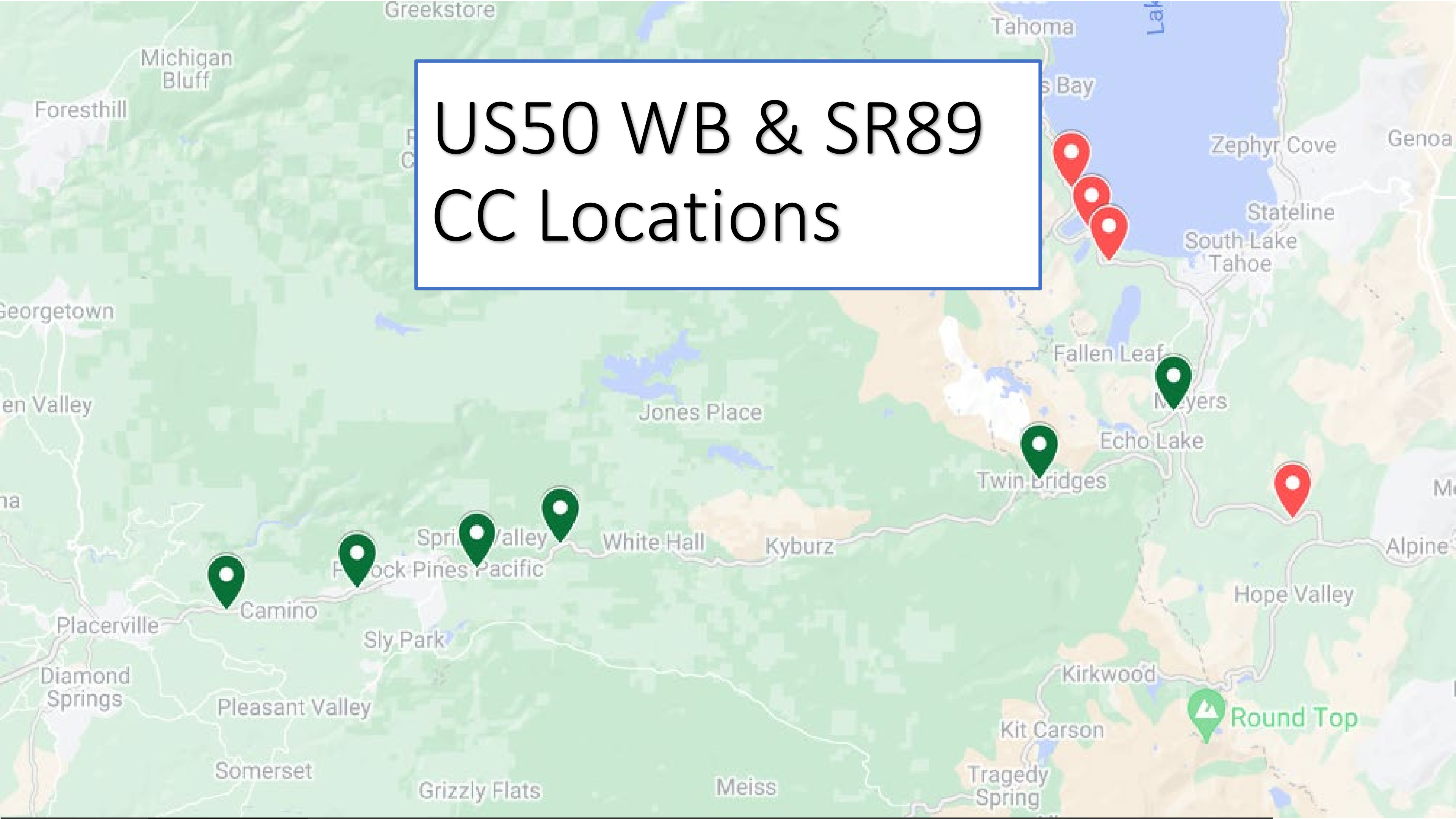
US50 EB & SR89 CC Locations

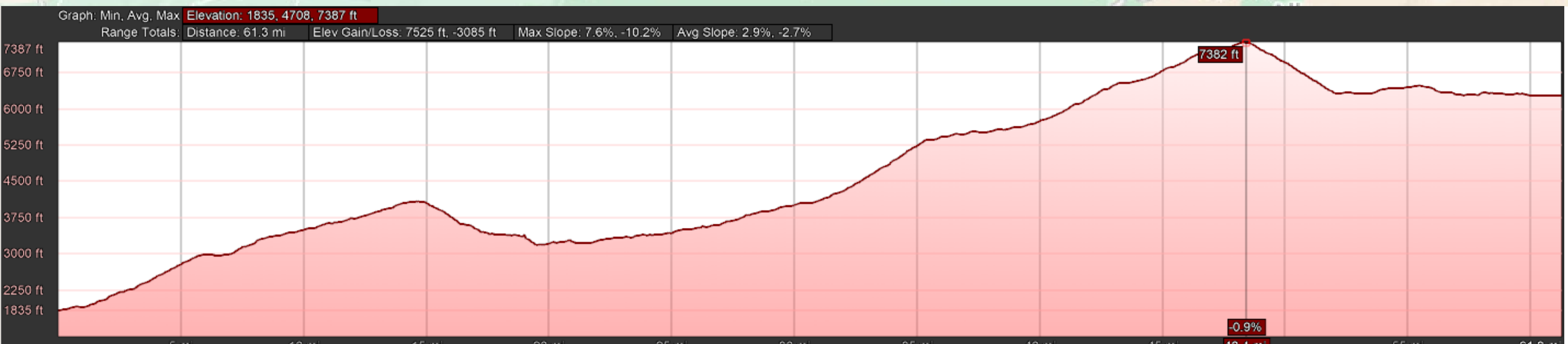



Graph: Min, Avg, Max Elevation: 1835, 4708, 7387 ft

Range Totals: Distance: 61.3 mi | Elev Gain/Loss: 7525 ft, -3085 ft | Max Slope: 7.6%, -10.2% | Avg Slope: 2.9%, -2.7%

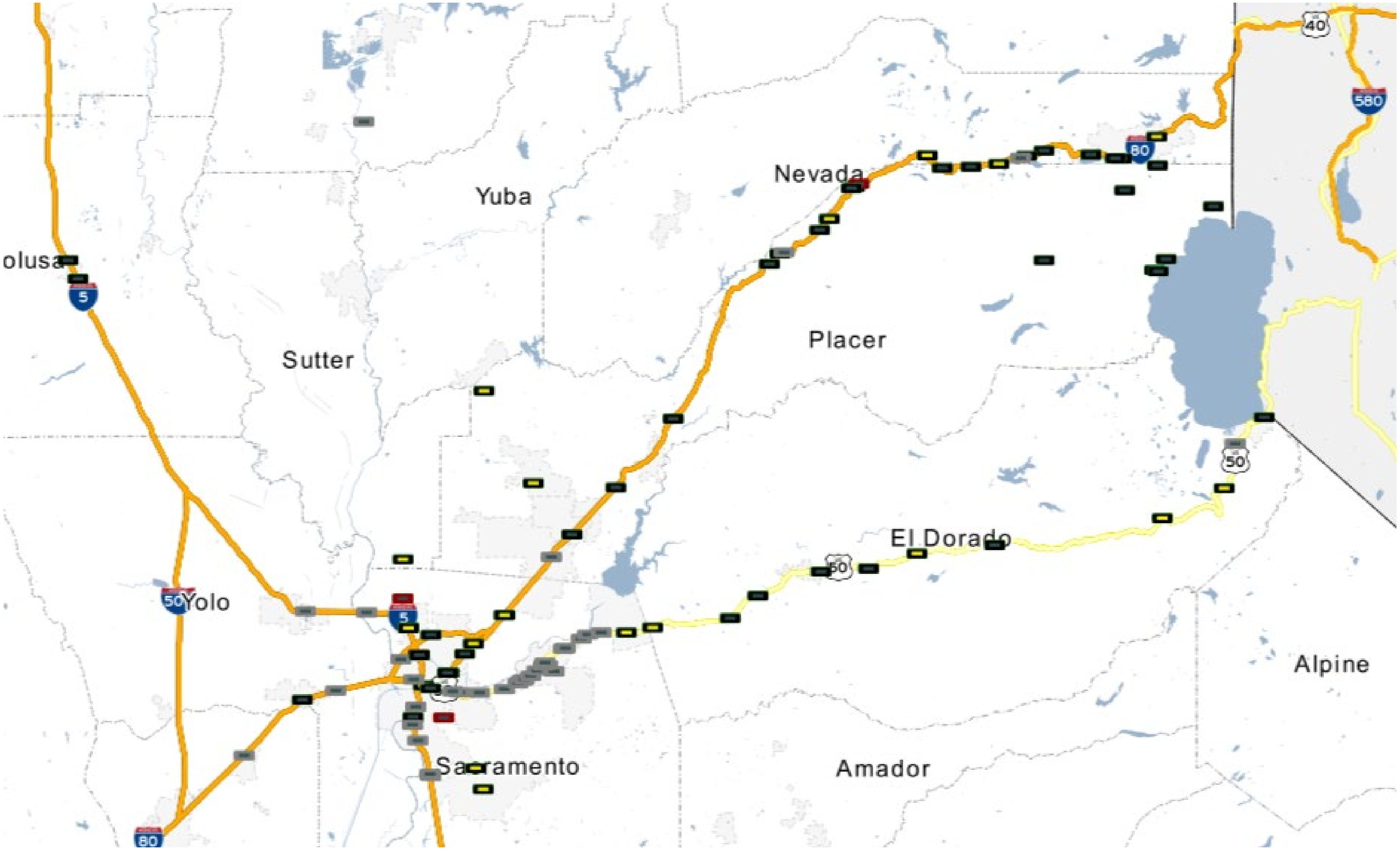
US50 WB & SR89 CC Locations

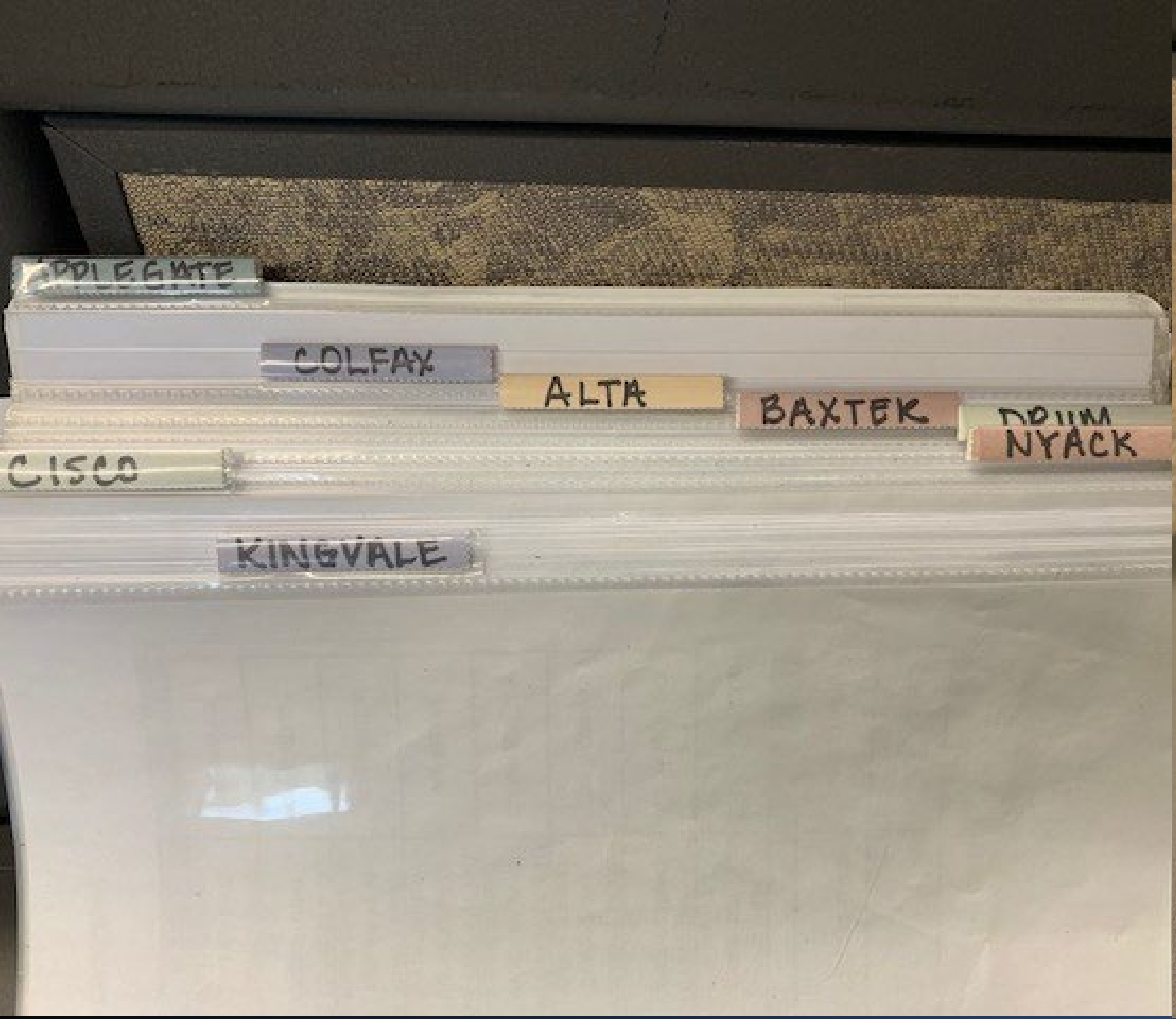



Graph: Min, Avg, Max | Elevation: 1835, 4708, 7387 ft

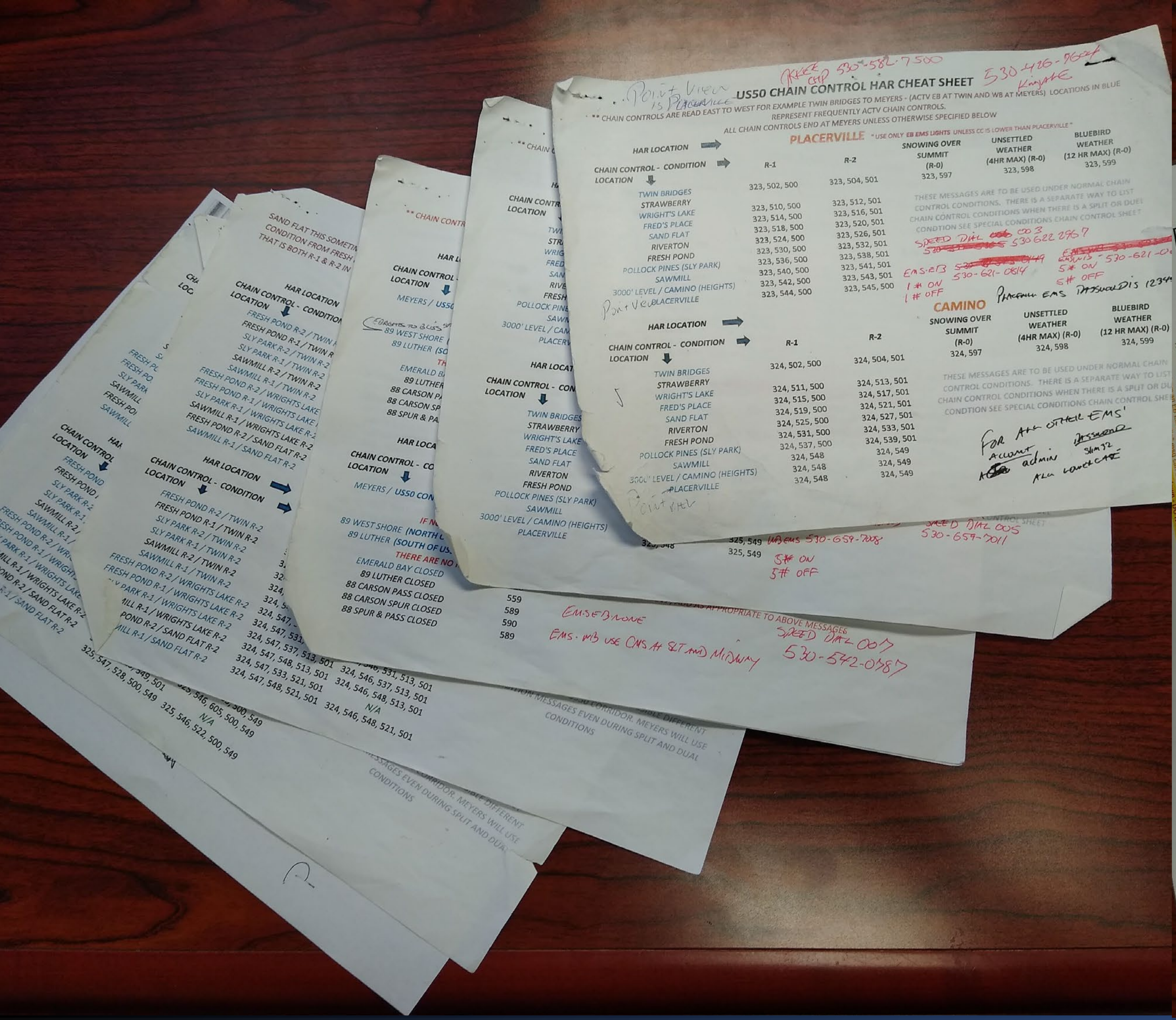
Range Totals: | Distance: 61.3 mi | Elev Gain/Loss: 7525 ft, -3085 ft | Max Slope: 7.6%, -10.2% | Avg Slope: 2.9%, -2.7%

CMS Locations





Message Sources



.. Point View is Placerville

CHP 530-582-7500

530-426-9604 Kingvale

US50 CHAIN CONTROL HAR CHEAT SHEET

** CHAIN CONTROLS ARE READ EAST TO WEST FOR EXAMPLE TWIN BRIDGES TO MEYERS - (ACTV EB AT TWIN AND WB AT MEYERS) LOCATIONS IN BLUE REPRESENT FREQUENTLY ACTV CHAIN CONTROLS.

ALL CHAIN CONTROLS END AT MEYERS UNLESS OTHERWISE SPECIFIED BELOW

PLACERVILLE "USE ONLY EB EMS LIGHTS UNLESS CC IS LOWER THAN PLACERVILLE"

HAR LOCATION → CHAIN CONTROL - CONDITION → LOCATION ↓	R-1	R-2	SNOWING OVER SUMMIT (R-0) 323, 597	UNSETTLED WEATHER (4HR MAX) (R-0) 323, 598	BLUEBIRD WEATHER (12 HR MAX) (R-0) 323, 599
TWIN BRIDGES	323, 502, 500	323, 504, 501			
STRAWBERRY	323, 510, 500	323, 512, 501			
WRIGHT'S LAKE	323, 514, 500	323, 516, 501			
FRED'S PLACE	323, 518, 500	323, 520, 501			
SAND FLAT	323, 524, 500	323, 526, 501			
RIVERTON	323, 530, 500	323, 532, 501			
FRESH POND	323, 536, 500	323, 538, 501			
POLLOCK PINES (SLY PARK)	323, 540, 500	323, 541, 501			
SAWMILL	323, 542, 500	323, 543, 501			
3000' LEVEL / CAMINO (HEIGHTS)	323, 544, 500	323, 545, 500			
Point View PLACERVILLE					

THESE MESSAGES ARE TO BE USED UNDER NORMAL CHAIN CONTROL CONDITIONS. THERE IS A SEPARATE WAY TO LIST CHAIN CONTROL CONDITIONS WHEN THERE IS A SPLIT OR DUEL CONDTION SEE SPECIAL CONDITIONS CHAIN CONTROL SHEET

SPEED DIAL ~~[illegible]~~ 003 ~~530-[illegible]~~ 530 622 2967

EMS-EB ~~530-[illegible]-0149~~ 530-621-0814 1* ON 1# OFF

~~EMS-WB 530-[illegible]~~ EMS-WB - 530-621-0[illegible] 5* ON 5# OFF

Placerville EMS Password is 1234

CAMINO

HAR LOCATION → CHAIN CONTROL - CONDITION → LOCATION ↓	R-1	R-2	SNOWING OVER SUMMIT (R-0) 324, 597	UNSETTLED WEATHER (4HR MAX) (R-0) 324, 598	BLUEBIRD WEATHER (12 HR MAX) (R-0) 324, 599
TWIN BRIDGES	324, 502, 500	324, 504, 501			
STRAWBERRY	324, 511, 500	324, 513, 501			
WRIGHT'S LAKE	324, 515, 500	324, 517, 501			
FRED'S PLACE	324, 519, 500	324, 521, 501			
SAND FLAT	324, 525, 500	324, 527, 501			
RIVERTON	324, 531, 500	324, 533, 501			
FRESH POND	324, 537, 500	324, 539, 501			
POLLOCK PINES (SLY PARK)	324, 548	324, 549			
SAWMILL	324, 548	324, 549			
3000' LEVEL / CAMINO (HEIGHTS)	324, 548	324, 549			
Point View PLACERVILLE					

THESE MESSAGES ARE TO BE USED UNDER NORMAL CHAIN CONTROL CONDITIONS. THERE IS A SEPARATE WAY TO LIST CHAIN CONTROL CONDITIONS WHEN THERE IS A SPLIT OR DU[illegible] CHAIN CONTROL CONDITIONS WHEN THERE IS A SPLIT OR DU CHAIN CONTROL SHE[illegible] CONDTION SEE SPECIAL CONDITIONS CHAIN CONTROL SHE[illegible]

For All other EMS'
Account admin
Password Slim32
All lowercase

325, 549 WB EMS 530-659-7008
325, 549 5* ON 5# OFF

SPEED DIAL 005 530-659-7011

EMS EB none
EMS-WB use CMS at SLT and Midway

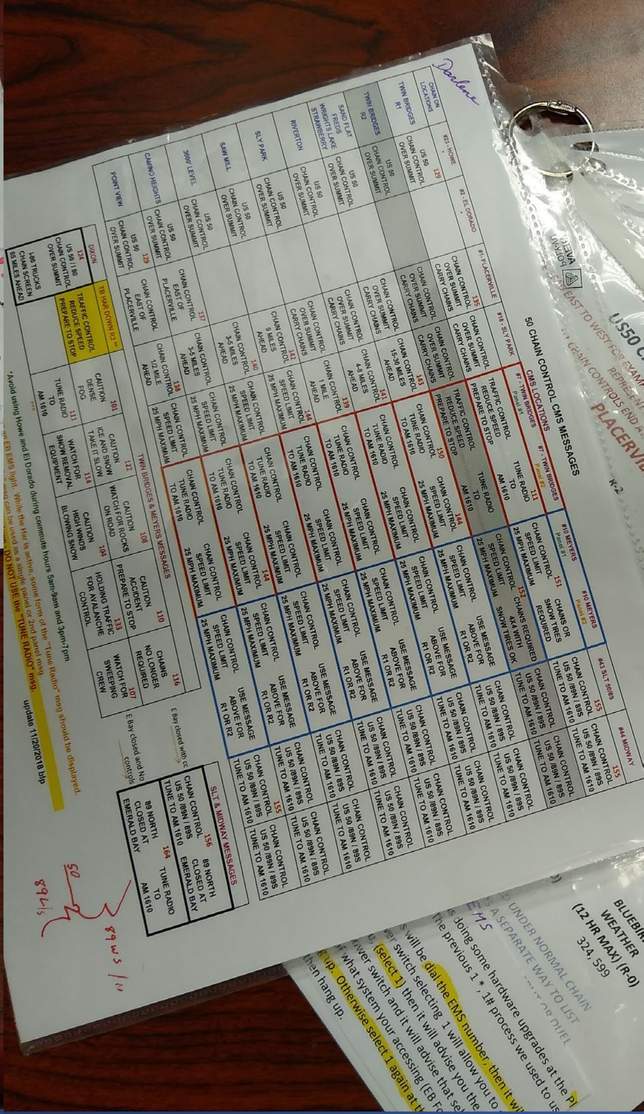
SPEED DIAL 007 530-542-0987

...APPROPRIATE TO ABOVE MESSAGES

SAND FLAT THIS SOMETIME CONDITION FROM FRESH THAT IS BOTH R-1 & R-2 IN

89 WEST SHORE (NORTH OF...)
89 LUTHER (SOUTH OF US...)
THERE ARE NO...
EMERALD BAY CLOSED
89 LUTHER CLOSED
88 CARSON PASS CLOSED
88 CARSON SPUR CLOSED
88 SPUR & PASS CLOSED

...CORRIDOR. MEYERS WILL USE ... MESSAGES EVEN DURING SPLIT AND DUAL CONDITIONS



US 50 CHAIN CONTROL CMS MESSAGES

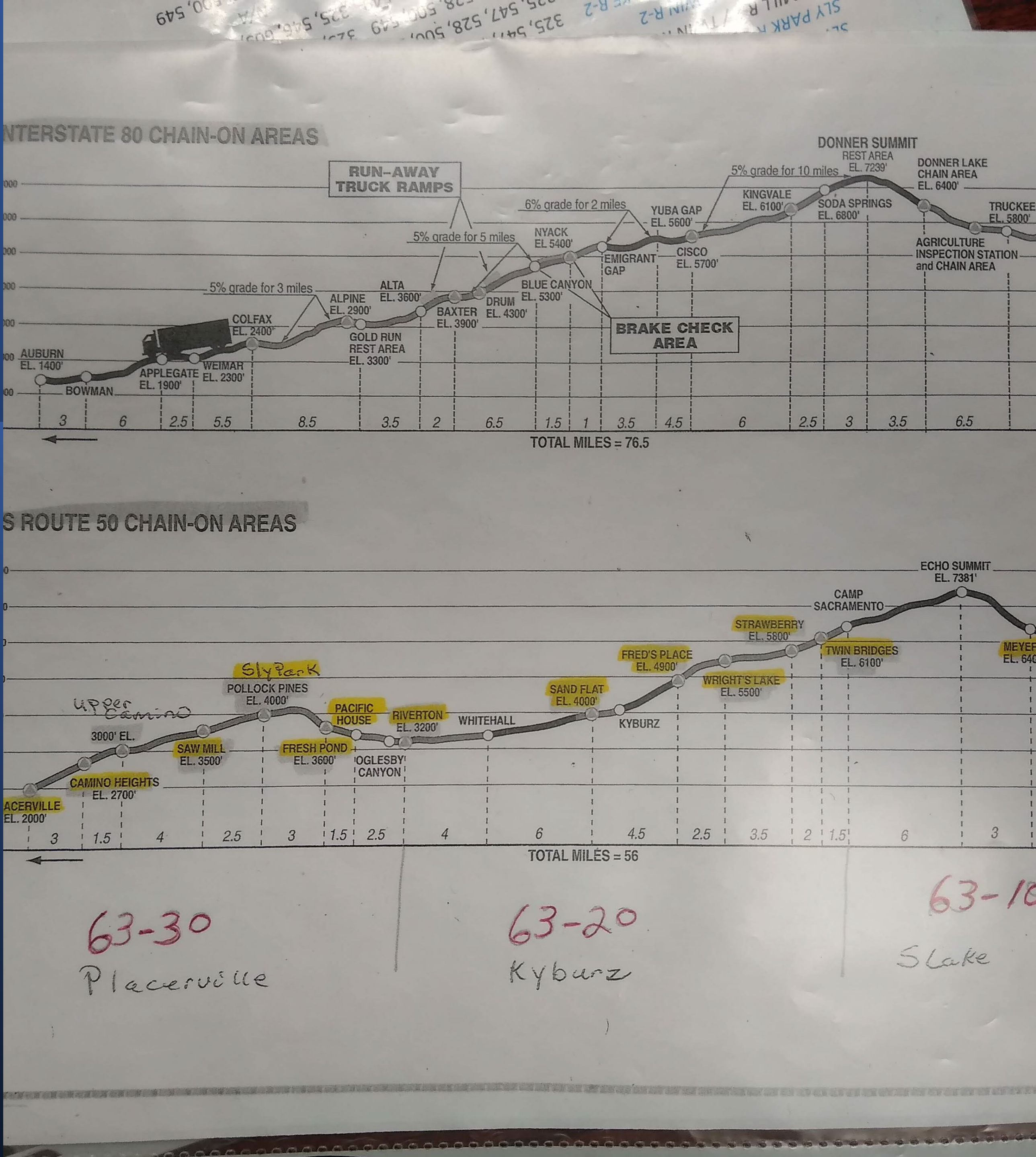
*Avoid using Howe and El Dorado during commute hours 5am-9am and 3pm-7pm

While the Har is active, some form of the "Tune Radio" msg should be displayed ... DO NOT USE the "TUNE RADIO" msg.

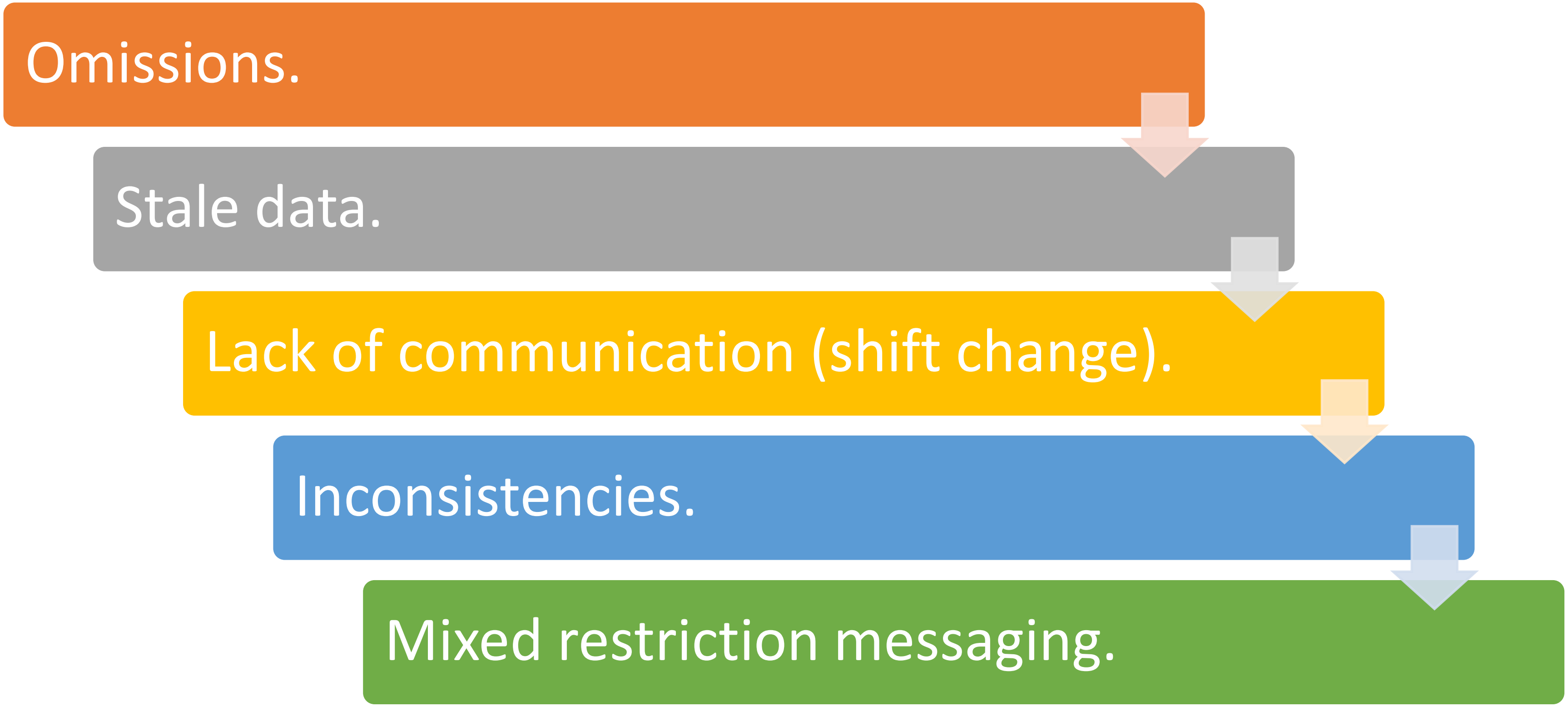
update 11/20/2018 blp

Message Sources

Message Sources – Split Notes



What Could Go Wrong?



Omissions.

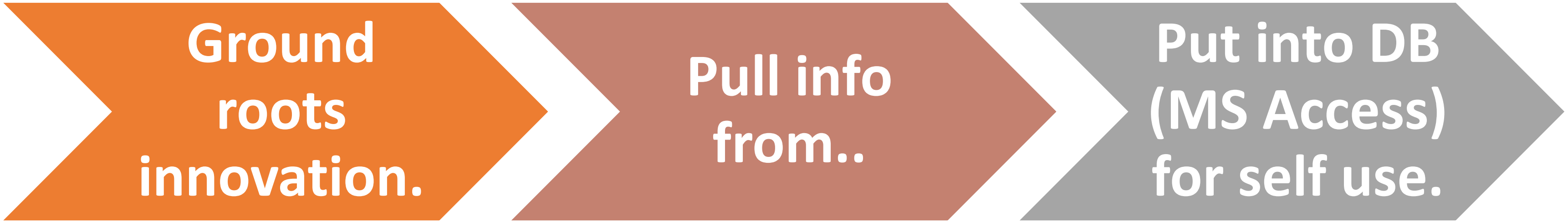
Stale data.

Lack of communication (shift change).

Inconsistencies.

Mixed restriction messaging.

The Solution - Gather Data



Ground roots innovation.

Pull info from..

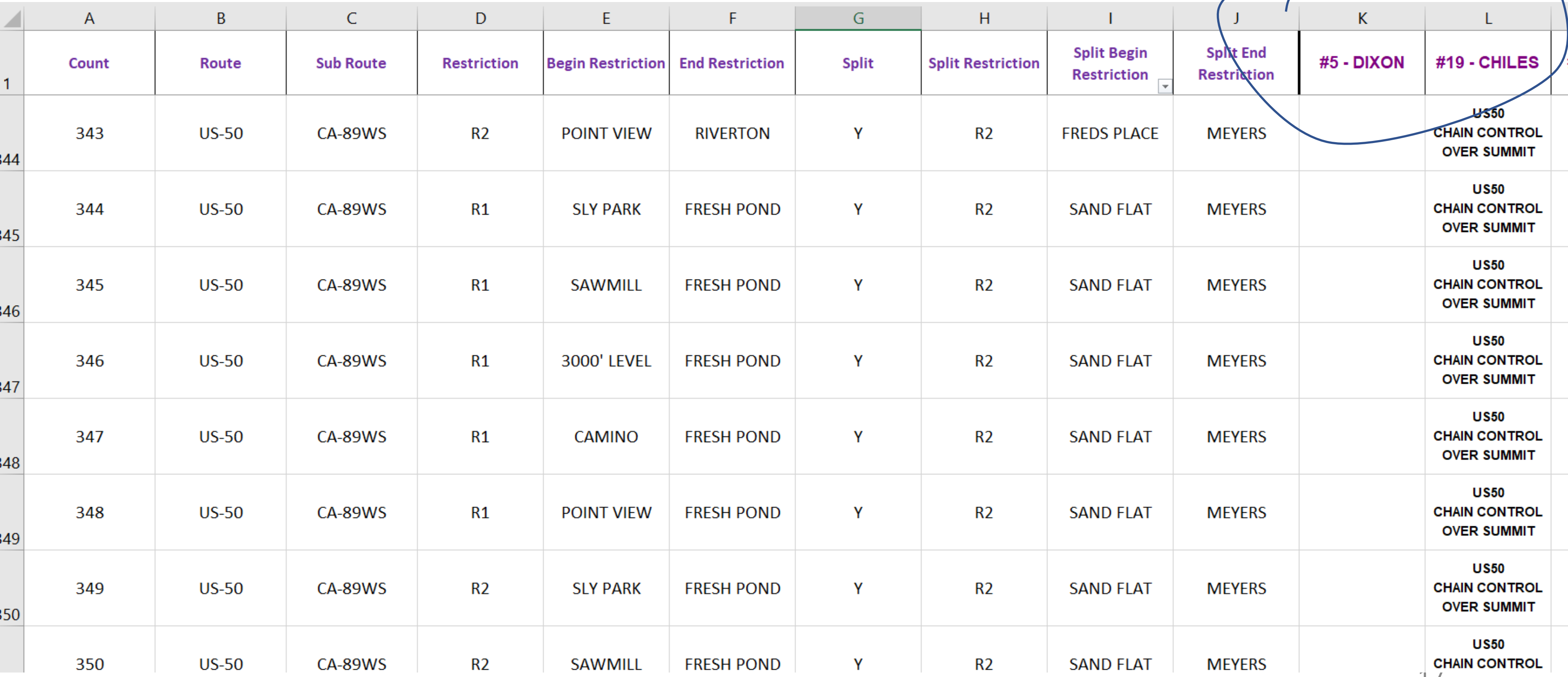
Put into DB (MS Access) for self use.

- Binders
- Leaflets
- Personal notebooks
- Operators' brains

Standardize And Populate Playbook

- Identify Variables
  - Route
  - Sub Route (optional)
  - Restriction
    - R1; Chains or snow tires
    - R2; Chains or 4x4/AWD
  - Begin Restriction
  - End Restriction
  - Split (optional)

First Pass
(US50 and SR89)
483 Permutation

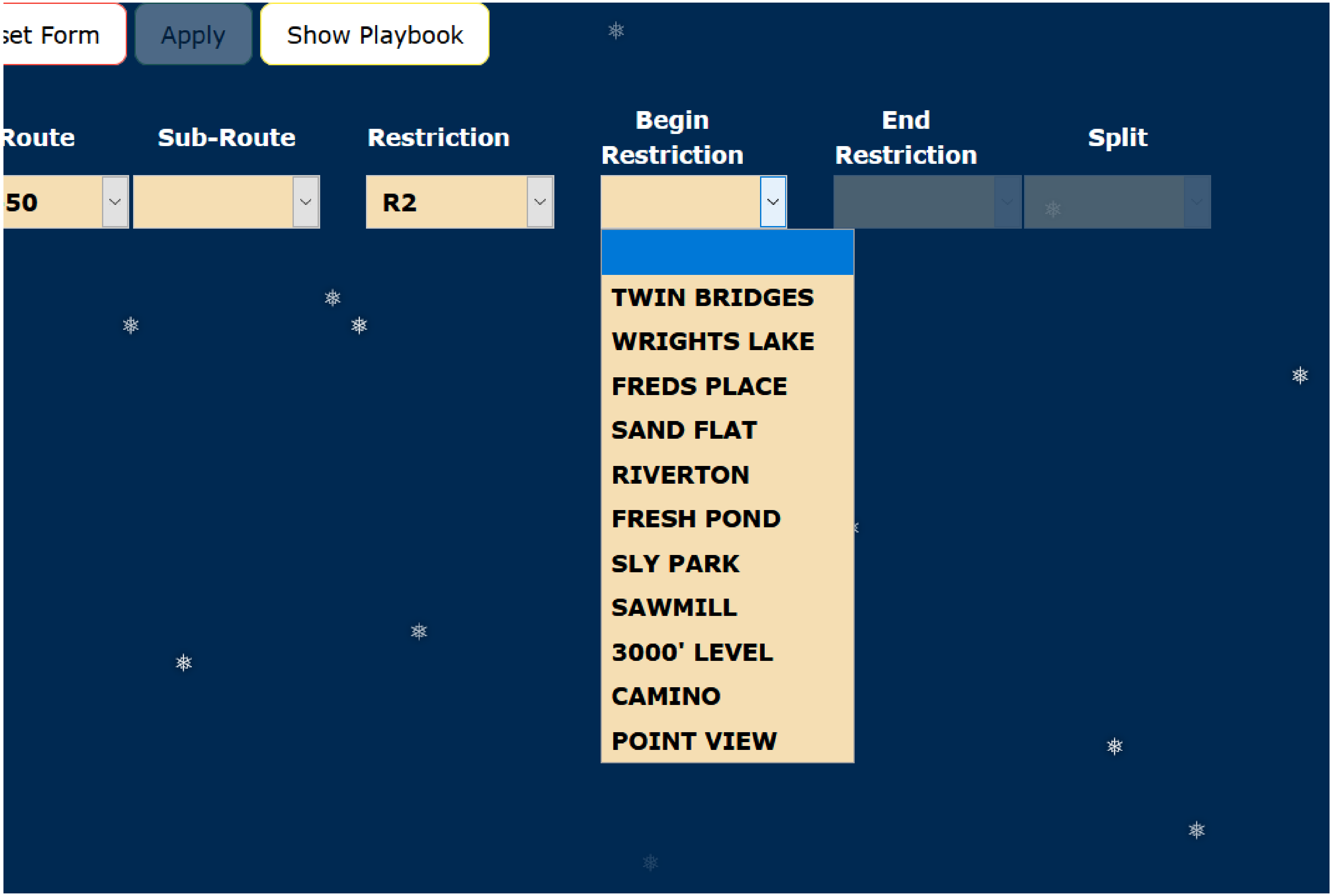


	A	B	C	D	E	F	G	H	I	J	K	L
1	Count	Route	Sub Route	Restriction	Begin Restriction	End Restriction	Split	Split Restriction	Split Begin Restriction	Split End Restriction	#5 - DIXON	#19 - CHILES
44	343	US-50	CA-89WS	R2	POINT VIEW	RIVERTON	Y	R2	FREDS PLACE	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
45	344	US-50	CA-89WS	R1	SLY PARK	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
46	345	US-50	CA-89WS	R1	SAWMILL	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
47	346	US-50	CA-89WS	R1	3000' LEVEL	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
48	347	US-50	CA-89WS	R1	CAMINO	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
49	348	US-50	CA-89WS	R1	POINT VIEW	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
50	349	US-50	CA-89WS	R2	SLY PARK	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL OVER SUMMIT
	350	US-50	CA-89WS	R2	SAWMILL	FRESH POND	Y	R2	SAND FLAT	MEYERS		US50 CHAIN CONTROL

Streamline/Automate

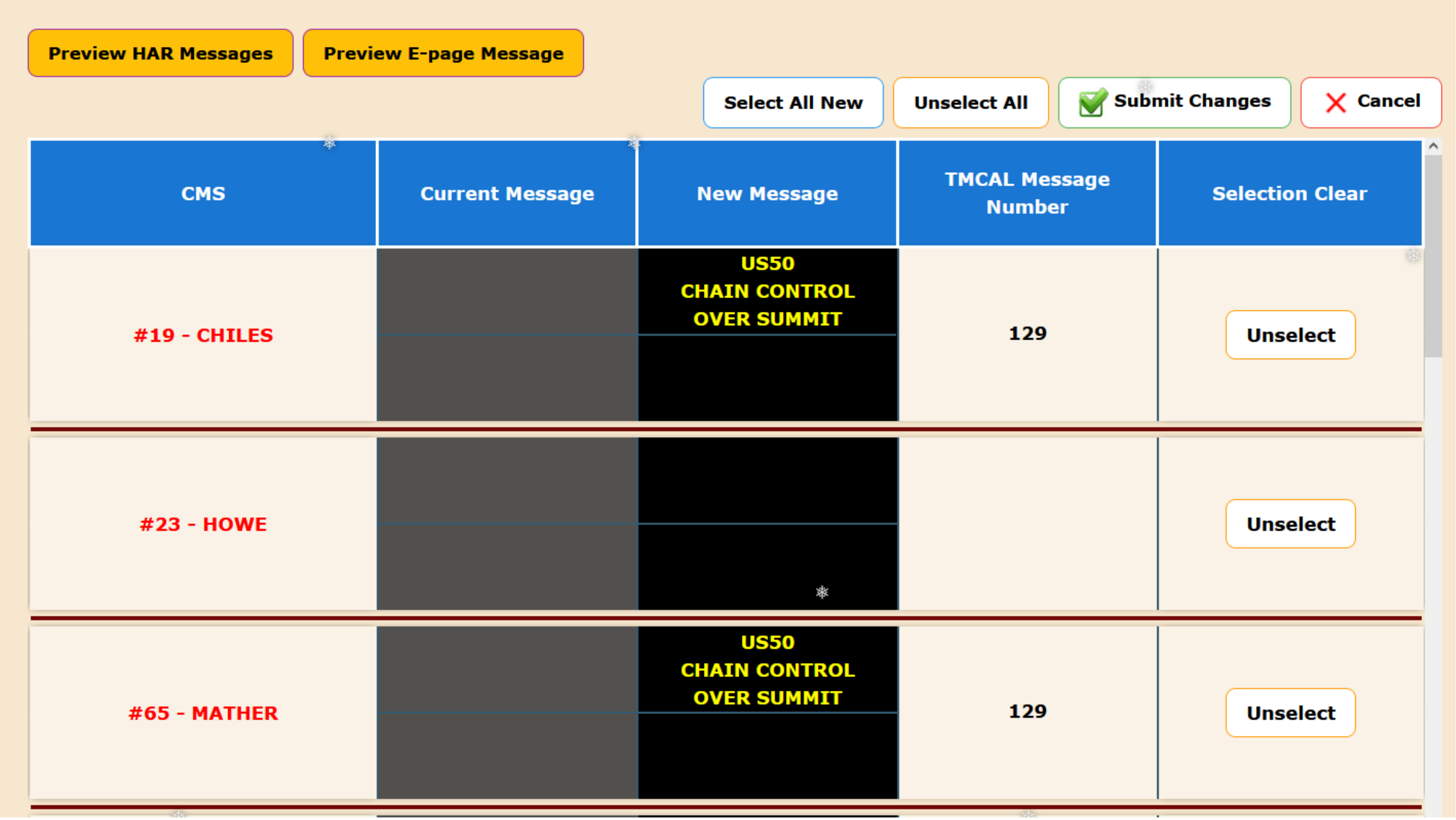
- Original goal – Gather info into searchable webpage. But what if…
  - Pushed messages to CMS's?
  - Activated HAR?
  - Activated EMS?
  - (Streamlined E-Pages?)
- Match made in heaven, enter ActiveITS from SwRI
  - Databus XML API.
  - Automatic logging.

Chain Control (First Release) Input



Route	Sub-Route	Restriction	Begin Restriction	End Restriction	Split
50		R2			

Chain Control Output



CMS	Current Message	New Message	TMCAL Message Number	Selection Clear
#19 - CHILES		US50 CHAIN CONTROL OVER SUMMIT	129	Unselect
#23 - HOWE				Unselect
#65 - MATHER		US50 CHAIN CONTROL OVER SUMMIT	129	Unselect

Why Stop There?

Add other chain-controlled routes

- I80
- I80 EB
- I80 WB
- CA 20
- CA 28
- CA 267

Add other chain control restrictions

- MIN; R2 + Minimum requirements for big rigs.
- MAX; R2 + Maximum requirements for big rigs.

Permutation count more than doubled.

Don’t Let A Catastrophe Go To Waste.

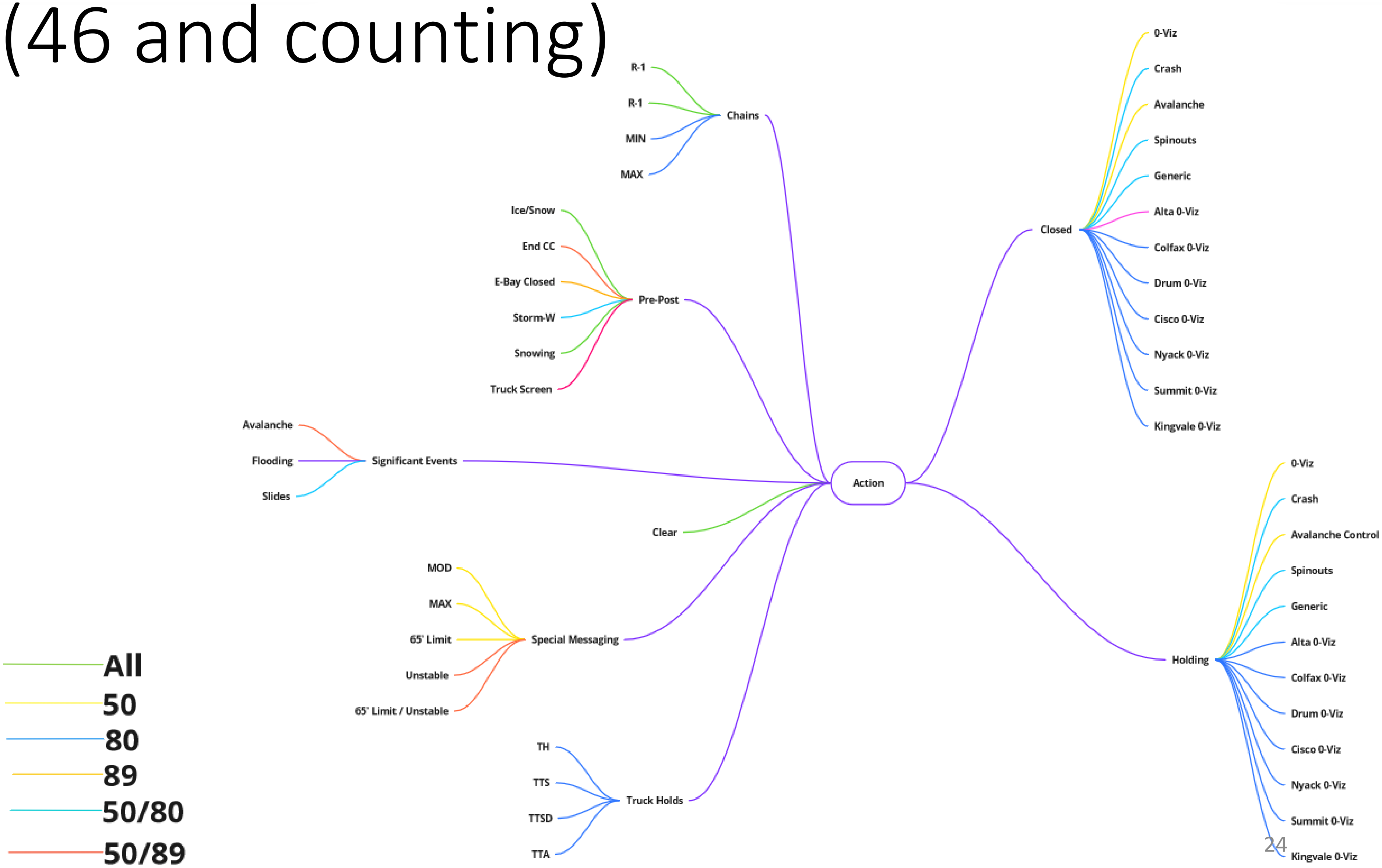
- Add Special (Recurring) Events
- (Snow) Avalanche
- (Mud) Slides
- Storm Warning
- Floods

Augmented Playbook Parameters

- New Variables
  - Action
  - Route
  - Sub Route
    - Route Dependent
  - Restriction
    - 46 total
    - Action Dependent
  - Begin Restriction
  - End Restriction
  - Split
    - Route Dependent

Permutation count ~2500.

Restriction Tree (46 and counting)



Action

- Chains
  - R-1
  - R-1
  - MIN
  - MAX
- Pre-Post
  - Ice/Snow
  - End CC
  - E-Bay Closed
  - Storm-W
  - Snowing
  - Truck Screen
- Significant Events
  - Avalanche
  - Flooding
  - Slides
- Clear
- Special Messaging
  - MOD
  - MAX
  - 65' Limit
  - Unstable
  - 65' Limit / Unstable
- Truck Holds
  - TH
  - TTS
  - TTSD
  - TTA
- Closed
  - 0-Viz
  - Crash
  - Avalanche
  - Spinouts
  - Generic
  - Alta 0-Viz
  - Colfax 0-Viz
  - Drum 0-Viz
  - Cisco 0-Viz
  - Nyack 0-Viz
  - Summit 0-Viz
  - Kingvale 0-Viz
- Holding
  - 0-Viz
  - Crash
  - Avalanche Control
  - Spinouts
  - Generic
  - Alta 0-Viz
  - Colfax 0-Viz
  - Drum 0-Viz
  - Cisco 0-Viz
  - Nyack 0-Viz
  - Summit 0-Viz
  - Kingvale 0-Viz

All

50

80

89

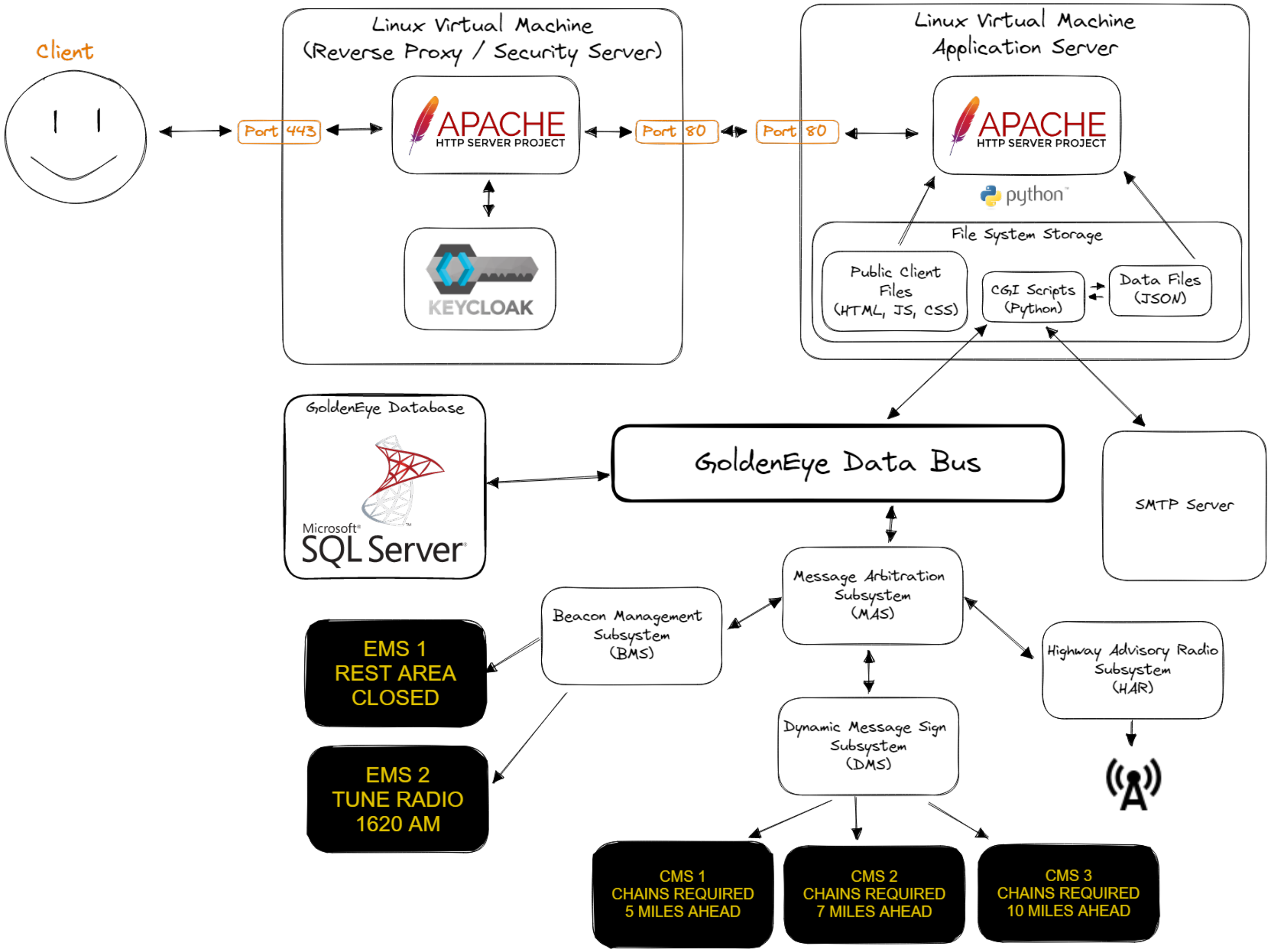
50/80

50/89

Demo

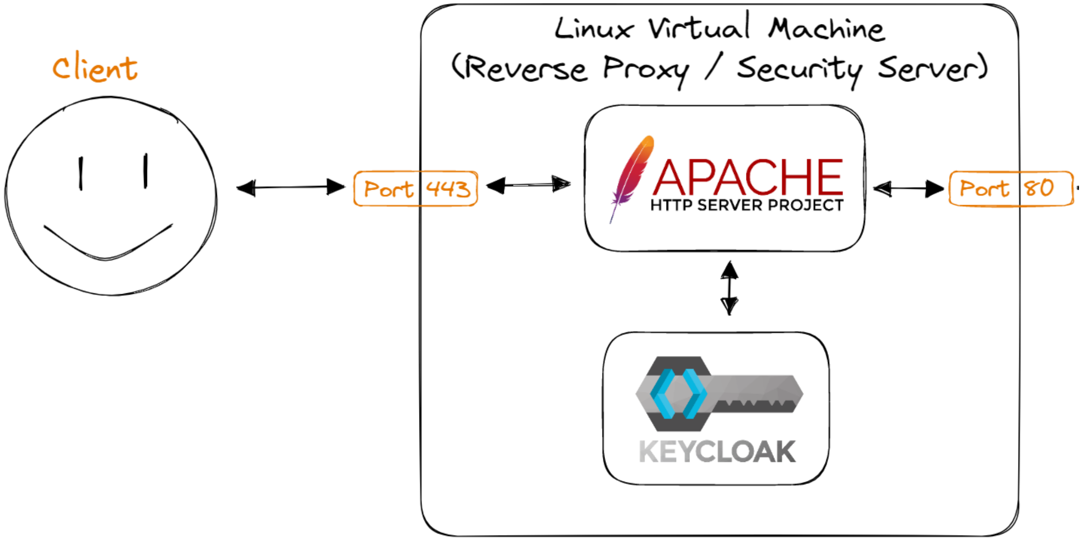
Chain Control System: The Nuts and Bolts

Architecture Overview



Component 1: Security Server

- **Apache HTTP Server**: the tried-and-true server for web applications
  - Extensible with modules
- **Keycloak**: a free, open-source Identity and Access Management solution
  - allows users to log in with a single-sign-on account and have specific access levels based on their assigned roles.



Apache HTTP Server

- Serves static files as fundamental functionality
- Can execute dynamic scripts via Common Gateway Interface (CGI) module mod_cgi
    - A little slow and “old-school” (popular in 1990s)
- Can reverse-proxy to origin application servers via mod_proxy
    - Used for load balancing and security
- Well-known and reliable, though it was dethroned by Nginx in 2022 as most used web server

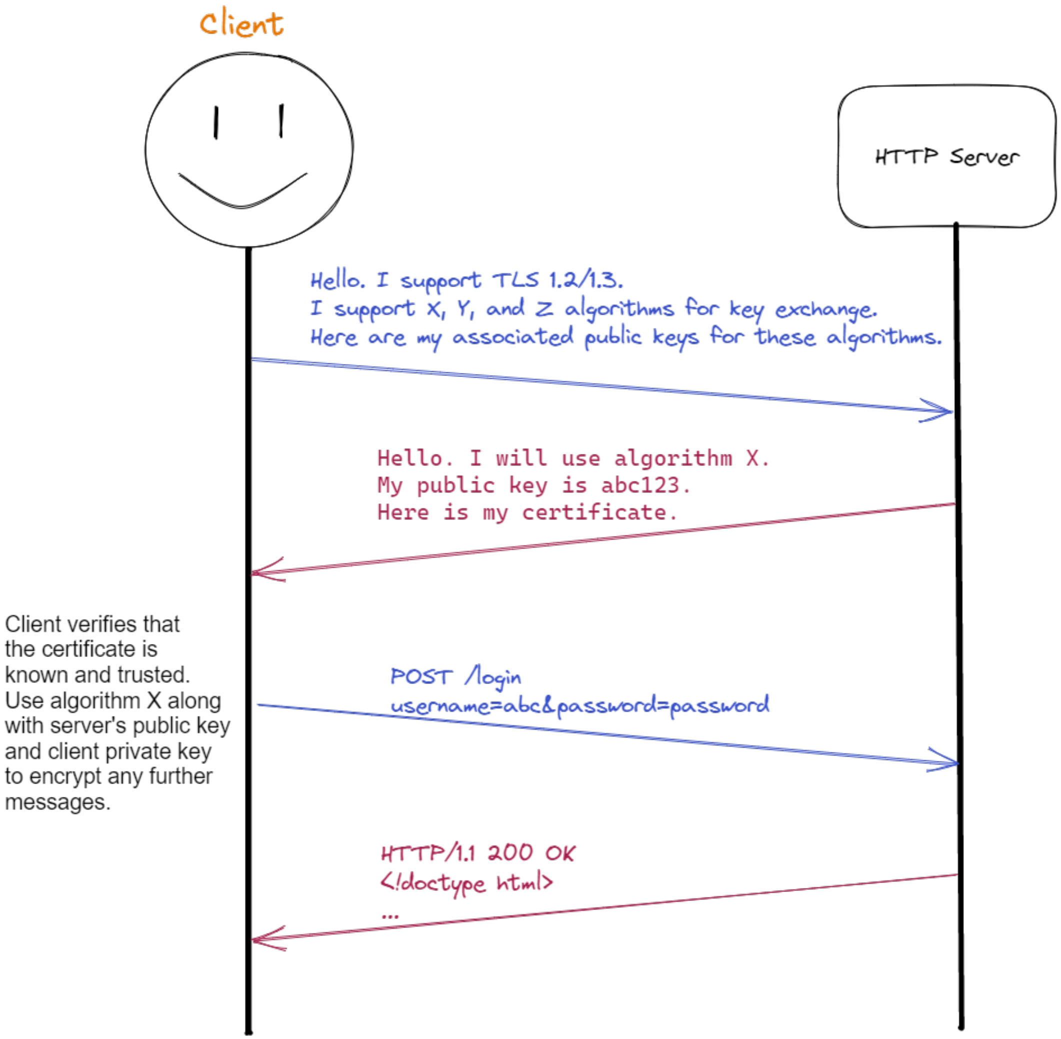
HTTP Fundamentals

- HTTP = Hypertext Transfer Protocol
- Serves as the most common “glue” between clients and servers, as well as servers to other servers
- Client sends a **request method** to indicate the desired action.
    - GET (request a resource)
    - POST (submit data, often as a new resource)
    - DELETE (delete the specified resource)
    - PUT (modify the resource)
- Optionally contains cookies, storing session information about the user

HTTPS and TLS/SSL

- HTTP is **unencrypted** by default
    - Any passwords or confidential info sent over HTTP can be intercepted and read in plain text
- HTTPS = HTTP + Transport Layer Security (TLS), also formerly known as Secure Sockets Layer (SSL)
- In HTTPS, client and server perform a “handshake” routine to see if the client can trust the server and decide on how to encrypt their subsequent communication

TLS 1.3 Handshake for HTTPS



Hello. I support TLS 1.2/1.3.
I support X, Y, and Z algorithms for key exchange.
Here are my associated public keys for these algorithms.

Hello. I will use algorithm X.
My public key is abc123.
Here is my certificate.

Client verifies that the certificate is known and trusted. Use algorithm X along with server's public key and client private key to encrypt any further messages.

POST /login
username=abc&password=password

HTTP/1.1 200 OK
<!doctype html>
...

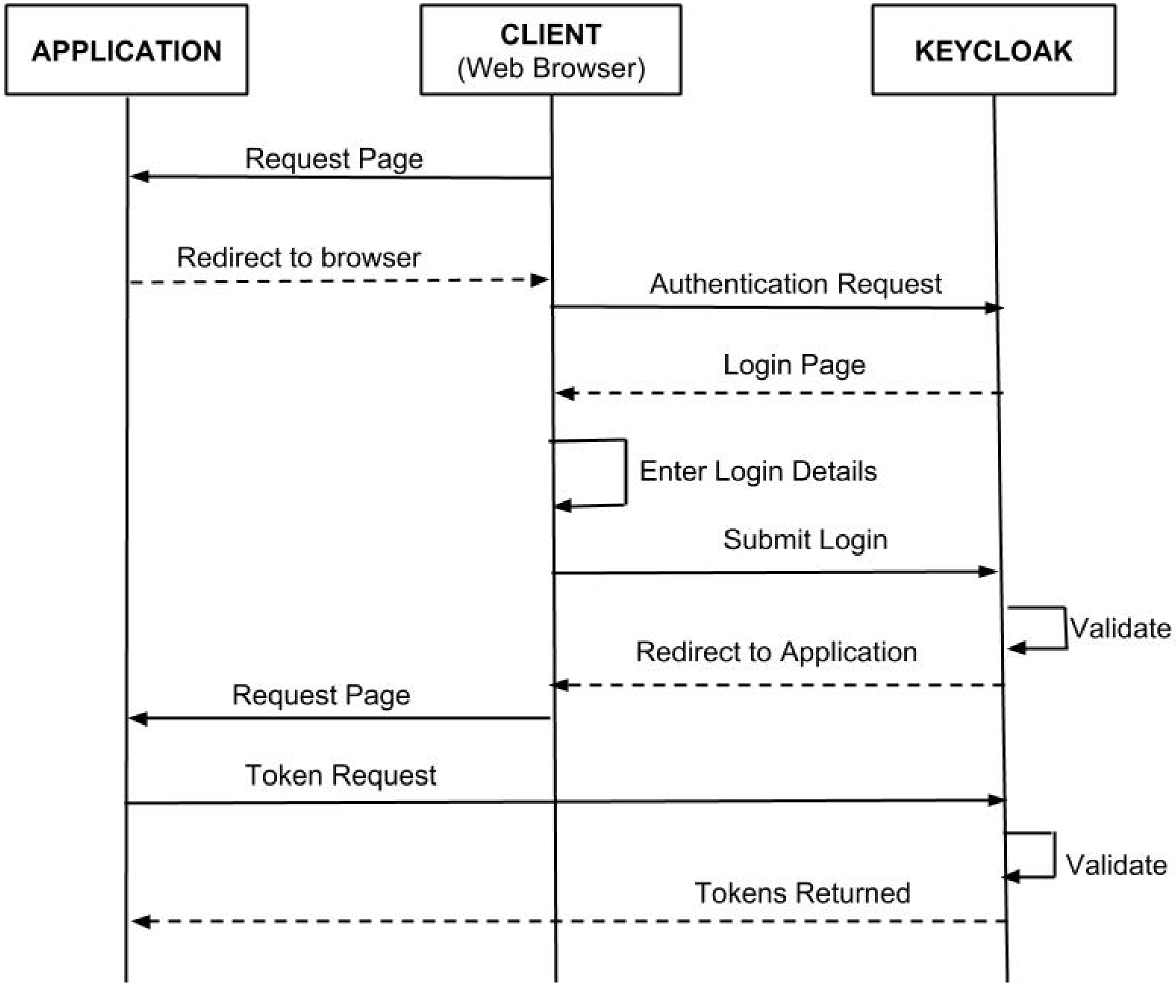
Security Part 2: Identity and Access Management (IAM)

- **Who** can perform **what actions** on **which resources** and **when?**
- Use OAuth 2.0 for authentication and authorization
  - Internet Standard RFC 6749
- Use OpenID Connect (OIDC) for identity
  - OIDC is a superset of Oauth 2.0

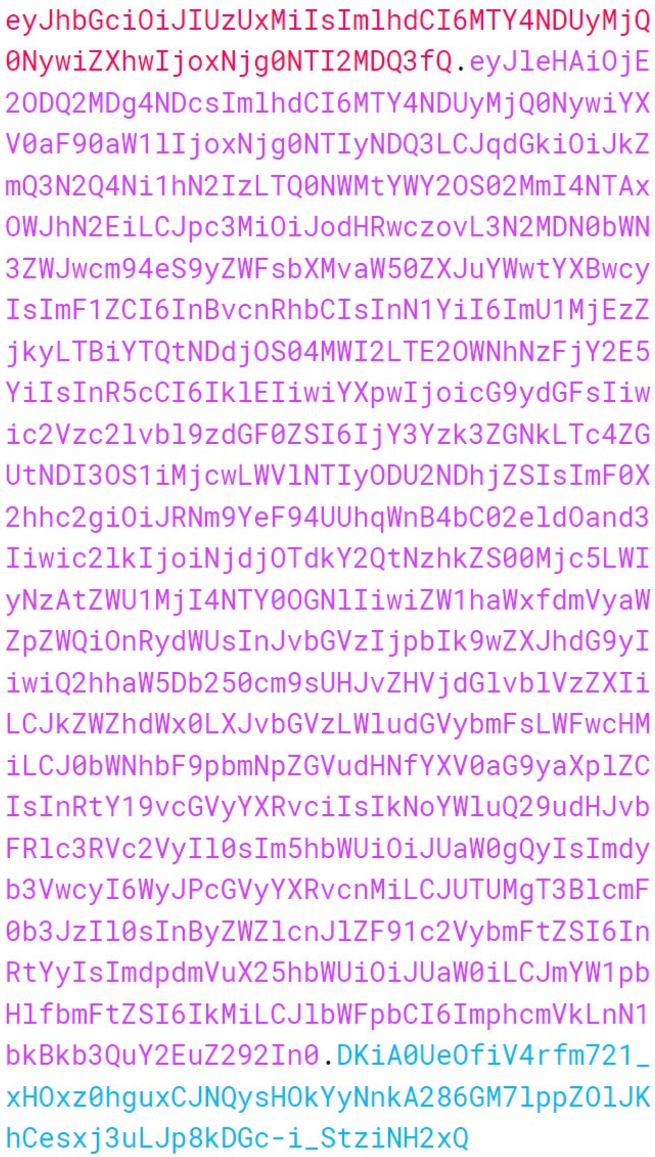
Keycloak

- Performs Identity and Access Management
- Allows single-sign-on
- Open source, self-hosted software under stewardship of Red Hat
- Assigns tokens to users over HTTP cookies with their user profile information, roles, and how long they have access
- Verifies cryptographic signatures in tokens to make sure clients have correct access

Authorization Flow

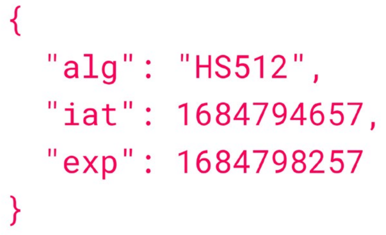


JSON Web Token (JWT)

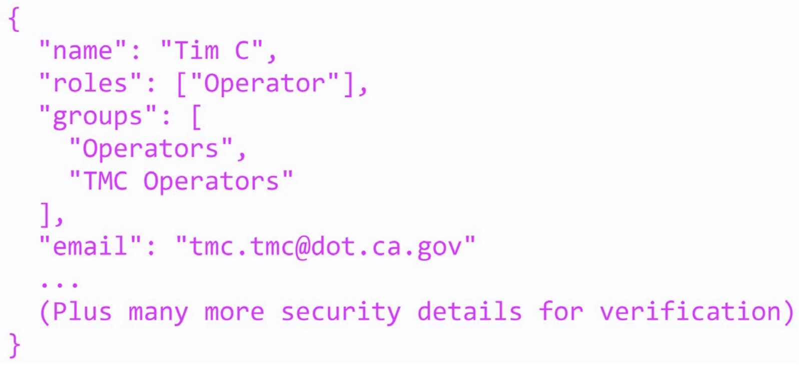


```
eyJhbGciOiJIUzUxMiIsImlhdCI6MTY4NDUyMjQ
0NywiZXhwIjoxNjg0NTI2MDQ3fQ.eyJleHAiOjE
2ODQ2MDg4NDcsImlhdCI6MTY4NDUyMjQ0NywiYX
V0aF90aW1lIjoxNjg0NTIyNDQ3LCJqdGkiOiJkZ
mQ3N2Q4Ni1hN2IzLTQ0NWMtYWY2OS02MmI4NTAx
OWJhN2EiLCJpc3MiOiJodHRwczovL3N2MDN0bWN
3ZWJwcm94eS9yZWFsbXMvaW50ZXJuYWwtYXBwcy
IsImF1ZCI6InBvcnRhbCIsInN1YiI6ImU1MjEzZ
jkyLTBiYTQtNDdjOS04MWI2LTE2OWNhNzFjY2E5
YiIsInR5cCI6IklEIiwiYXpwIjoicG9ydGFsIiw
ic2Vzc2lvbl9zdGF0ZSI6IjY3Yzk3ZGNkLTc4ZG
UtNDI3OS1iMjcwLWVlNTIyODU2NDhjZSIsImF0X
2hhc2giOiJRNm9YeF94UUhqWnB4bC02eldOand3
Iiwic2lkIjoiNjdjOTdkY2QtNzhkZS00Mjc5LWI
yNzAtZWU1MjI4NTY0OGNlIiwiZW1haWxfdmVyaW
ZpZWQiOnRydWUsInJvbGVzIjpbIk9wZXJhdG9yI
iwiQ2hhaW5Db250cm9sUHJvZHVjdGlvblVzZXIi
LCJkZWZhdWx0LXJvbGVzLWludGVybmFsLWFwcHM
iLCJ0bWNhbF9pbmNpZGVudHNfYXV0aG9yaXplZC
IsInRtY19vcGVyYXRvciIsIkNoYWluQ29udHJvb
FRlc3RVc2VyIl0sIm5hbWUiOiJUaW0gQyIsImdy
b3VwcyI6WyJPcGVyYXRvcnMiLCJUTUMgT3BlcmF
0b3JzIl0sInByZWZlcnJlZF91c2VybmFtZSI6In
RtYyIsImdpdmVuX25hbWUiOiJUaW0iLCJmYW1pb
HlfbmFtZSI6IkMiLCJlbWFpbCI6ImphcmVkLnN1
bkBkb3QuY2EuZ292In0.DKiA0UeOfiV4rfm721_
xHOxz0hguxCJNQysHOkYyNnkA286GM7lppZOlJK
hCesxj3uLJp8kDGc-i_StziNH2xQ
```

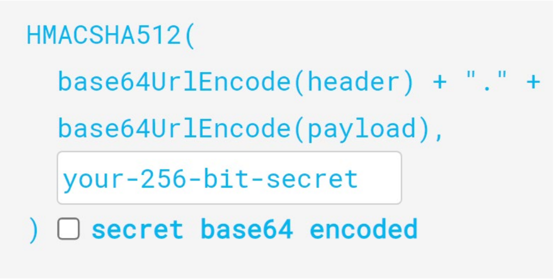
Decode



```
{
  "alg": "HS512",
  "iat": 1684794657,
  "exp": 1684798257
}
```



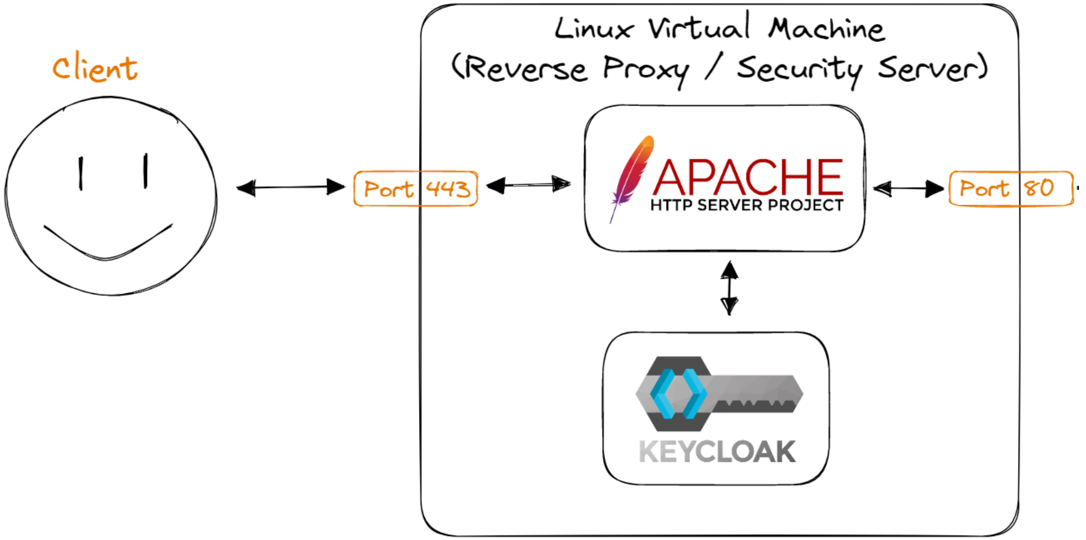
```
{
  "name": "Tim C",
  "roles": ["Operator"],
  "groups": [
    "Operators",
    "TMC Operators"
  ],
  "email": "tmc.tmc@dot.ca.gov"
  ...
  (Plus many more security details for verification)
}
```



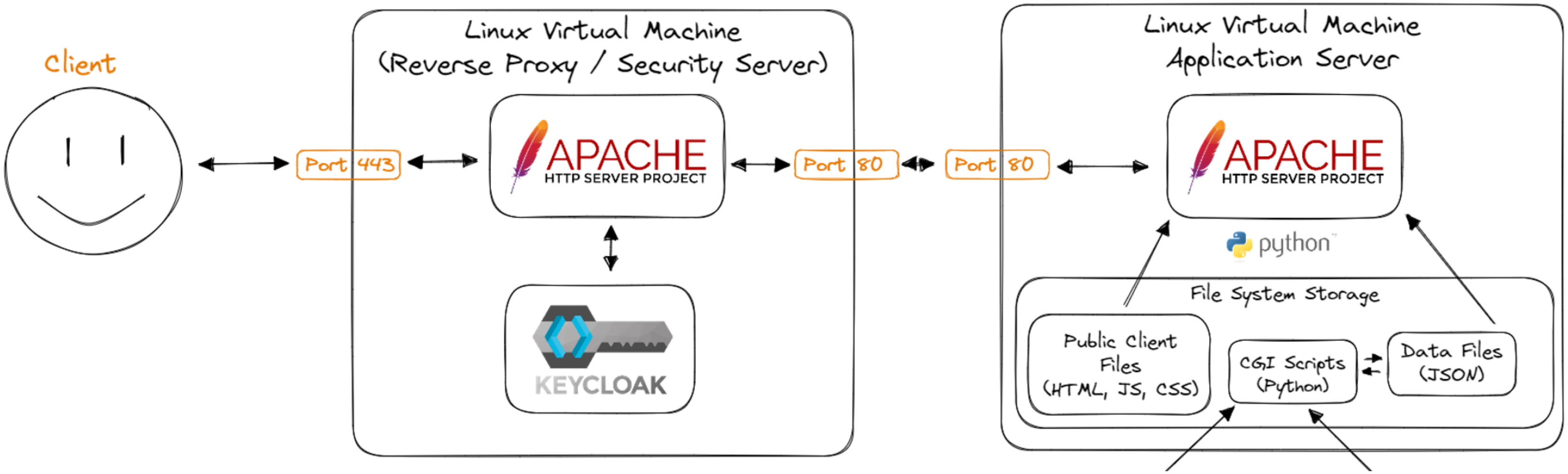
```
HMACSHA512(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

Security Server Summary

- Client uses TLS on port 443 to start an encrypted HTTPS connection to Apache reverse-proxy server
- Keycloak works with Apache server to identify the user and their access level



Component 2: Application Server



The application server:

1. Serves public client files (HTML, JS, CSS) to the client
2. Stores and sends data files in JSON format
3. Allows remote procedure calls via CGI scripts

Public Files (HTML, CSS, JS)

1. Client requests index.html from the server and receives it
2. Client’s browser parses index.html, which lists various CSS and JavaScript resources it would like to fetch
    1. `fetch` being a modern JS replacement of XMLHTTPRequest / AJAX supported by all browsers
3. Client’s browser fetches resources while parsing the rest of the page
4. JavaScript executes startup code in the browser when the browser has finished building the body of the page
    1. This includes fetching the Playbook (JSON) and History (JSON)

Application Server Data Files

- Playbook (JSON): a large list of objects with permutations of possible selections and the corresponding sign messages that will be pushed
- History (JSON): list of actions taken in the GUI, i.e. who submitted which messages and what time
- These files can be requested but not modified by the user
  - Due to their size, they are compressed with GZIP by Apache before sending to the client

CGI Scripts: Definition

- CGI = Common Gateway Interface
- Created in the early 1990s to allow web pages to be interactive
- CGI allows a server to construct an HTTP response based on dynamic data constructed or fetched on the server
- CGI also allows "remote procedure calls" – allowing the client to execute commands on another server without knowing the details of how the command is implemented

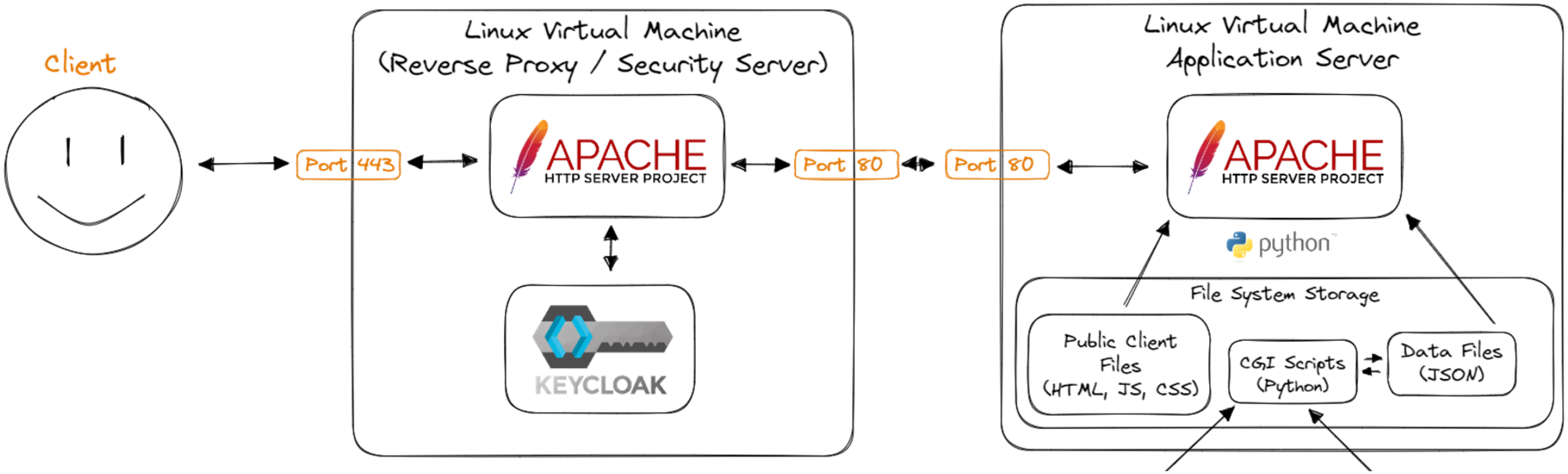
CGI Scripts: Application

- CGI serves as the gateway between Apache HTTP server and a Python runtime
- While Apache can only serve documents by default, we can extend it with the mod_cgi module to allow it to execute anything with Python.
- Python scripts establish connections with GoldenEye’s Databus and Database in order to find real-time information about CMS status and push new messages to CMSs, EMSs, and HARs.

CGI Scripts: Limitations

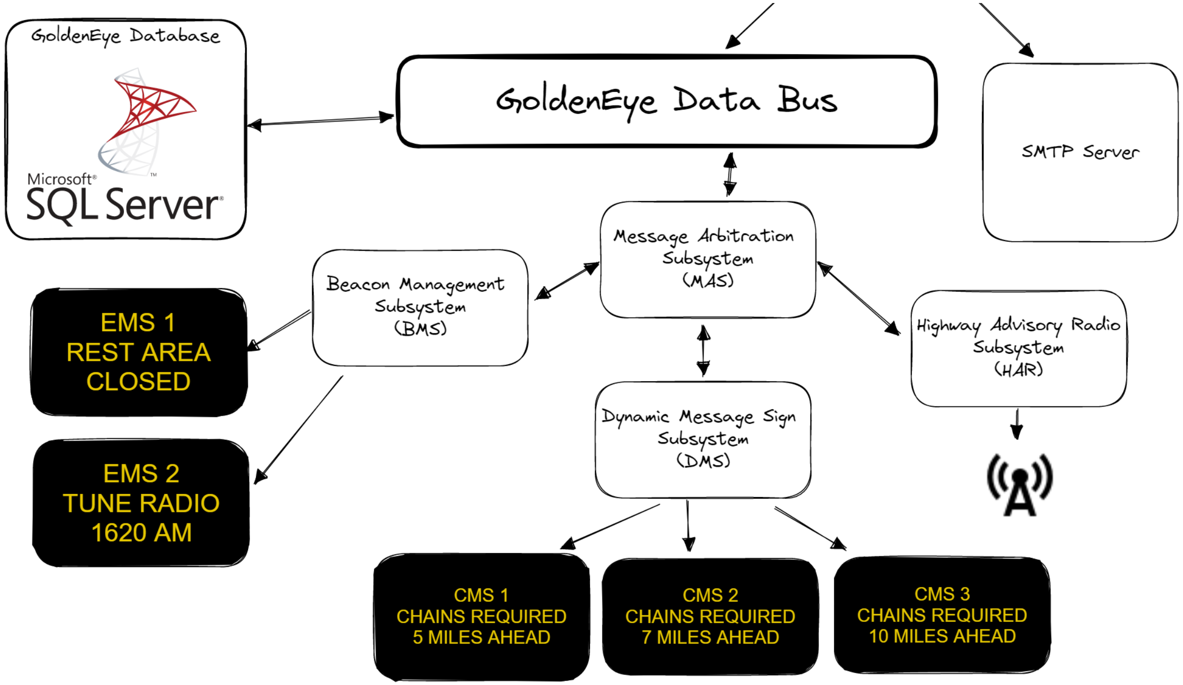
- Apache server must fork and execute a new process and load all dependencies on every call
- Overtaken in popularity in the late 90s / early 2000s by PHP and later by web frameworks (Ruby on Rails, Node.js, Next.js, etc.)
- Imperative and low-level: easy to make a mistake that can create a security vulnerability

Application Server Summary



1. Apache server gives the client the necessary files to build the interface in the web browser
2. Client’s browser then executes `fetch`
3. Allows remote procedure calls via CGI scripts

Component 3: Data Bus


GoldenEye Database
Microsoft SQL Server
GoldenEye Data Bus
SMTP Server
Message Arbitration Subsystem (MAS)
Beacon Management Subsystem (BMS)
Highway Advisory Radio Subsystem (HAR)
Dynamic Message Sign Subsystem (DMS)
EMS 1 REST AREA CLOSED
EMS 2 TUNE RADIO 1620 AM
CMS 1 CHAINS REQUIRED 5 MILES AHEAD
CMS 2 CHAINS REQUIRED 7 MILES AHEAD
CMS 3 CHAINS REQUIRED 10 MILES AHEAD

- The Data Bus is our main API gateway to accomplish all field element management
- Communicates over its own protocol on TCP via XML documents

GoldenEye Data Bus

- Databus: a system that transfers data between components inside a system
- The Message Arbitration Subsystem (MAS) must communicate with the Dynamic Message Sign Subsystem (DMS), Beacon Management Subsystem (BMS), and HAR Subsystem (HAR) via the databus
- We will also use the databus to communicate with the Message Arbitration Subsystem via our CGI scripts

Data Bus Full Diagram

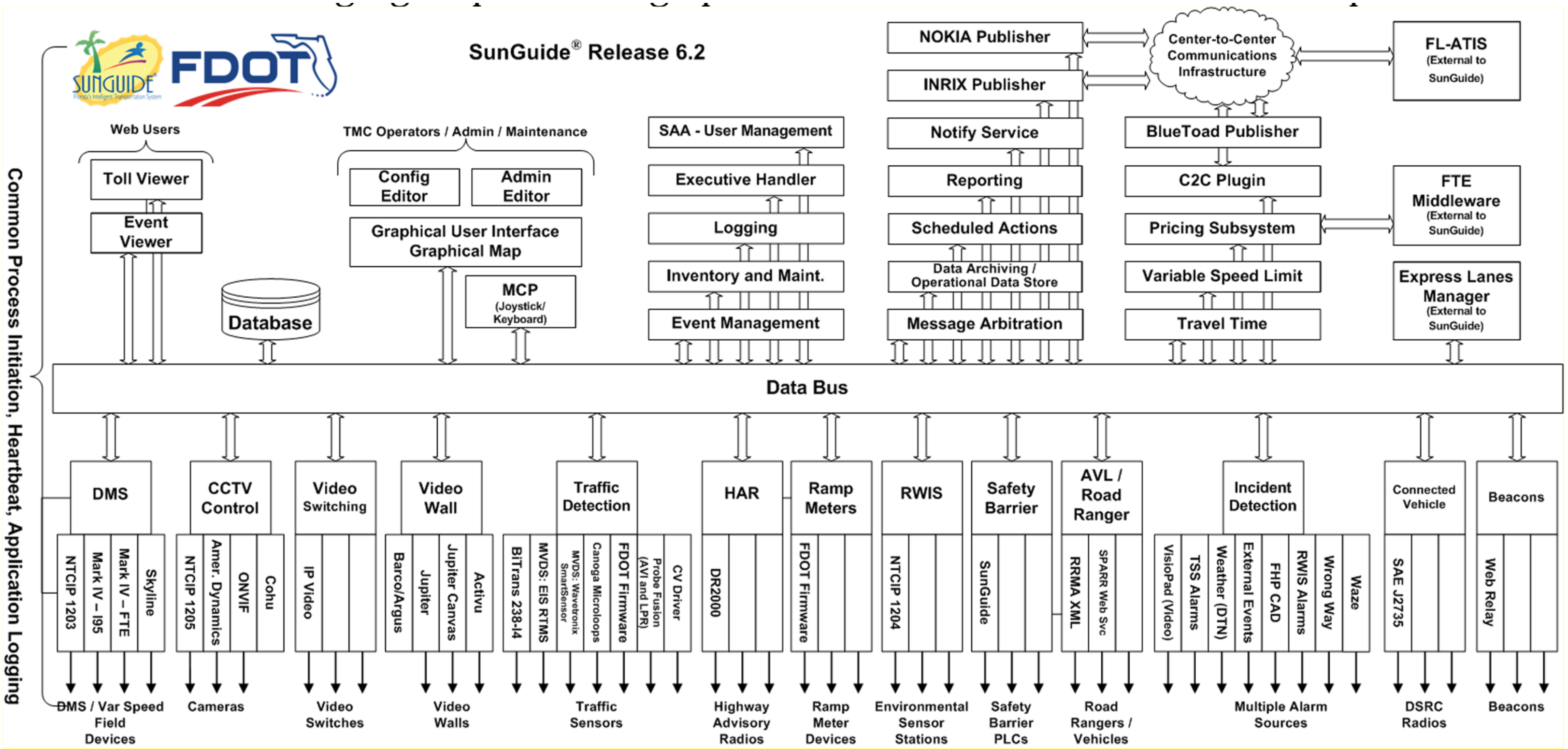
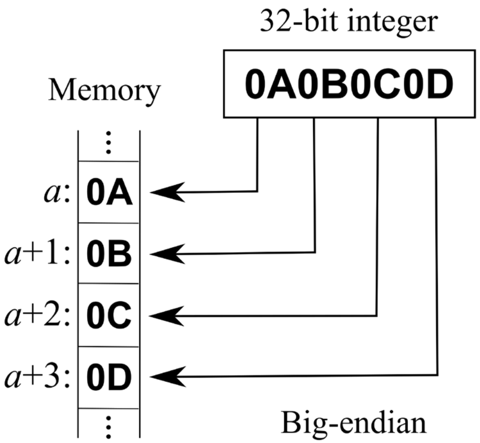


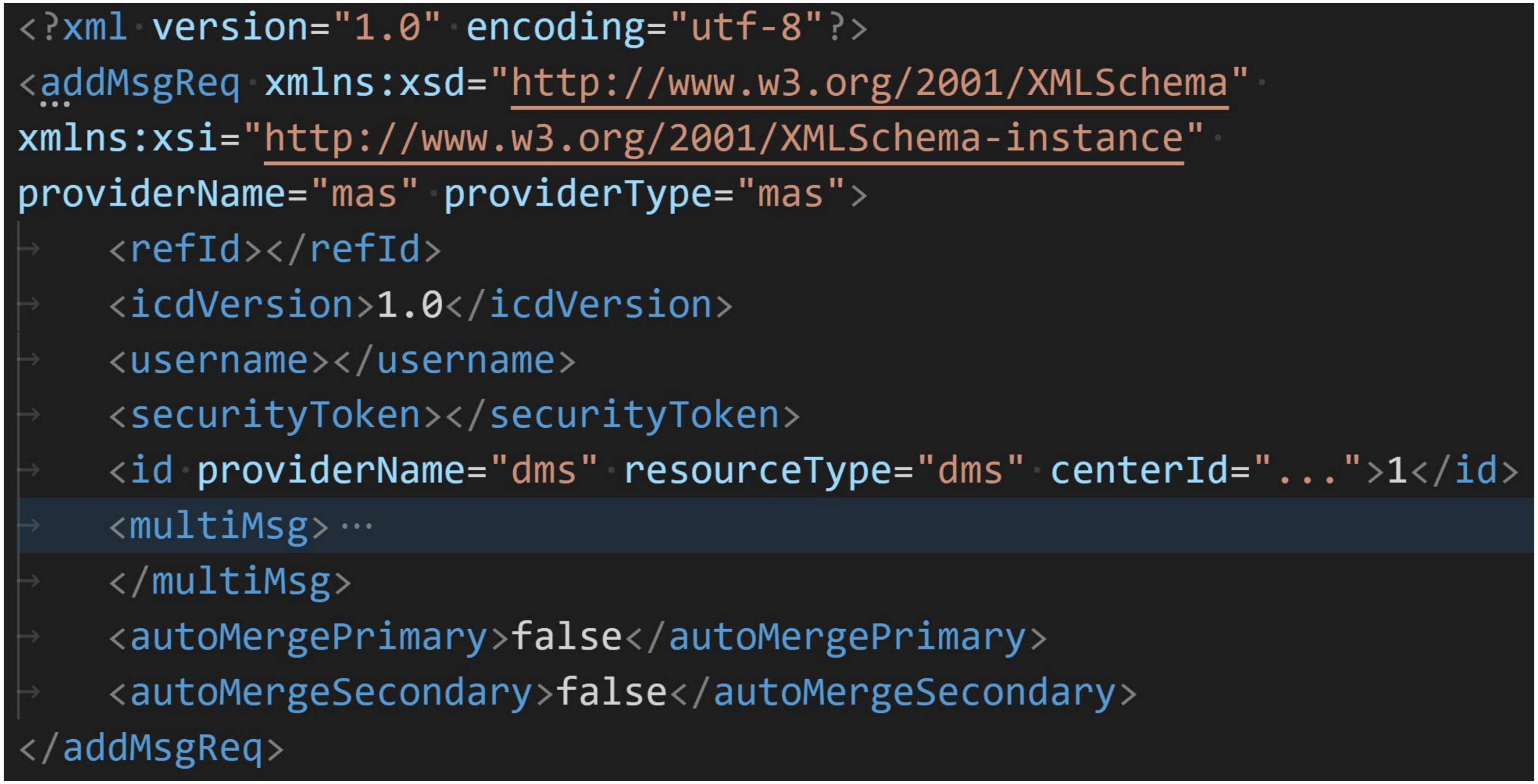
Figure 1-1 - High-Level Architectural Concept

Data Bus Protocol Specifications

- All messages must first include a 32-bit integer for Transmitted Size, then 32-bit integer for Decompressed Size, then the request in XML format
- Integers and bitmaps in big endian format



Data Bus XML

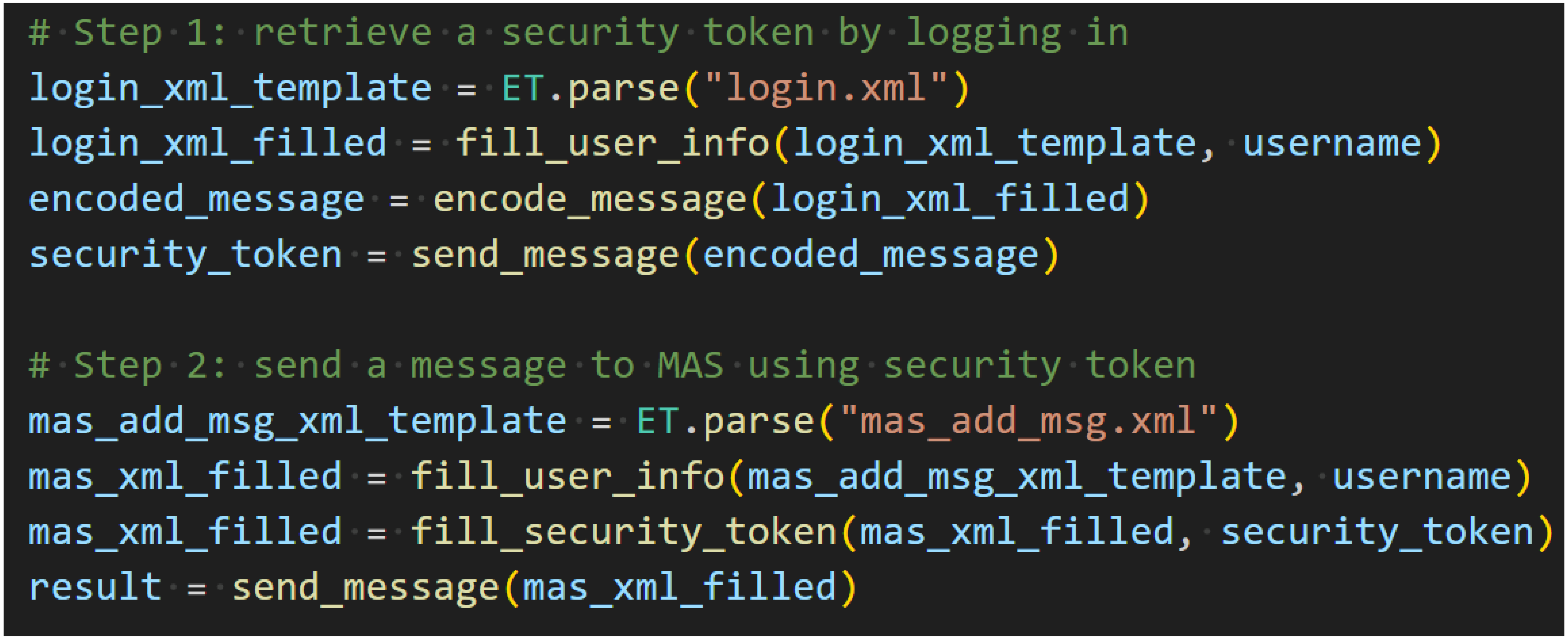


```xml
<?xml version="1.0" encoding="utf-8"?>
<addMsgReq xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
providerName="mas" providerType="mas">
    <refId></refId>
    <icdVersion>1.0</icdVersion>
    <username></username>
    <securityToken></securityToken>
    <id providerName="dms" resourceType="dms" centerId="...">1</id>
    <multiMsg>···
    </multiMsg>
    <autoMergePrimary>false</autoMergePrimary>
    <autoMergeSecondary>false</autoMergeSecondary>
</addMsgReq>
```

Data Bus XML Details

- refId: unique identifier sent by the client
- icdVersion: “Interface Control Document” version, aka. API version, which is still 1.0 as of 2023
- username: the username of the operator sending the message. We enforce that this username is the same as their Keycloak username
- securityToken: a required token retrieved by a simple XML exchange providing a username and password for a token
- Other specifications are provided in the SunGuide General Interface Control Document

Data Bus Control Pseudocode



```
# Step 1: retrieve a security token by logging in
login_xml_template = ET.parse("login.xml")
login_xml_filled = fill_user_info(login_xml_template, username)
encoded_message = encode_message(login_xml_filled)
security_token = send_message(encoded_message)

# Step 2: send a message to MAS using security token
mas_add_msg_xml_template = ET.parse("mas_add_msg.xml")
mas_xml_filled = fill_user_info(mas_add_msg_xml_template, username)
mas_xml_filled = fill_security_token(mas_xml_filled, security_token)
result = send_message(mas_xml_filled)
```

Data Bus Extensibility

- With the ability to connect with and communicate with the databus, we can control all elements, including EMS and HARs
- HARs may be sent messages using the same command via MAS
- EMS’s are similar, but we have complications to resolve first

Implementing EMS: Problem

- In order to turn our Extinguishable Message Signs (EMS) on and off, we generally use three different devices for internet remote relay control:
  - Ambery IP-P3
  - WebRelay Single
  - iBoot G2(+)
- GoldenEye only has a driver for WebRelay

EMS Remote Relay Devices

Implementing EMS: Solution

- Solution: Add an abstraction layer to translate all Ambery and iBoot messages and commands into equivalent WebRelay messages and commands
- We only need to implement “Get Status”, “Turn On”, and “Turn Off”
- This sits in between GoldenEye and EMS’s, listening for all communications between GoldenEye and all EMS Ambery, iBoot, and WebRelay devices
- We can now talk with all our EMS’s via GoldenEye’s databus!

EMS Abstraction Layer Implementation Details

- Accomplished using Python asyncio to listen on 50+ ports at the same time and accomplish asynchronous, non-blocking communication
- All device “clients” derive from an abstract base class, ensuring that from they all implement “Get Status”, “Turn On”, and “Turn Off”
- Copied WebRelay’s `status.xml` response file to pretend to be a WebRelay for all devices

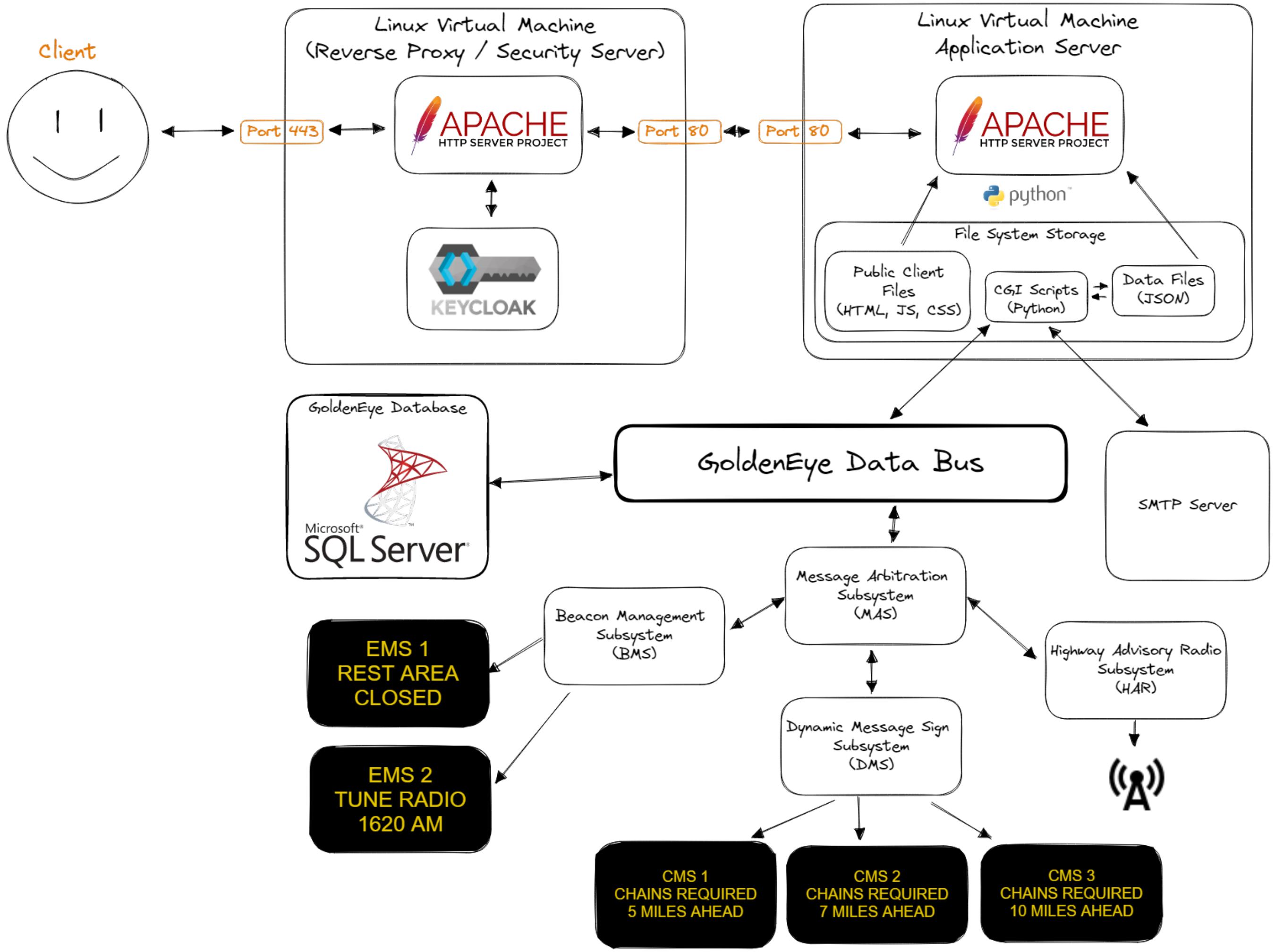
Extra: HAR Communication

- We use **MH Corbin Platinum** to control our HARs
- **SIM** (Software Interface Module) allows GoldenEye to communicate with Platinum via a shared file directory
  - A command to HAR 37 is sent by placing a COMMAND37.txt file in the shared directory to be read and processed by Platinum
- After configuration, GoldenEye handles communication with Platinum, and our CGI scripts communicate with GoldenEye via MAS subsystem

Extra: SMTP

- When we send chain control messages, we also send an **executive page** (epage) to the relevant notifying parties
- CGI scripts read what command is being sent, and generates a message such as:
  - **@1327 R2 SLYPARK TO FRESH POND // R2 FREDS TO MEYERS - TKS MIN**
- CGI scripts then send this message to the notification list to Caltrans SMTP server over port 25 to send out via email.
- For text messages, carriers provide gateways such as: @vtext.com, @txt.att.net, @tmomail.net, etc. to forward emails as texts

Putting it All Together



Putting it All Together: Part 1

1. Client requests /ChainControl/index.html from our security server over HTTPS
2. Security server first redirects the client to login to Keycloak
3. Client logs in, and the security server verifies that the client user has permission to access the page. Client receives a token as proof of identity.
4. Sec. server requests /ChainControl/index.html from the App. Server.
5. Sec. Server gets index.html and relays it to the client.

Putting It All Together: Part 2

6. Client’s browser parses index.html and starts requesting other public files like CSS and JS files. Those get sent by App. and Sec. servers.
7. Client’s browser starts executing JavaScript to fetch Playbook and History data. Those get sent as Gzipped JSON, which the client decompresses and parses.
8. Client’s web application is now fully ready for commands.

Putting It All Together: Part 3

9. Client selects a Chain Control action and chooses messages to send to 10 signs.
10. Client JS executes a fetch to a CGI script with a command to set these sign messages. The body of this fetch HTTP request contains username, timestamp, action, and sign messages.
11. CGI script logs into the Databus using the user's credentials and translates the command into the Databus's API format.
12. Databus receives the command and executes it.

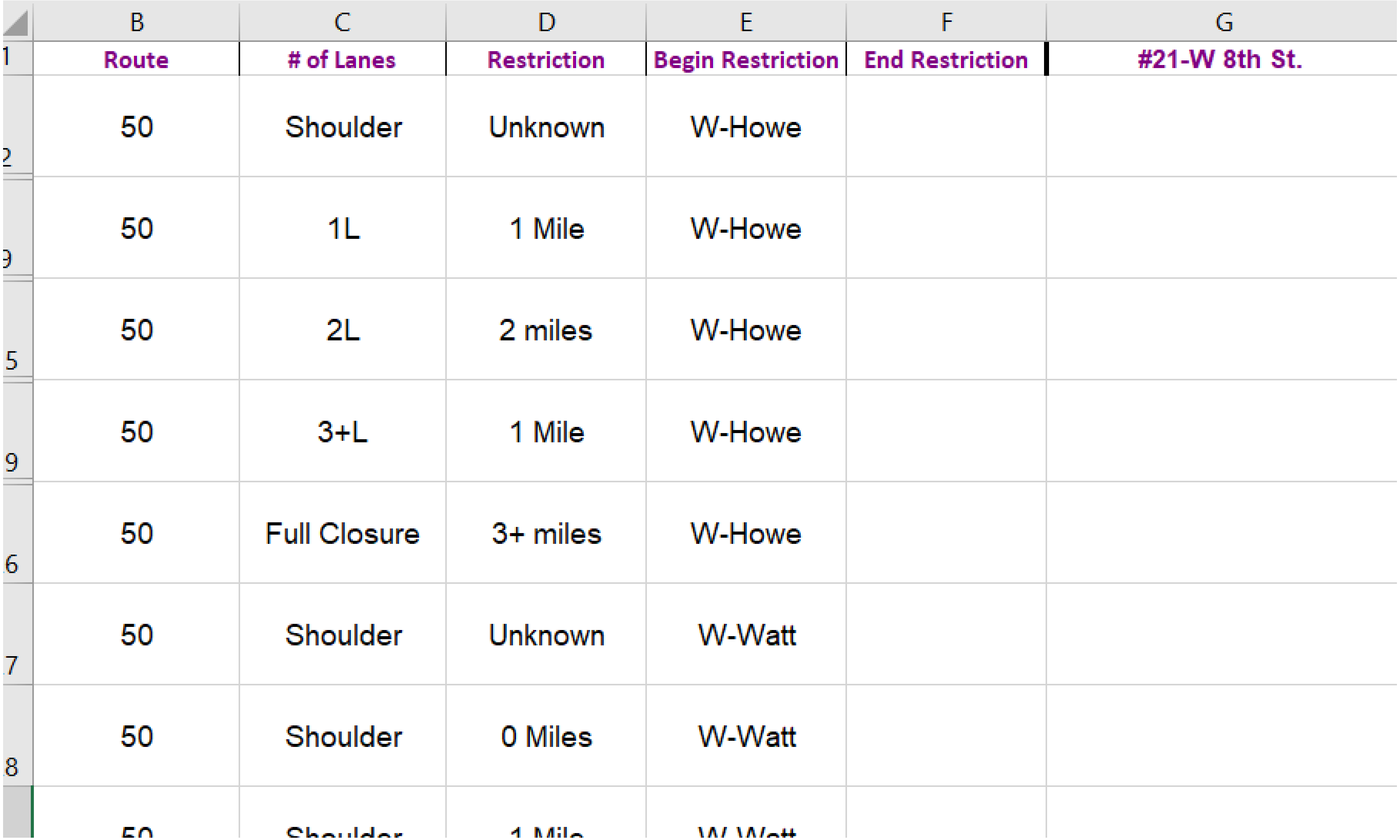
Putting It All Together: Part 4

13. Databus responds to CGI script saying that the command has executed successfully.
14. CGI script sends an email to the Caltrans SMTP server with an epage message to send to notifying parties.
15. CGI script responds to the Client JS with a 200 OK status.
16. Client user sees in their browser that the command was successful.

Next Steps - ICM

- Similar input parameters
    - Route.
    - Num of lanes
      Shoulder, 1L, 2L, 3+L, Full Closure.
    - Restriction
      Unknown, 0 miles, 1 mile, 2 miles, 3+ miles.
    - Begin Restriction
    - End Restriction

ICM Playbook



	B	C	D	E	F	G
1	Route	# of Lanes	Restriction	Begin Restriction	End Restriction	#21-W 8th St.
2	50	Shoulder	Unknown	W-Howe		
9	50	1L	1 Mile	W-Howe		
5	50	2L	2 miles	W-Howe		
9	50	3+L	1 Mile	W-Howe		
6	50	Full Closure	3+ miles	W-Howe		
7	50	Shoulder	Unknown	W-Watt		
8	50	Shoulder	0 Miles	W-Watt		
	50	Shoulder	1 Mile	W-Watt		

Benefits

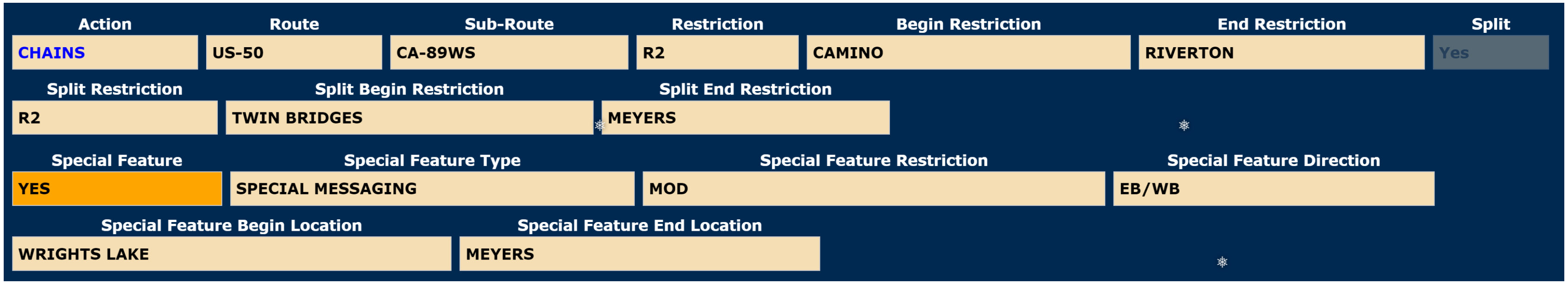
- Increased accuracy.
- Consistent messaging.
- Quick message deployment.
- Quick message update.
- Quick message removal.

Issues Encountered

- Synchronization issues (WYSI-N-WYG).
  - Queued messages get pushed after incident has cleared.
- Operators no longer operate.
  - Smartphone analogy.
- HAR integration.
- ActiveITS only supports WebRelay.
- Operators confused about interface, needed additional training.
- Requests for corner cases had to be dropped.

Future Improvements?

- Frontend became too cluttered and hard for users to understand



Action: CHAINS

Route: US-50

Sub-Route: CA-89WS

Restriction: R2

Begin Restriction: CAMINO

End Restriction: RIVERTON

Split: Yes

Split Restriction: R2

Split Begin Restriction: TWIN BRIDGES

Split End Restriction: MEYERS

Special Feature: YES

Special Feature Type: SPECIAL MESSAGING

Special Feature Restriction: MOD

Special Feature Direction: EB/WB

Special Feature Begin Location: WRIGHTS LAKE

Special Feature End Location: MEYERS

- Backend code was hard to maintain and hard to debug for those not used to CGI Scripts and parsing Apache or system logs

Proposed Solutions and Ideas:

- Next.js server to replace application server
  - Everything is JS, which is easy for new hires or student assistants to understand and write
  - Easier to set up development and testing environments compared to installing a local Apache server
  - React.js allows much easier control over complicated front-end interfaces
- Connections to TMC Activity Logging server via MySQL connection?