/\*\*

\* Script for parsing the input message and building clean JSON

\*/

function executeScript(event) {

const input = event.getParameter('message\_to\_claims\_ops');

if (typeof input !== 'string') {

throw new Error('Input must be a string.');

}

const jsonOutput = {

message: {}

};

// Set default values

jsonOutput.message.Other\_Carrier\_Insurance = null;

jsonOutput.message.Member\_is\_calpers = null;

jsonOutput.message.claim\_details = {};

jsonOutput.message.Action = "Routed to claims ops";

// Define regex patterns

const carrierPattern = /(do NOT have|does NOT have)/i;

const calpersPattern = /member\s+is\s+calpers/i;

const claimTypePattern = /Claim\s\*Type:\s\*(\w+)/i;

const claimIdPattern = /Claim\s\*id[:\s]\*([\w\d]+)/i;

const serviceFromPattern = /Service\s\*From\s\*Date[:\s]\*([\d.E]+)/i;

const serviceThruPattern = /Service\s\*Thru\s\*Date[:\s]\*([\d.E]+)/i;

const allowedAmountPattern = /total\s\*allowed\s\*amount[:\s]\*([\d.]+)/i;

const chargeAmountPattern = /total\s\*charge\s\*amount[:\s]\*([\d.]+)/i;

const memberNamePattern = /member\s\*name\s\*is\s\*([a-zA-Z\s]+)/i;

const memberIdPattern = /member\s\*id\s\*[:\s]\*([\d]+)/i;

const dobPattern = /date\s\*of\s\*birth[:\s]\*([\d\/]+)/i;

// Match patterns

if (carrierPattern.test(input)) {

jsonOutput.message.Other\_Carrier\_Insurance = "No";

} else {

jsonOutput.message.Other\_Carrier\_Insurance = "Yes";

}

if (calpersPattern.test(input)) {

jsonOutput.message.Member\_is\_calpers = "Yes";

} else {

jsonOutput.message.Member\_is\_calpers = "No";

}

const claimTypeMatch = input.match(claimTypePattern);

if (claimTypeMatch) jsonOutput.message.claim\_details["Claim Type"] = claimTypeMatch[1];

const claimIdMatch = input.match(claimIdPattern);

if (claimIdMatch) jsonOutput.message.claim\_details["Claim id"] = claimIdMatch[1];

const serviceFromMatch = input.match(serviceFromPattern);

if (serviceFromMatch) jsonOutput.message.claim\_details["Service From Date"] = serviceFromMatch[1];

const serviceThruMatch = input.match(serviceThruPattern);

if (serviceThruMatch) jsonOutput.message.claim\_details["Service Thru Date"] = serviceThruMatch[1];

const allowedAmountMatch = input.match(allowedAmountPattern);

if (allowedAmountMatch) jsonOutput.message.claim\_details["total allowed amount"] = parseFloat(allowedAmountMatch[1]);

const chargeAmountMatch = input.match(chargeAmountPattern);

if (chargeAmountMatch) jsonOutput.message.claim\_details["total charge amount"] = parseFloat(chargeAmountMatch[1]);

const memberNameMatch = input.match(memberNamePattern);

if (memberNameMatch) jsonOutput.message.claim\_details["Member Name"] = memberNameMatch[1].trim();

const memberIdMatch = input.match(memberIdPattern);

if (memberIdMatch) jsonOutput.message.claim\_details["Member ID"] = memberIdMatch[1];

const dobMatch = input.match(dobPattern);

if (dobMatch) {

const dob = dobMatch[1];

const dobRegex = /^\d{1,2}\/\d{1,2}\/\d{2}$/;

if (dobRegex.test(dob)) {

jsonOutput.message.claim\_details["Member DOB"] = dob;

} else {

throw new Error(`Invalid DOB format: Expected MM/DD/YY, got: ${dob}`);

}

}

// Remove any keys that are null

function clean(obj) {

for (const key in obj) {

if (obj[key] && typeof obj[key] === 'object') clean(obj[key]);

if (obj[key] === null || (typeof obj[key] === 'object' && Object.keys(obj[key]).length === 0)) {

delete obj[key];

}

}

}

clean(jsonOutput);

event.setParameter('output', JSON.stringify(jsonOutput));

}