

Cognitive Self-Service Virtual Banker

Architecture

And

Consolidated Test Results Document

(Sprint-1 to Sprint-5)

Final

Author(s):

Jim Barneebee

Nitin Kulkarni

Customer: Regions

29th November 2017

Document Location

<Where is document stored>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Revision Date** | **Summary of Changes** | **Changes marked** |
|  |  |  |  |

Approvals

This document requires following approvals.

|  |  |
| --- | --- |
| **Name** | **Role** |
|  |  |

Signed approval forms are filed in Evidence database in the same location of this document.

Distribution

This document has been distributed to the people on the approval list and to:

Contents

Table of Contents

[Table of Contents 3](#_Toc499750936)

[1 Introduction 5](#_Toc499750937)

[1.1 Architecture overview: 6](#_Toc499750938)

[1.2 Technical Components: 8](#_Toc499750939)

[1.3 Node-Red flows 9](#_Toc499750940)

[2 Create Separate Package for Encryption Related flows 13](#_Toc499750941)

[2.1 Overview 13](#_Toc499750942)

[2.2 Encryption utility 13](#_Toc499750943)

[2.3 How the flow works? 13](#_Toc499750944)

[2.4 SOE-Flow: Batch Encryption (Docker) 14](#_Toc499750945)

[2.4.1 Subflow – (5-Batch Encryption (Docker)) 14](#_Toc499750946)

[2.4.2 Subflow – (5.6 - Batch Process flow for Encryption) – Encrypt string using Crypto-JS/AES algorithm. 15](#_Toc499750947)

[2.5 Test Result- “Successful” 18](#_Toc499750948)

[3 CVP API Service to provide user data 20](#_Toc499750949)

[3.1 Overview 20](#_Toc499750950)

[3.2 CVP API flow – Invoked by UCG/SOE Main flow 20](#_Toc499750951)

[3.3 How flow works? 21](#_Toc499750952)

[3.4 Test Result- “Successful” 21](#_Toc499750953)

[4 CVP API flow – Stores user data in CVP 22](#_Toc499750954)

[4.1 Overview 22](#_Toc499750955)

[4.2 CVP API flow – Invoked by IVR 22](#_Toc499750956)

[4.3 How flow works? 22](#_Toc499750957)

[5 UCG/SOE Main flow 23](#_Toc499750958)

[5.1 Overview 23](#_Toc499750959)

[5.2 SQL Server Table Setup for session management 23](#_Toc499750960)

[5.3 Flow to Load SOE environment file, Call Transfer Routes and WCS node mapping into memory 25](#_Toc499750961)

[5.3.1 Subflow – Docker Read Environment File (Docker) 25](#_Toc499750962)

[5.3.2 Subflow – LoadCallTransferNumbers 27](#_Toc499750963)

[5.3.3 Subflow – Load Containment Mapping 27](#_Toc499750964)

[5.3.4 SOE Environment Content, Global Variables 28](#_Toc499750965)

[5.3.5 Context Variables / vgw Action Sequence parameters 29](#_Toc499750966)

[5.4 SOE/UCG Main flow 32](#_Toc499750967)

[5.4.1 Subflow : (15-GetMssqlSession) 37](#_Toc499750968)

[5.4.2 Subflow: (30-CVP API) 37](#_Toc499750969)

[5.4.3 Subflow : (40-Get Rest 1st Fee) 38](#_Toc499750970)

[5.4.4 Subflow : (90-Post REST 1st Fee) 38](#_Toc499750971)

[5.4.5 Subflow : (95-Post Refund Decline) 39](#_Toc499750972)

[5.4.6 Subflow : (135-Post Refund Decline) 40](#_Toc499750973)

[6 Enabling Splunk Logging 42](#_Toc499750974)

[6.1 Overview 42](#_Toc499750975)

[6.2 How to enable Splunk Logging? 42](#_Toc499750976)

[6.3 Test Result- “Successful” 42](#_Toc499750977)

[7 Error handling and Time-out Handling Strategy for SOE 43](#_Toc499750978)

[7.1 Time-outs within SOE: 44](#_Toc499750979)

[7.2 Time-outs within vgw: 44](#_Toc499750980)

[7.3 SOE flow for catch node 45](#_Toc499750981)

[8 Test Result – vgwLog and vgwSession output captured in SQL server database 46](#_Toc499750982)

[9 Instructions to create Sprint5 docker images 47](#_Toc499750983)

[9.1 Build Docker Images for each Flow 48](#_Toc499750984)

[9.1.1 Folders on Host server (NFS Share) 48](#_Toc499750985)

[9.1.2 Build Container with Volume Mapping 49](#_Toc499750986)

[10 Deployment Tasks and Dependencies 51](#_Toc499750987)

# Introduction

The Regions Cognitive Self-Service Virtual Banker (“CSSVB”) leverages Watson technology to transform automated service to customers. It offers a cognitive, conversational self-service experience that can provide answers and take action. This phase of the project focuses on a) Online Banking FAQ and b) First Fee Refund. The technical activities include the following:

* Designing the end-to-end solution for integration
* Setting up base infrastructure
* Developing Rest APIs for integration
* Integrating various components to provide seamless business solution and customer experience

Regions and IBM have collaborated during the lifecycle of this project. The team has used agile methodology, spanning over 5 sprints.

This document combines and discusses activities completed during all 5 sprints to deliver the final solution. Each sprint provided the base foundation for the subsequent sprint. Sprint-4 & Sprint-5 includes the following key activities:

The following were enhancements added as part of Sprint-4 (Oct 9th 2017 – Nov 3rd 2017):

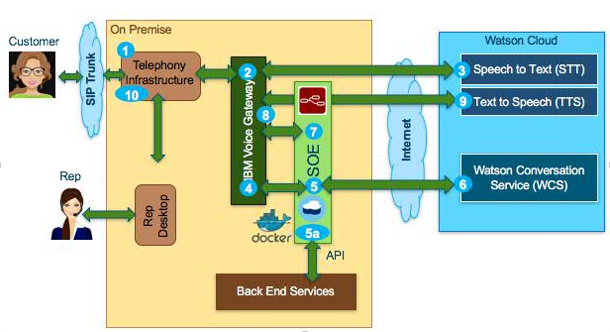
* Integrating CVP to fetch customer data needed for 1st Fee Refund
* Integrating 1st Fee Refund
* Integrating 1st Fee Refund customer decline posting for reporting
* Integrating call transfer to vgw
* Adding call containment variable to support generating VA conversation metrics
* Docker load balancing
* Error Handling and Time-out Handling Strategy for SOE

The following were enhancements added as part of Sprint-5 (Nov 6th 2017 – Nov 17th 2017):

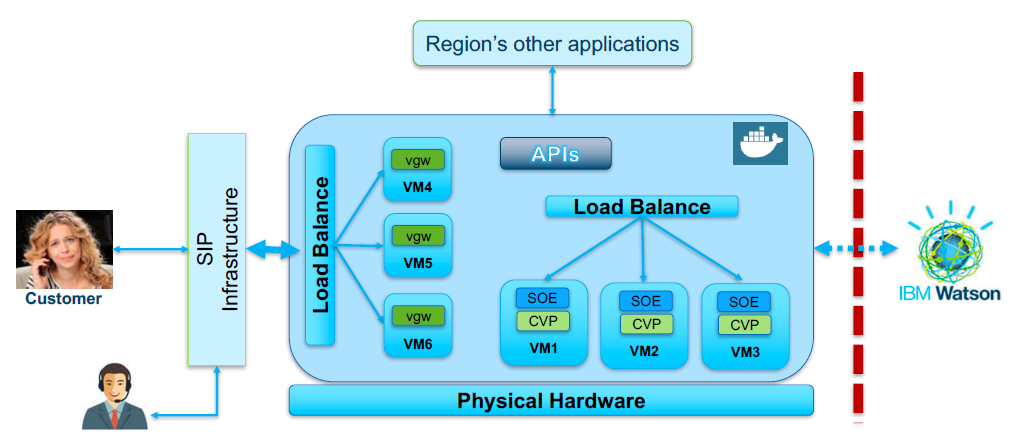
* Enabling Bi-directional HTTPS/TLS for SOE
* Adding Basic Authentication for HTTPS calls
* Enhancing feature of disabling UCG/SOE UI in higher-up environments

## Architecture overview:

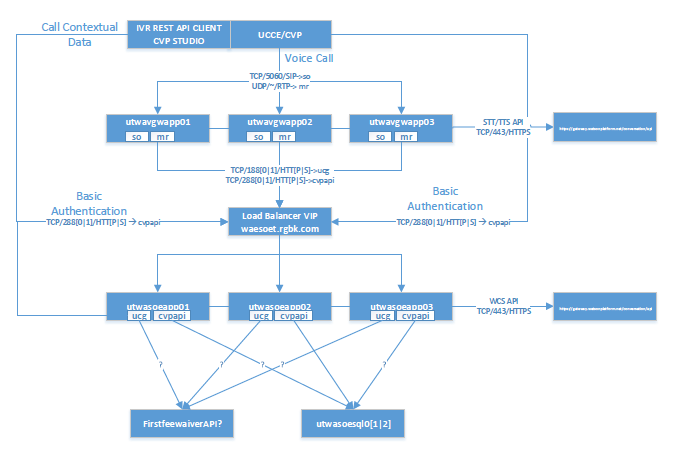
The following diagram provides the overall architecture view of the solution:



*Cognitive Self-Service Virtual Banker (“CSSVB”)– overall Architecture component view*



*Cognitive Self-Service Virtual Banker (“CSSVB”)– overall Architecture Docker View*



*Cognitive Self-Service Virtual Banker (“CSSVB”)– overall Integration View*

## Technical Components:

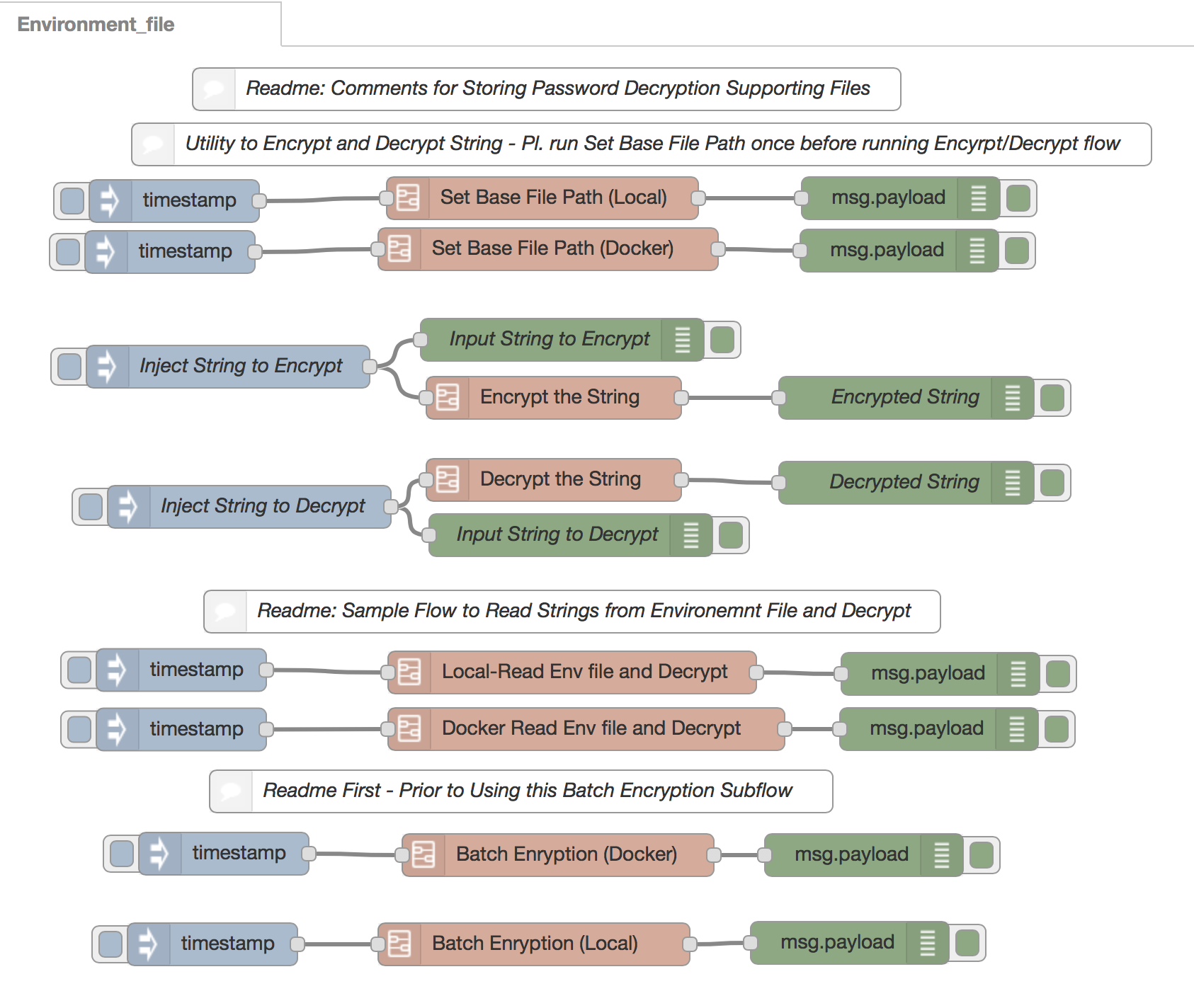
The Cognitive Self-Service Virtual Banker (“CSSVB”) includes the following components:

|  |  |
| --- | --- |
| Technical Component | Comments |
| SIP | SIP Infrastructure |
| IBM Voice Gateway | Gateway connects to SIP and UCG/SOE leveraging Watson APIs- Speech to Text (STT) and Text to Speech (TTS) |
| SOE/UCG | Node-Red flow version 0.17.5 / UCG 3.0-  This is the core orchestration layer of the solution which provides business solutions by integrating Regions’ business applications, telephonic components, Watson services, and session management.  It also captures conversation logs and call containment data for subsequent analysis which Regions intends to generate in a report/scorecard. |
| Backend applications/components | This includes:   * Mainframe application for first fee * CVP hosts call user data received from IVR for the current call * Fee Decline Reporting * SQL Server database to store session management data, maintain conversation log and call containment data. * Splunk to capture Docker/container level exceptions, SOE, CVP flow exceptions |
| Watson Services | Watson Conversation Service running on Premium Instance |
| Docker / Container | Docker / Container running SOE/UCG, CVP and VGW applications on virtual machines with Linux. |
| Web Servers | * Web servers running SOE UI, CVP UI Docker/container with HTTPS and basic authentication enabled * Web server running Environment file encryption utility (Docker/container) with https enabled |
| Load Balancer | Load Balancer deployed for VGW, CVP and UCG/SOE |
| SOE Environment File | Holds environment settings which provides run time parameters to Docker/container based applications. These can be managed externally without impacting base application images. |
| settings.js | Setting file for Node-red flow, which resides within running container |
| Dockerfile | Utility file to create Docker image |
| Certificates | HTTPs Certificates |

## Node-Red flows

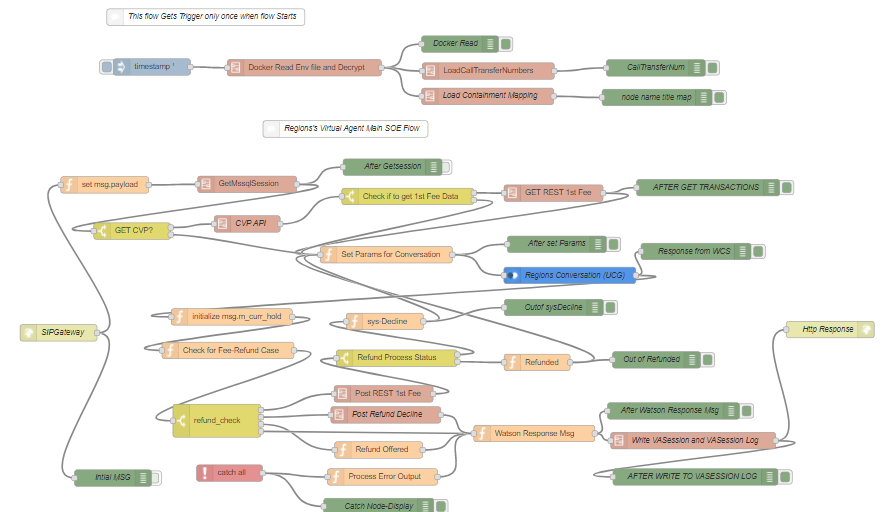
**UCG/Environment File Encryption Flow**

The Environment file encryption utility based on Cryto-JS algorithm. The Regions CSSVBsupport team will manage this flow with the help of Regions security team.

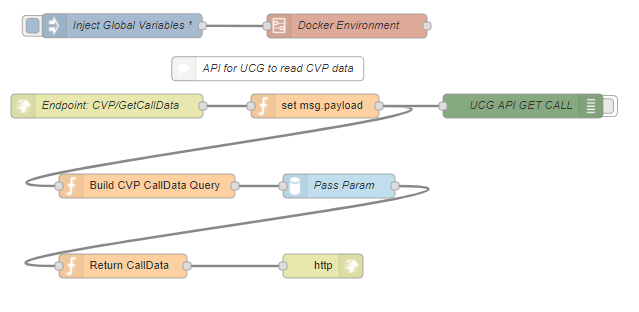
****

**Overall USG/SOE Flow**

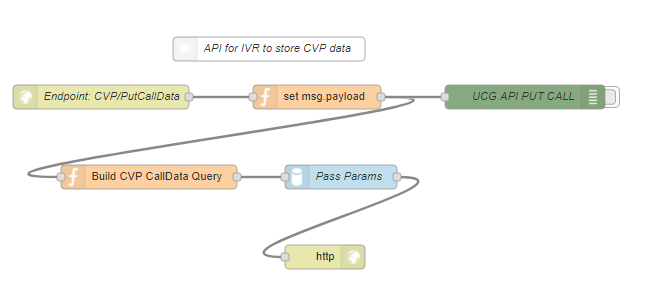
The core UCG/SOE flow which provides orchestration for the CSSVB.



**CVP API service built within Node-Red – UCG/SOE to read CVP Data**



**CVP API to store IVR Data (used by IVR)**



# Create Separate Package for Encryption Related flows

## Overview

Crypto.JS/AES is Region’s corporate standard for encryption. The separate encryption package includes the following flows:

1. The flows to encrypt and/or decrypt an individual string, which is passed using an inject node
2. The flow to read SOE environment file and decrypt
3. The batch encryption utility

## Encryption utility

Encryption utility reads input values from “***SOE\_environment\_file\_input***”, uses secrete password phrase from the file “***password\_phrase.txt***” and generates “***SOE\_environment\_file\_output***”, which contains encrypted values based on the flag provided in “***SOE\_environment\_file\_input***” and generates “***SOE\_environment\_revalidation\_output***” file to validate that the encrypted string is decrypted correctly.

**Structure of Input file:**

|  |
| --- |
| Subject Flag Value  ====================== ==== ====================================  conversation\_workspace |R| 321e996d-efee-42a0-8aed-3dc93006a026  mf\_ivrh\_url |N| <https://devl:16180/iv/IVRH>  Flag 'R' indicates - Value needs to be encrypted  'N' indicate4s - encryption is not needed and can be pass as is  ‘|’ – Pipe is used as delimiter (this cannot be use as part of subject or value. |

|  |  |
| --- | --- |
| File locations | “Supporting File Folder” on the NFS Share.  Recommended:  Separate Share for Dev, Test & QA and Production & DR. |
| Input files | SOE\_environment\_file\_input & password\_phrase.txt |
| Output files | SOE\_environment\_file\_output & SOE\_environment\_revalidation\_output |
| UI URL | “***http: udwasoeapp02:3880”*** |

## How the flow works?

The flow reads Input environment file, and password phrase from the password phrase file.

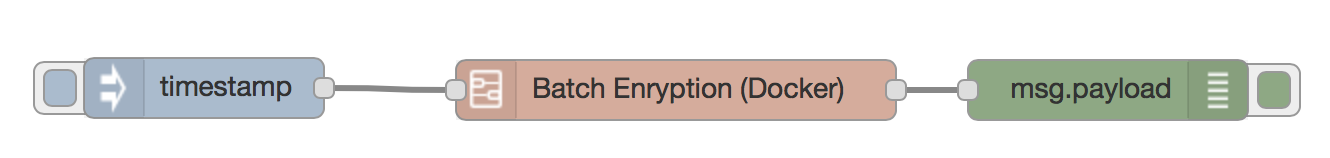
The flow uses Crypto-JS/AES algorithm and encrypts the value based on the flag.

The flow reads one line at a time from Input file, perform encryptions based on the flag and writes output to the output file. In the output file:

* on encrypted rows, the flag will be set to "Y"
* on pass-thru lines the flag will remain asis i.e. "N"

**Revalidation File:** The purpose of the revalidation file is to confirm encrypted string is decrypted correctly and matches original value. The flow passes the encrypted string to the decrypt function and generates the original string.

## SOE-Flow: Batch Encryption (Docker)



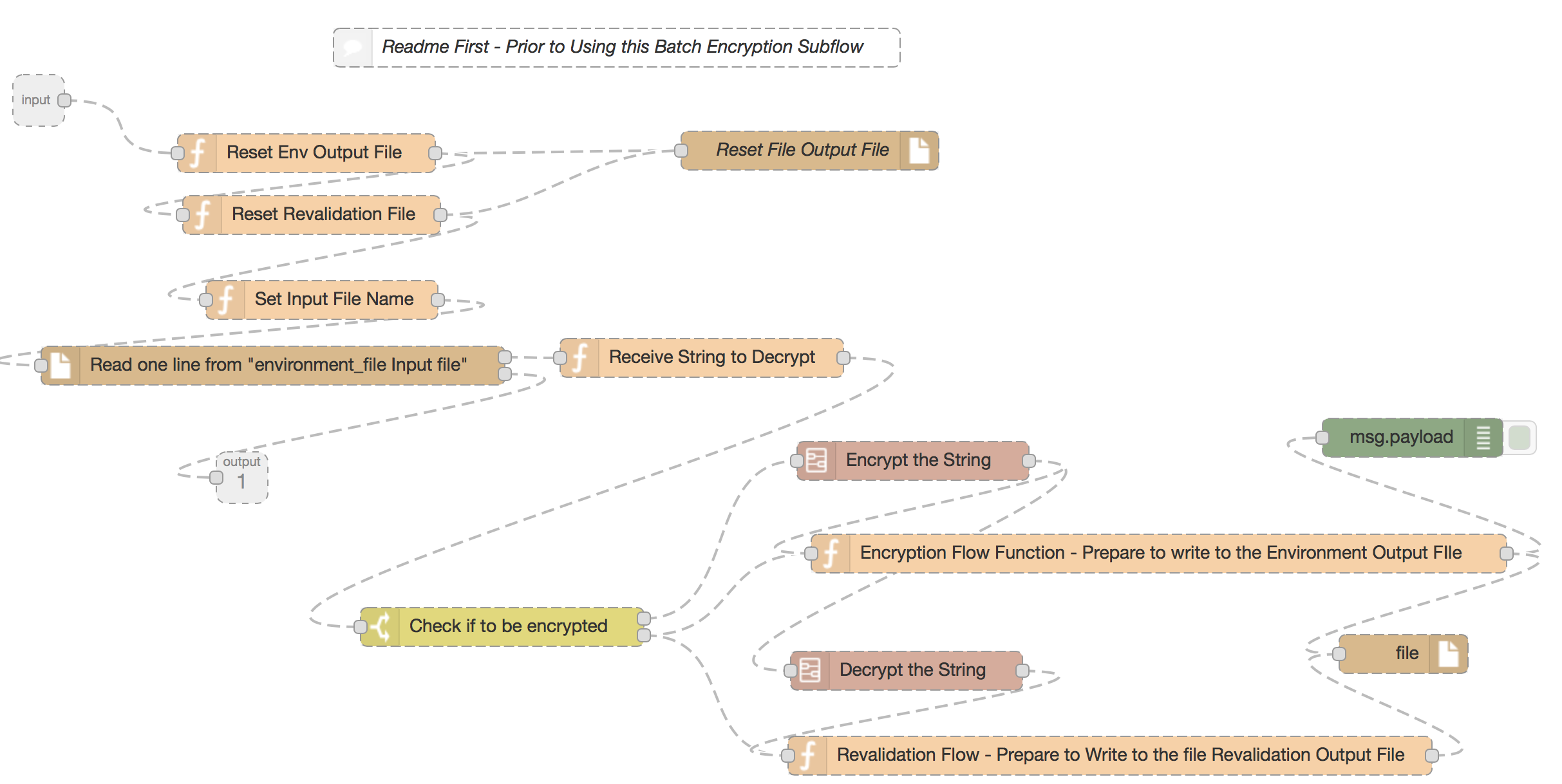
|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 1 |  | Inject timestamp to initiate the flow. |
| 5 |  | Subflow for the Batch Encryption |
| 10 |  | Message log to display return message |

### Subflow – (5-Batch Encryption (Docker))



|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 5.1 |  | Input and Output nodes for the subflow |
| 5.3 |  | Sets path and file names for SOE environment files (input, output and revalidation), password phase. |
| 5.6 |  | Subflow for encryption |

### Subflow – (5.6 - Batch Process flow for Encryption) – Encrypt string using Crypto-JS/AES algorithm.



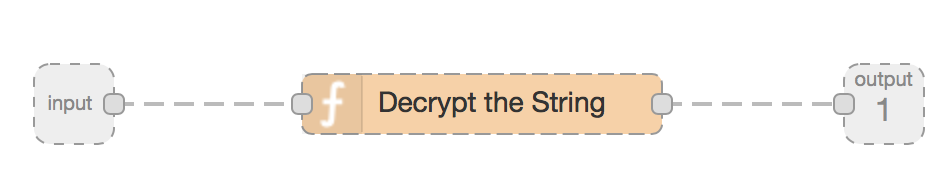
|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 5.6.1 |  | Input and Output nodes for the subflow |
| 5.6.3 |  | Set msg.filename to SOE environment output file to Initialize |
| 5.6.6 |  | Set msg.filename to SOE environment revalidation file to Initialize |
| 5.6.9 |  | Node Initialize SOE output and revalidation files |
| 5.6.12 |  | Set msg.filename to SOE environment input file to start reading |
| 5.6.15 |  | Reads one line at time from the input files and passes the contents to ‘Receive string to Decrypt’ node  Once it reaches end of file – it passes control to Output node. |
| 5.6.18 |  | Receives one line at a time from SOE environment file; function node split the contents based on delimiter and assign to separate variables. |
| 5.6.21 |  | Split node checks the encryption flag; if ‘R’ then it sends the strings to encryption node; otherwise it sends to strings to environment file output node. |
| 5.6.23 |  | Subflow Encrypt the string – receives string to encrypt |
| 5.6.26 |  | Prepare to write encrypted string to SOE environment output file |
| 5.6.29 |  | Decrypt String Subflow – Receives encrypted string to decrypt for validation |
| 5.6.32 |  | Prepare to write decrypted string to SOE environment revalidation output file |
| 5.6.35 |  | Node writes to output and revalidation files |
| 5.6.38 |  | All green color nodes are debug node to display object ‘msg’ |

#### Subflow – (5.6.23 Encrypt the String) – function to encrypt the string.



|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 5.6.23.1 |  | Input and Output nodes for the subflow |
| 5.6.23.3 |  | Encrypts the string using Crypto.JS/AES algorithm |

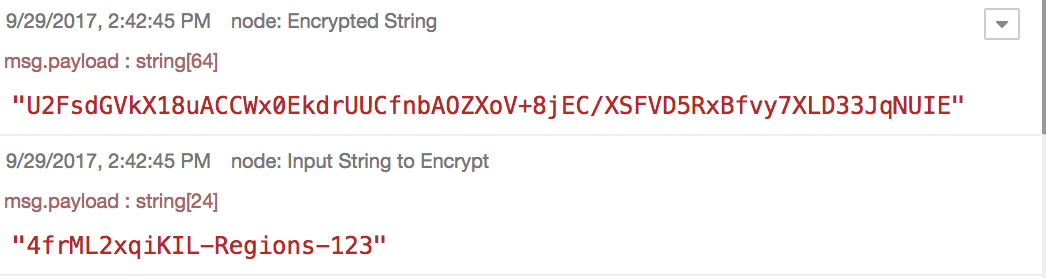
#### Subflow – (5.26.29 Decrypt the String) – function to decrypt the string



|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 5.26.29.1 |  | Input and Output nodes for the subflow |
| 5.26.29.3 |  | Decrypts the string using Crypto.JS/AES algorithm |

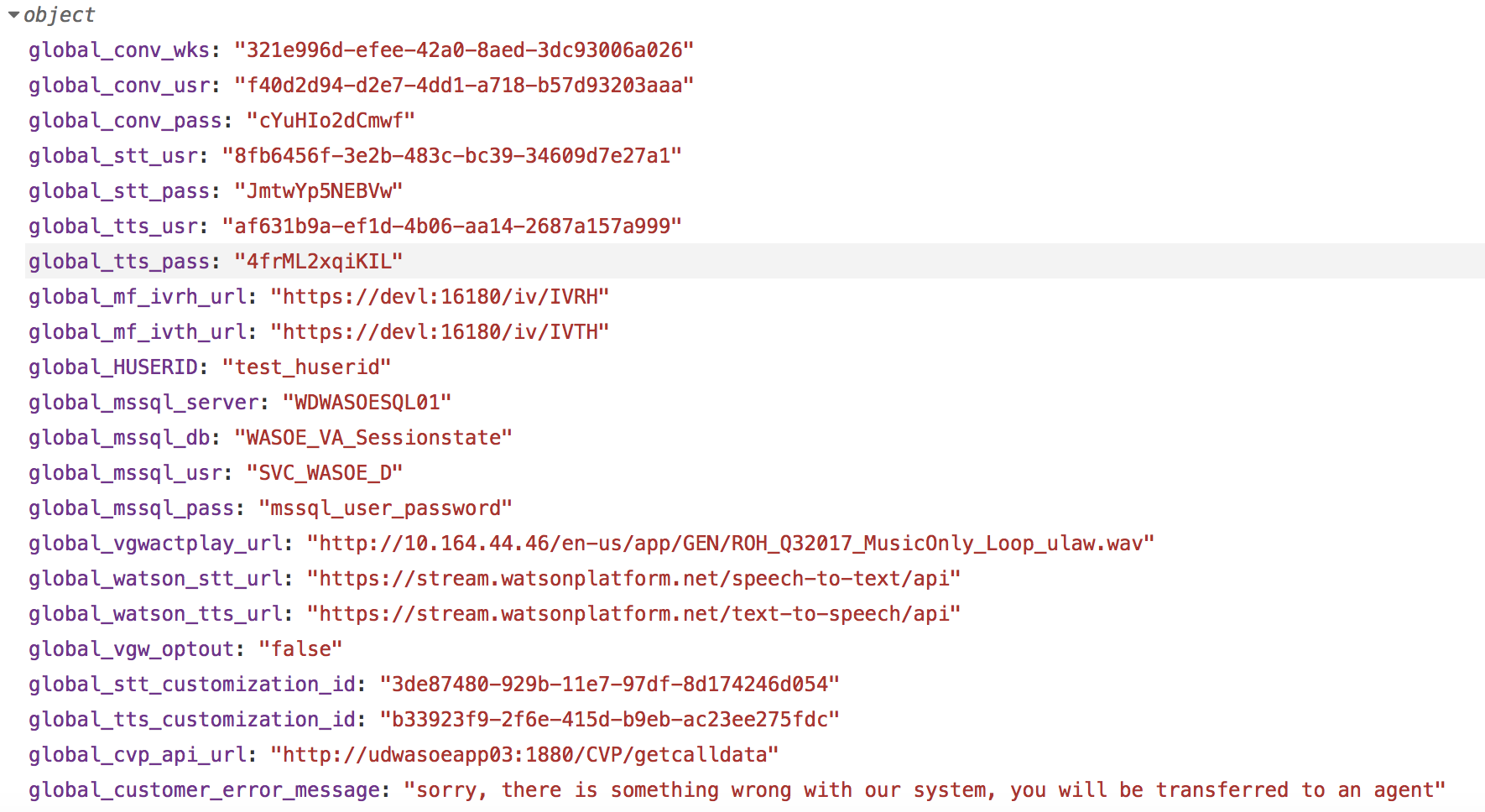
## Test Result- “Successful”

1. **The flows to encrypt and/or decrypt individual string which is passed using an inject node**



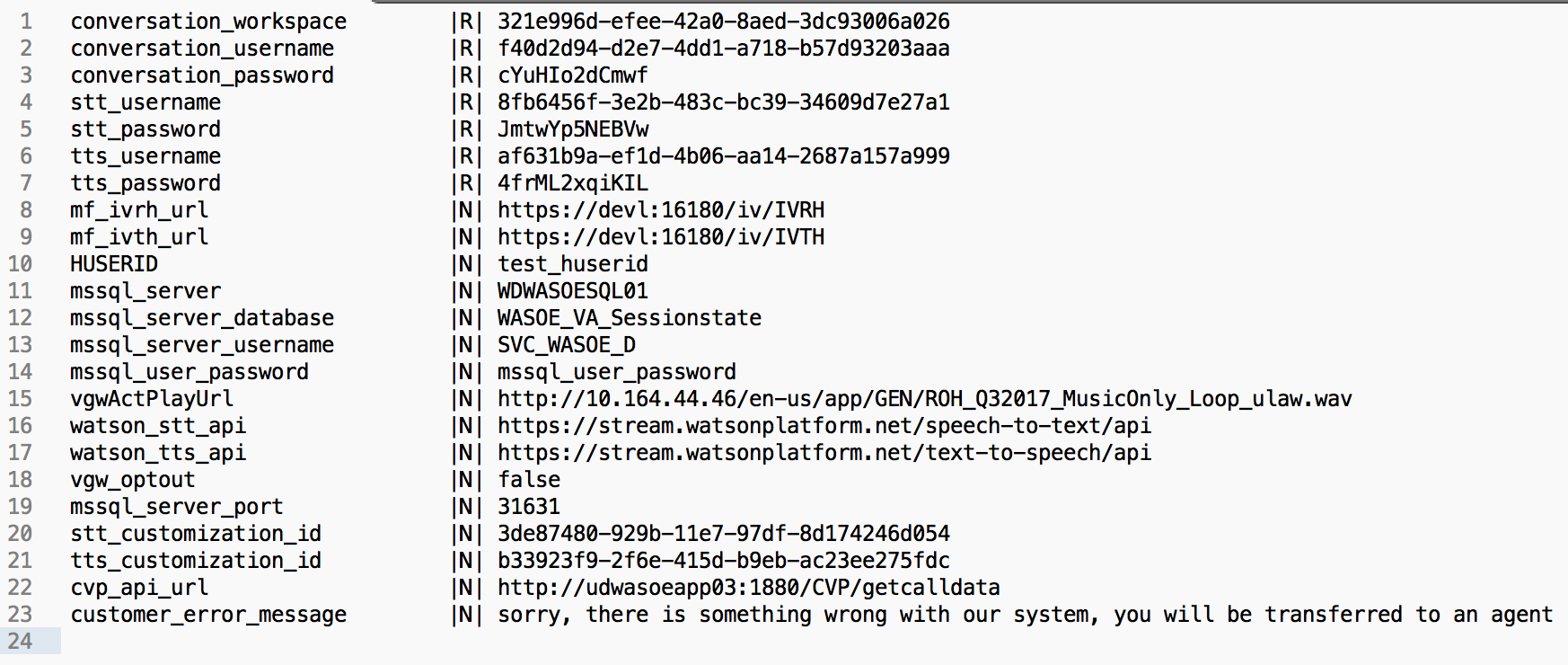


1. **The flow to read SOE environment file and decrypt**

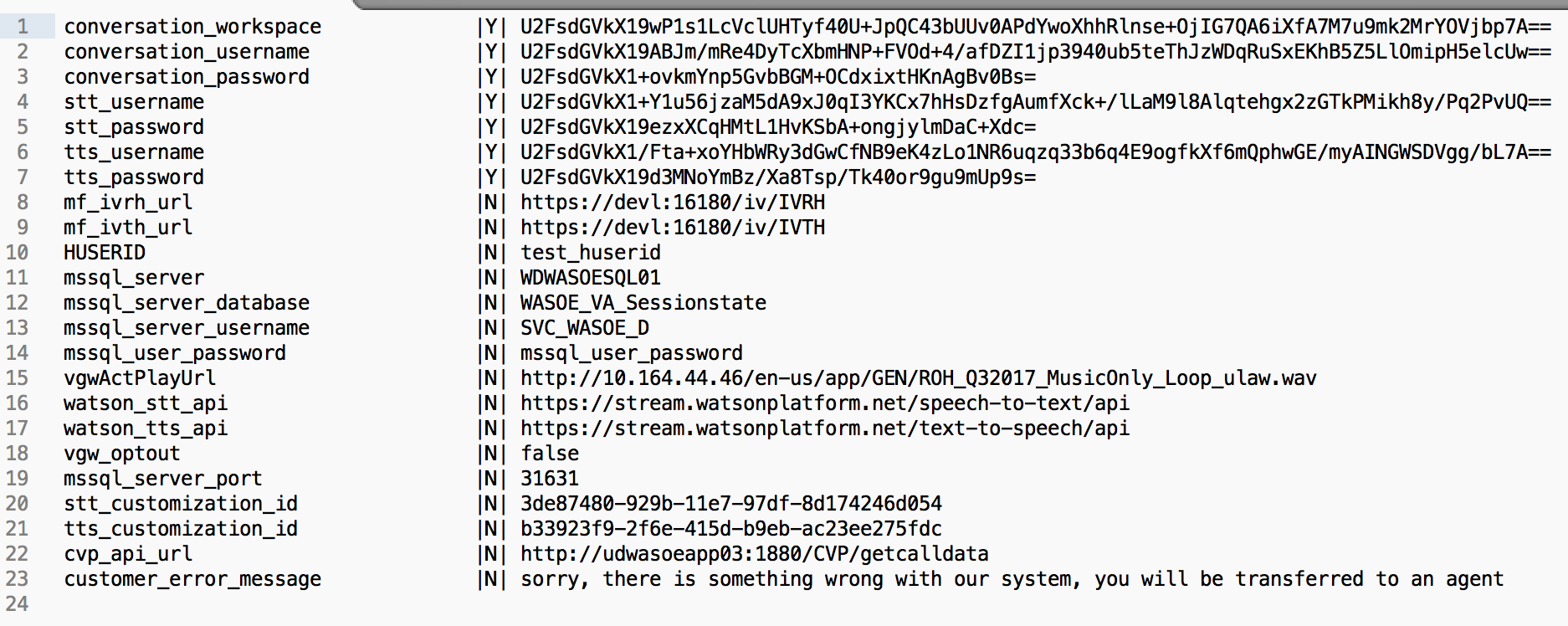


1. **The batch encryption Utility**

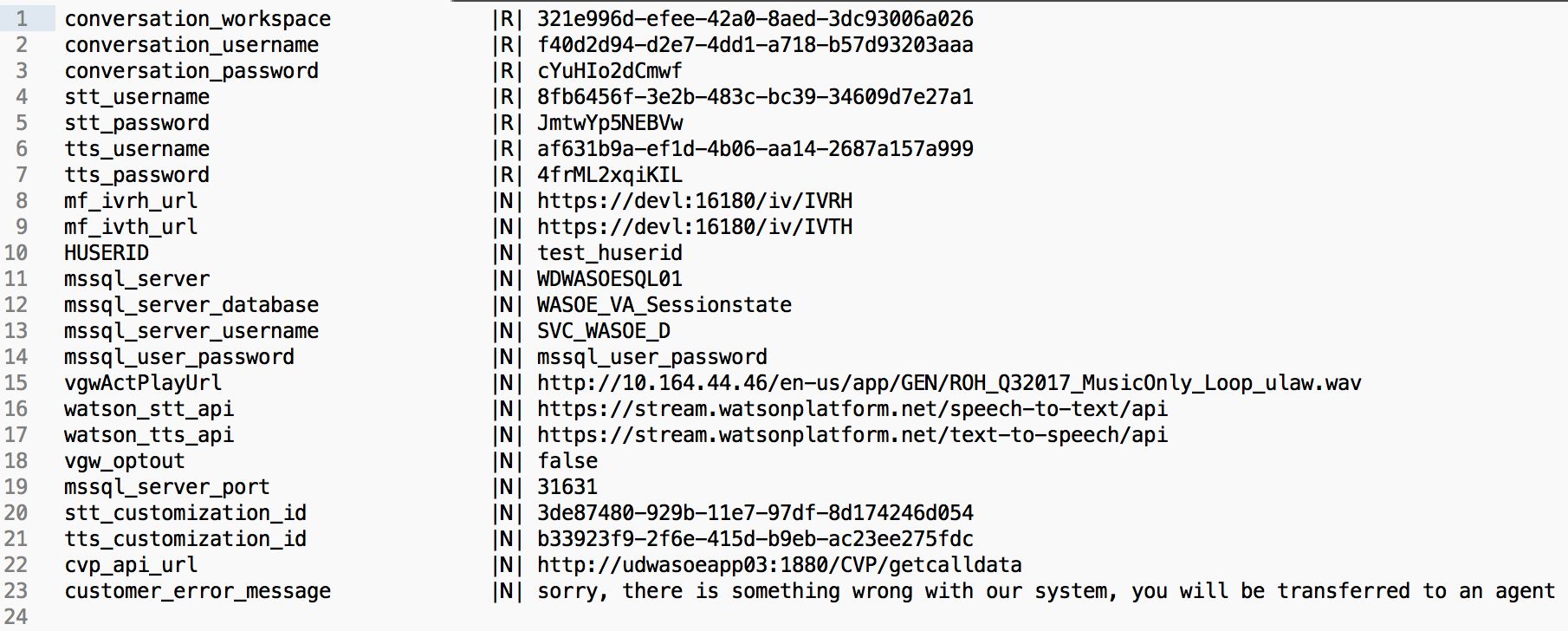
**SOE\_environment\_file\_input:**

****

**SOE\_environment\_file\_output:**

****

**SOE\_environment\_revalidation\_output:**

****

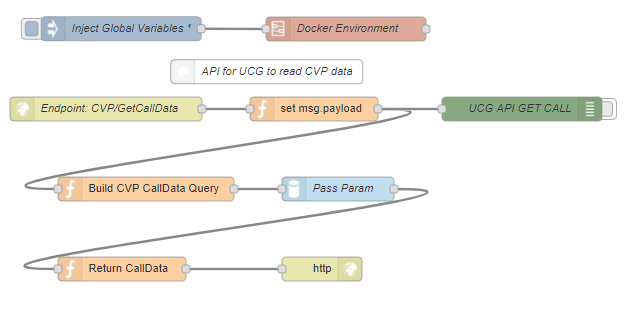
**Note**: The contents of the SOE environment file are discussed later in this document.

# CVP API Service to provide user data

## Overview

This is the Rest API build within Node-Red to provide user data to UCG/SOE. When VGW provides GUCI ID to UCG/SOE as part of call initiation, the SOE invokes this API to get additional details about the user calling in. API returns attributes related to their bank account to process a first fee refund for eligible users if requested.

## CVP API flow – Invoked by UCG/SOE Main flow



|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 1 |  | The inject node, initiate the flow only once when the container starts |
| 5 |  | Loads SOE environment variables in the memory only once when the flow starts. |
| 10 |  | Provides end-point for the CVP Rest API |
| 15 |  | Initializes msg.payload |
| 20 |  | Function node to prepare the SQL Query to fetch the data from watson.CVP table |
| 25 |  | MSSQL-UCG node for SQL Server which executes the query |
| 30 |  | Function Node – Receives the Result from the SQL Node |
| 35 |  | http Response Node |

## How flow works?

URL: “https://udwasoeapp03:2880/CVP/getcalldata”

Input to CVP Rest API: “GUCI\_ID”

Invoke Method: “Get”

Returns: {bank, appl, account, PQ}

## Test Result- “Successful”

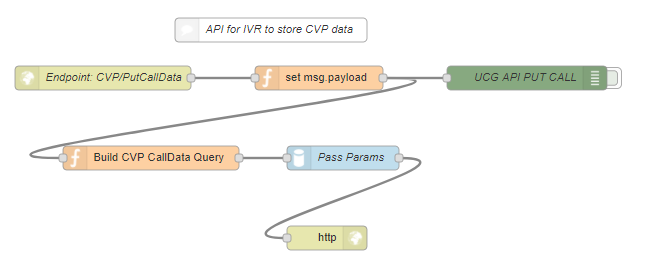
Test Result is included as part of main SOE Test result.

# CVP API flow – Stores user data in CVP

## Overview

This is the Rest API build within Node-Red. It is invoked by the IVR to store user data for the current authenticated user into CVP table. This data is used by the main UCG/SOE flow to process a first fee refund, if requested.

## CVP API flow – Invoked by IVR



|  |  |  |
| --- | --- | --- |
| Sr.# | Node | Comments |
| 1 |  | Provides end-point for the CVP Rest API |
| 5 |  | Initializes msg.payload |
| 10 |  | Function node to prepare the SQL Query to fetch the data from watson.CVP table |
| 15 |  | MSQ SQL UCG node which executes the query |
| 20 |  | http Response Node |

## How flow works?

URL: “https://udwasoeapp03:2880/CVP/PutCalldata”

Invoke Method: “Post”

Returns: Status

# UCG/SOE Main flow

## Overview

This section of the document focuses on the Main UCG/SOE flow. The flow is developed using the Node-red which integrates various components of CSSVB, including Voice Gateway (vgw), Watson Conversation Services (WCS), CVP, and calls the mainframe system for the first fee refund. The flow is deployed as docker/container images running on Linux VM with load-balancing enabled.

The flow reads various environment parameters which are kept in SOE-Environment file. The SOE environment files reside on the NFS share drive, which is mapped to the docker/container as part of volume mapping.

The flow uses: SQL server tables to store CVP data, Call Transfer numbers, Conversation log and session data. ***Please note, data archival from these tables will be handled by Regions as per the enterprise data archival policy.***

The Splunk logging is enabled, to captures exceptions of UCG/SOE as well as Docker/Container.

## SQL Server Table Setup for session management

The following tables have been created in SQL Server Database for the session management and conversation logging.

**Schema/Table: Watson.VARegionsSession**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Comments |
| convoUser | Varchar (255) (PK) | Current conversation user (GUCID) |
| convoTS | DateTime (PK) | Conversation Time for the current exchange |
| convoSource | Varchar(255) | Conversation Source |
| convoPayload | Nvarchar(max) | Holds current value of msg.payload |

**Schema/Table: Watson.VARegionsLog**

| Column Name | Data Type | Comments |
| --- | --- | --- |
| Loggingtimestamp | DateTime (PK) | Conversation Time for the current exchange |
| userid | Varchar(255) (PK) | Conversation userId (GUCID) |
| convoDialogIntents | Char(100) | Current Dialog intent |
| convoDialogConfidence | Decimal(4,3) | Dialog confidence based on Natural Language Processing (NLP) |
| currentInputText | Char(256) | Conversation Input Text |
| currentOuputText | Char(512) | Conversation Output Text |
| currentRepeatRequestCount | Int (10) | Response Repeat Request |
| currentFlowRepeatRequestCount | Int(10) | Flow Repeat Request |
| currentUserID | Char(10) | Employee User ID |
| currentPhoneFrom | Bigint(19) | User Caller ID |
| startTimeStamp | DateTime | Conversation Start Time |
| conversationID | Char(256) | Conversation ID |
| Channel | Char(14) | Conversation Channel |
| convoCounter | Samll Integer | Conversation Counter |
| convoTransID | Varchar(100) | Conversation Transaction ID |
| pqID | Varchar(30) | PQ ID from CVP |
| Call\_Transfer\_Num | Varchar(30) | Call Transfer number when requested |
| convoIntent | Char(100) | Conversation Intent |
| convoDialogNode | Varchar(64) | Dialog Node Name |
| convoDialogType | Varchar(1024) | Dialog Type – Thread |
| convoDialogEntities | Varchar(1024) | Dialog Entities – Thread |
| convoContainementScore | Integer | Containment Score |
| rootRepeatRequestCount | Integer | Number of times root was repeated |

**Schema/Table: Watson.VARegionsCallTransfer**

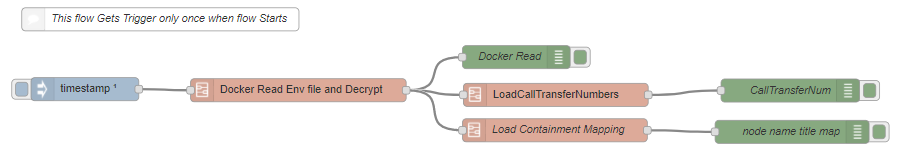
| Column Name | Data Type | Comments |
| --- | --- | --- |
| Transfer\_Route | Integer (PK) | Call Transfer Route ID |
| Transfer\_Phone | Varchar(30) | Call Transfer Number |

**Schema/Table: Watson.CVP**

| Column Name | Data Type | Comments |
| --- | --- | --- |
| ConvoID | Char(256) (PK) | Conversation ID |
| CallData | Varchar(max) | Call Data |
| Time\_Stamp | Datetime | Date/Time of row |

## Flow to Load SOE environment file, Call Transfer Routes and WCS node mapping into memory

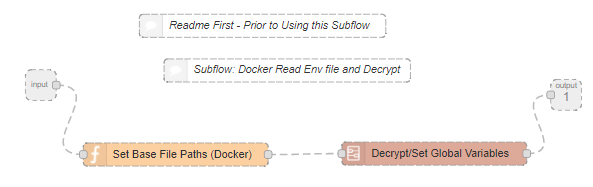
This information is loaded only once in the memory when docker/container starts.



|  |  |
| --- | --- |
| Node | Comments |
|  | The inject node, initiate the flow only once when the container starts |
|  | Subflow to read, decrypt SOE environment file and load it as global variables |
|  | Subflow to load call transfer numbers from VARegionsCallTransfer table into memory to support SOE main flow. |
|  | Subflow to load WCS node ID and Node Name mapping array for call containment |

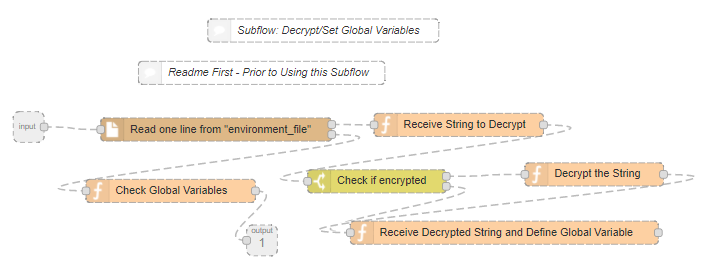
**Note**: *When changes are made to SOE environment file or WCS, either container needs to be restarted or manually execute ‘****Inject Global Variable node’*** *to refresh global variables loaded in the memory.*

### Subflow – Docker Read Environment File (Docker)



|  |  |
| --- | --- |
| Node | Comments |
|  | Input and Output nodes for the subflow |
|  | Set file base file path and name for SOE environment file and password phrase. |
|  | Subflow – to Decrypt string and set global variable |

#### Subflow – Decrypt/Set Global Variables



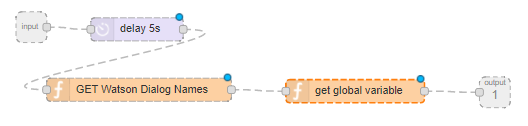
| Node | Comments |
| --- | --- |
|  | Input and Output nodes for the subflow |
|  | Reads one line at time from the SOE environment file and passes to the next node.  Once it reaches end of file – it passes control to Output node. |
|  | Receives one line at a time from SOE environment file; function node split the contents based on delimiter and assign to separate variables. |
|  | Split node checks the string is encryption flag; if ‘Y’ then it sends the strings to decryption node; otherwise it sends to strings to the next node define global variables. |
|  | Decrypt String– Receives encrypted string to decrypt for validation |
|  | Receives decrypted string and defines global variables, which are then available for other nodes in the flow. |
|  | Check Variables and load in msg.payload for verification. |

### Subflow – LoadCallTransferNumbers



| Node | Comments |
| --- | --- |
|  | Input and Output nodes for the subflow |
|  | Inject 5 second delay to allow all environment variables get loaded in memory |
|  | Defines query to extract call transfer numbers from sql server table.  Setup other parameters like database server, port, credentials to be passed to SQL server node to execute the query |
|  | MSQ SQL UCG node which executes the query |
|  | Loads call transfer numbers in memory |

### Subflow – Load Containment Mapping



| Node | Comments |
| --- | --- |
|  | Input and Output nodes for the subflow |
|  | Inject 5second delay to allow all environment variables get loaded in memory |
|  | Reads WCS workspace and creates mapping array for mapping between System (internal) Node ID and Node Title |
|  | Display mapping array for verification |

### SOE Environment Content, Global Variables

The following table provides a list of attributes kept in the SOE environment file, related to the global variable defined within the SOE main flow.

| **SOE Environment File**  **Attribute / Topic** | **Related Global**  **Variable** | **Remark** |
| --- | --- | --- |
| conversation\_workspace | global\_conv\_wks | Watson Conversation Workspace |
| conversation\_username | global\_conv\_usr | Watson Conversation Workspace User ID for Bluemix |
| conversation\_password | global\_conv\_pass | Watson Conversation Workspace password for Bluemix |
| stt\_username | global\_stt\_usr | Watson Speech to Text User ID for Bluemix |
| stt\_password | global\_stt\_pass | Watson Speech to Text Password for Bluemix |
| tts\_username | global\_tts\_usr | Watson Text to Speech User ID for Bluemix |
| tts\_password | global\_tts\_pass | Watson Text to Speech Password for Bluemix |
| mf\_ivrh\_url | global\_mf\_ivrh\_url | Mainframe IVRH URL |
| mf\_ivth\_url | global\_mf\_ivth\_url | Mainframe IVTH URL |
| HUSERID | global\_HUSERID | User ID for IVRH, IVTH API calls |
| mssql\_server | global\_mssql\_server | MS SQL server Name |
| mssql\_server\_port | global\_mssql\_server\_port | MS SQL server port number |
| mssql\_server\_database | global\_mssql\_db | MS SQL server Database Name |
| mssql\_server\_username | global\_mssql\_usr | MS SQL server User Name |
| mssql\_user\_password | global\_mssql\_pass | MS SQL server user password |
| vgwActPlayUrl | global\_vgwactplay\_url | VGW Media Play URL |
| watson\_stt\_api | global\_watson\_stt\_url | Watson Speech to Text API URL |
| watson\_tts\_api | global\_watson\_tts\_url | Watson Text to Speech API URL |
| vgw\_optout | global\_vgw\_optout | VGW output Default |
| stt\_customization\_id | global\_stt\_customization\_id | STT custom model ID |
| tts\_customization\_id | global\_tts\_customization\_id | TTS custom model ID |
| cvp\_api\_url | global\_cvp\_api\_url | CVP api url |
| customer\_error\_message | global\_customer\_error\_message | Custom Error message to be played for user, if SOE needs to initiate and redirect call to live agent instead of WCS |
| timezone | global\_timezone | Time Zone |
| sip\_dns\_string | global\_sip\_dns\_string | SIP DNS String for call transfer |
| vgw\_timeOut\_default | global\_vgw\_timeOut\_default | Vgw default time-out value |
| vgw\_firmup\_silence\_time | global\_vgw\_firmup\_silence\_time | Vgw firm-up silence time-out |
| vgw\_jitterBufferDelay | global\_vgw\_jitterBufferDelay | Vgw Buffer Value |
| vgw\_cacheTimeToLive | global\_vgw\_cacheTimeToLive | VGW Cache refresh value |
| vgw\_dtmfCount | global\_vgw\_dtmfCount | VGW DTFM default count |
| mf\_timeOut | global\_mf\_timeOut | Default time-out for mainframe APIs |
| mssql\_timeOut | global\_mssql\_timeOut | Default time-out for MSSQL server calls |
| cvp\_timeOut | global\_cvp\_timeOut | Default time-out for CVP calls |
| watson\_timeOut | global\_watson\_timeOut | Default time-out for WCS |
| containment\_success\_score | global\_containment\_success\_score | Call Containment Success Scoring |
| containment\_fail\_score | global\_containment\_fail\_score | Call Containment Fail scoring |
| containment\_high\_conf\_start | global\_containment\_high\_conf\_start | Call Containment High Confidence lower limit |
| containment\_med\_conf\_start | global\_containment\_med\_conf\_start | Call Containment Medium Confidence lower limit |
| containment\_low\_conf\_start | global\_containment\_low\_conf\_start | Call Containment Low Confidence lower limit |
| containment\_no\_conf\_start | global\_containment\_no\_conf\_start | Call Containment No Confidence lower limit |
| containment\_high\_conf\_score | global\_containment\_high\_conf\_score | Call Containment High Confidence score |
| containment\_med\_conf\_score | global\_containment\_med\_conf\_score | Call Containment Medium Confidence score |
| containment\_low\_conf\_score | global\_containment\_low\_conf\_score | Call Containment Low Confidence score |
| containment\_no\_conf\_score | global\_containment\_no\_conf\_score | Call Containment No Confidence score |
| default\_route\_auth | global\_default\_route\_auth | Default Call transfer Route for authenticated users |
| default\_route\_nonauth | global\_default\_route\_nonauth | Default Call Transfer Route for Non-authenticated users |
| vgwSessionInactivityTimeout | global\_vgwSessionInactivityTimeout | Vgw Inactivity time-out |
| httpNodeBasicAuthUsername | N/A | HTTPS Node Base Authentication User Name |
| httpNodeBasicAuthUsername | N/A | HTTPS Node Base Authentication User Password |
| disableNodeRedEditor | N/A | Disable Node Red Editor UI. Valud Values true, false |
| N/A | global\_call\_transfer\_num | Call Transfer Number Array |
| N/A | node\_name\_title\_map | WCS Node-ID and Node-Name Mapping Array |

### Context Variables / vgw Action Sequence parameters

The Context variables are used to exchange supporting information between SOE and WCS and SOE and VGW. The context variables help WCS navigate through dialogs and select an appropriate response based on utterances. WCS sets context variables to control the SOE flow as well as the behavior of VGW.

vgwActionSequence parameters are set by SOE to be sent to vgw. The default values for these parameters are taken from the SOE environment file, however these can be overwritten by WCS. These variables control the behavior of vgw.

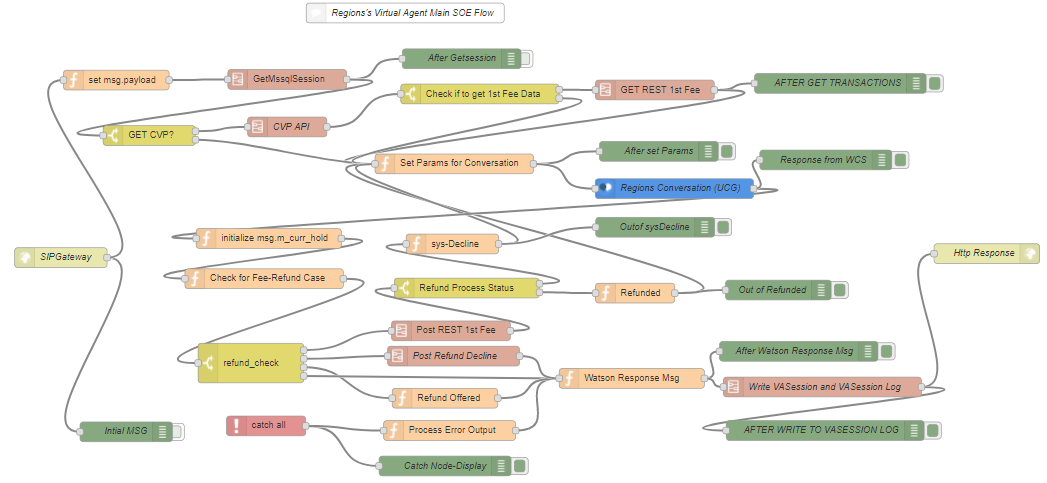
The following table gives a list of key context variables and vgwActionSequence:

|  |  |
| --- | --- |
| Context Variable | Values / Comments |
| vgwSIPCustomInviteHeader | VGW use this variable to set GUCID to SOE, if user is authenticated user. |
| msg.params.context.authenticated | SOE sets this variable based on a) GUCID passed by VGW and if CVP data is populated.  WCS receives this to guide the conversation.  “**Yes**” – Calling user is authenticated user and potential candidate for the fee refund.  “**No**” – Calling user is not authenticated user, hence conversation will be for Online Banking FAQ |
| msg.params.context.fee\_eligibility | This variable is set by SOE and passed to WCS.  “Yes” – User is eligible for 1st fee refund  “No” – User is not eligible for 1st fee refund |
| msg.params.context.fee\_refund\_status | This variable is set by SOE and passed to WCS; depending on conversation, WCS sets the value and passes it back to SOE.  “**None**” – SOE sets to notify WCS – No fee is available to refund  “**Ready**” – SOE sets to notify WCS - Fee is available for refund if customer requests.  “**cust-Offered**” – WCS sets to notify SOE that the customer has asked for the refund and in response WCS has offered the refund the 1st fee and requested customer to confirm to proceed with refund processing.  “**cust-Decline**” – WCS sets to notify SOE that customer has declined the refund.  “**cust-Accepted**” - WCS sets to notify SOE that customer has accepted the refund.  “**sys-Decline**” – SOE sets this to notify WCS that it encountered some technical error while processing refund.  “**Refunded” –** SOE sets this to notify WCS that the refund was processed successfully. |
| BargeIn | SOE sets default value based on SOE environment file. WCS can overwrite this with “Yes” to allow user to interrupt conversation when CSSVB is talking if necessary. |
| TransferRouteID | WCS sets this for SOE to retrieve appropriate call transfer number if the call transfer is to be initiated |
| vgwHangUp | WCS sets this to pass to SOE/VGW to initiate call hang-up and end the conversation. |
| TimeOut | Default timeout for vgw based on SOE environment file. WCS can overwrite the default value based on intent/response if necessary. |
| firmup\_silence\_time | Amount time vgw will wait before start processing user utterance. Default is set from SOE environment file, which WCS can overwrite if necessary. |
| vgwActSetSTTConfig | Configuration parameters for Speech to Text service |
| vgwActSetTTSConfig | Configuration parameters for Text to Speech service |
| vgwActCollectDtmf | Number of digits to collect in DTMF (dual tone multi frequency) mode |
| vgwActDisableSpeechBargeIn / vgwActEnableSpeechBargeIn | Disable / Enable Speech Barge In |
| vgwActPauseSTT / vgwActUnPauseSTT | Pause / Un-pause Speech to Text processing |
| vgwActPlayText | Message to play by VGW |
| vgwActTransfer | Call Transfer parameters |

## SOE/UCG Main flow

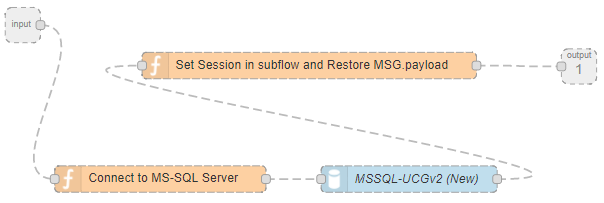
The main SOE/UCG flow is secured with SSL certificates with bi directional authentication enabled for https call. The ***https auth userId*** and ***password*** is stored in SOE environment file and can be encrypted using crypto-JS. When a customer calls in and call gets directed to CSSVB, VGW invokes the flow using https URL and bi-authentication user ID & password.

The call-in user can be either authenticated or un-authenticated. In case of authenticated user, IVR generates temporary user ID (GUCID) and invoke CVP flow to write User’s bank account information to CVP table resides in SQL server database. VGW invokes the main SOE flow and passes GUCID as vgwSIPCustomInviteHeader, (if user is authenticated user), vgwSIPFromURI & vgwSessionID.



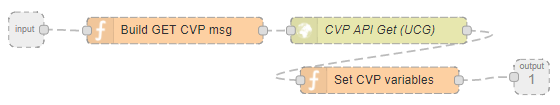
| Sr.# | Node | Comments |
| --- | --- | --- |
| 1 |  | Entry point for UCG/SOE flow.  End point URL : */SIPGateway/v1/workspaces/321e996d-efee-42a0-8aed-3dc93006a026/message* **(Do not Change)** |
| 5 |  | Display msg object received from vgw |
| 10 |  | Defines/initializes objects for the conversation includes- msg.getsession, msg.curr\_hold, msg.last\_convo\_payload, msg.containment.  Saves msg.curr\_hold = msg.payload.  This node also assigns session user ID, in case of authenticated user, GUCID (vgwSIPCustomInviteHeader) is used as session User ID whereas in case of unauthenticated user, the node generates session user ID from vgwSessionID. |
| 15 |  | The subflow retrieves last session data for the current session User ID from vaRegionsSession table and assigns it to msg.getssion object. If there is a row that exists for the current Session User ID in vaRegionsSession, that indicates it is continuation of current conversation, else it is new conversation.  The node also set the flag getsession.cvp\_get\_call = “Yes” if session user is Authenticated user else sets it “No” |
| 20 |  | Display msg object after GetMssqlSession subflow |
| 25 |  | The Split node, redirect flow based on getsession.cvp\_get\_call. When the value is “Yes’, this indicates, the user is authenticated user and the flow continues to retrieve user bank data from CVP table.    If getsession.cvp\_get\_call. = “No”, this indicates, the customer is not authenticated and the flow continues for Online Banking – FAQ conversation. |
| 30 |  | The subflow retrieves, the session users’s bank data from CVP table |
| 35 |  | The Split node redirect flow based on the flag  getsession.cvp\_get\_call. When ‘Yes’ the flow proceeds to retrieve customer’s refund data.  If getsession.cvp\_get\_call = “No”, the flow proceeds for continues for Online Banking FAQ conversation. |
| 40 |  | The Subflow to check 1st fee refund eligibility and retrieves refund amount. |
| 45 |  | Display msg object after retrieving 1st fee data. |
| 50 |  | Sets parameters for invoking Watson Conversation Service API includes-  WCS credentials and Workspace ID  User’s input (utterance)  msg.params.context.authenticated  msg.params.context.fee\_eligibility  msg.params.context.fee\_refund\_status |
| 55 |  | Display msg object being sent to WCS |
| 60 |  | Watson Conversation Service Node |
| 65 |  | Display msg object received from WCS |
| 70 |  | Save msg.m\_curr\_hold = msg.payload |
| 75 |  | Sets the flag msg.refund\_check as msg.payload.context.fee\_refund\_status  If 1st fee Refund is already processed then it sets- msg.refund\_check = "No" |
| 80 |  | The split node divert the flow based on msg. refund\_check |
| 85 |  | This node is executed if refund\_check = “***cust-Offered***’. The node replaces following variable with actual values in WCS response message-   1. ‘**fee\_refund\_amount’** with refund amount from ***msg.getsession.fee\_refund.fee\_refund\_amount*** 2. ‘**fee\_date**’ with fee date from ***msg.getsession.fee\_refund.fee\_date*** |
| 90 |  | This subflow is executed if **refund\_check** = “***cust-Accepted***’ |
| 95 |  | This subflow is executed if **refund\_check** = “***cust-Decline***’ |
| 100 |  | This subflow is executed-   1. if **refund\_check** value does not match to any other above value 2. As part of continuation of the flow after executing the subflow ‘Post Refund Decline’ or ‘Refund Offered’ 3. “processed Error Output’ |
| 105 |  | The Split diverts the flow based on –  **getsession.fee\_refund.fee\_refund\_status** |
| 110 |  | This node is executed when **getsession.fee\_refund.fee\_refund\_status = ‘*Refunded*’**  The node sets context variable **msg.params.context.fee\_refund\_status** = "***Refunded***" to notify WCS.  The node also sets  **msg.getsession.fee\_refund.fee\_refund\_status** = "***processed-Refund***";  Which prevents processing another refund within a same conversation flow.  The flow is redirected to the node ‘Set Params for Conversation’ |
| 115 |  | Display msg object after execution of the node ‘Refunded’ |
| 120 |  | This node is executed when **getsession.fee\_refund.fee\_refund\_status = ‘*sys-Decline’***  The node sets context variable **msg.params.context.fee\_refund\_status** = "***sys-Decline***" to notify WCS.  The node also sets  **msg.getsession.fee\_refund.fee\_refund\_status** = "***processed-sys-Decline***";  Which prevents customer requesting refund again when system has ran into technical issue.  The flow is redirected to the node ‘Set Params for Conversation’ |
| 125 |  | The node performs following-   1. If refund is successfully posted then it substitute variable ‘***refund\_post\_date***’ in WCS response with refund post date returned by Rest API 2. If WCS and initiated the call Transfer to the banker, this node retrieves the call transfer number from the ‘***global\_call\_transfer\_num***’ array using Transfer Route provided by WCS, and sets vgwActionSequenceVariables accordingly. 3. Node sets vgw specific values loaded from SOE environment file for the following context variables and sends to vgw along with the response message.  * vgwActSetSTTConfig * vgwActSetTTSConfig * jitterBufferDelay * cacheTimeToLive * vgwActCollectDtmf * vgwActDisableSpeechBargeIn or vgwActEnableSpeechBargeIn * vgwActPauseSTT or vgwActUnPauseSTT * vgwActPlayText (response meesage) * vgwPostResponseTimeoutCount * vgwSessionInactivityTimeout * vgwHangUp |
| 130 |  | Display msg object set to be sent to vgw |
| 135 |  | The subflow to write session data and conversation log to the tables in SQL server database. |
| 140 |  | Display msg object set to be sent to vgw after session data and conversation log was written. |
| 145 |  | HTTPS Response node |
| 150 |  | This node catches technical issues encountered during the execution of the flow. |
| 155 |  | This node writes the error caught by ‘Catch all’ node to the Splunk. |
| 160 |  | Display error captured by Catchall node |

### Subflow : (15-GetMssqlSession)



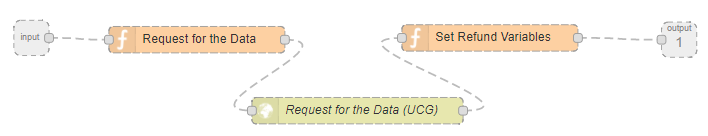
| Sr.# | Node | Comments |
| --- | --- | --- |
| 15.1 |  | Input and Output nodes for the subflow |
| 15.3 |  | Defines query to select last session data for the current session User ID. |
| 15.5 |  | MS Sql Server Node which executes the query against SQL server database. |
| 15.7 |  | Receives the data return by SQL node and sets msg.getsession for the subsequent flow. It also restores msg.payload from msg.m\_curr\_hold |

### Subflow: (30-CVP API)



| Sr.# | Node | Comments |
| --- | --- | --- |
| 30.1 |  | Input and Output nodes for the subflow |
| 30.3 |  | Node sets parameter to make CVP Rest API call to retrieve session user’s bank account information. |
| 30.6 |  | HTTP node which makes Rest API call. The CVP call requires TLS and HTTPs basic authentication. |
| 30.9 |  | Node validates if user’s bank data is available and sets the flag getsession.cvp\_data.authenticated = “Yes’ or “No” |

### Subflow : (40-Get Rest 1st Fee)



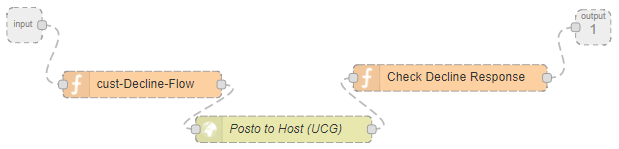
| Sr.# | Node | Comments |
| --- | --- | --- |
| 40.1 |  | Input and Output nodes for the subflow |
| 40.3 |  | Node sets parameter to make 1s Fee Data retrieval Rest API call |
| 40.6 |  | HTTP node which makes Rest API call. The 1st API call requires TLS and HTTPs authenticated user ID (no password is needed) |
| 40.9 |  | Node validates 1st fee data returned by API to validate if the customer is eligible to receive refund and saves the related data under msg.getsession. If the customer is eligible to receive refund then following flags are set-  fee\_refund.fee\_eligibility = "Yes";  fee\_refund.fee\_refund\_status = "Ready"  fee\_refund.fee\_refund\_amount =  maxRefund.amount  fee\_refund.fee\_date = Refund Date. |

### Subflow : (90-Post REST 1st Fee)



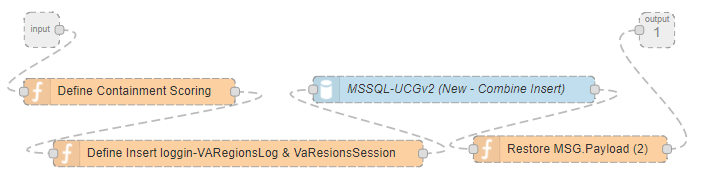
| Sr.# | Node | Comments |
| --- | --- | --- |
| 90.1 |  | Input and Output nodes for the subflow |
| 90.3 |  | Node sets parameter to make 1s Fee Data post Rest API call. This will post the refund to the customer account. |
| 90.6 |  | HTTP node which makes Rest API call. The 1st API call requires TLS and HTTPs authenticated user ID (no password is needed) |
| 90.9 |  | Node validates the return result of 1st fee Post Request call and sets following variable based on the result.-  If refund is successfully posted then it sets-  **getsession.fee\_refund.fee\_refund\_status**= ‘***Refunded’***  If refund is not successfully posted then it sets-  **getsession.fee\_refund.fee\_refund\_status**= ‘***sys-Decline*** |

### Subflow : (95-Post Refund Decline)



| Sr.# | Node | Comments |
| --- | --- | --- |
| 95.1 |  | Input and Output nodes for the subflow |
| 95.3 |  | Node sets parameter to make Rest API call to record ‘***customer decline refund’*** for the reporting. |
| 95.6 |  | HTTP node which makes Rest API call. The call requires TLS and HTTPs authenticated user ID (no password is needed) |
| 95.9 |  | Node validates the return result of the REST Api call, based on the return value it sets following parameters.  If the ‘cust-Decline’ action was successfully posted then it sets-  **getsession.fee\_refund.fee\_refund\_status**= ‘***processed-cust-Decline’***  If the ‘cust-Decline’ action was not successfully posted then it sets-  **getsession.fee\_refund.fee\_refund\_status**= ‘***processed-cust-Decline-failed’***  The flow continues to the next node. |

### Subflow : (135-Post Refund Decline)



| Sr.# | Node | Comments |
| --- | --- | --- |
| 135.1 |  | Input and Output nodes for the subflow |
| 135.3 |  | Node stores msg.payload to msg.m\_curr\_hold.  msg.m\_curr\_hold = msg.payload;  Node then uses Node\_ID and Node-Name mapping array to derive Node-Name for the current node, calculate containment scoring based on following-   * Confidence Value * Success or failure of the conversation thread * Containment computation factors loaded in memory from SOE environment file   The node populates containment scoring object msg.containment which is passed to the next node for writing containment score for each dialog to VARegiosnlog SQL server table |
| 135.6 |  | Defines query to insert the row to following sql server tables-   * VARegionsSession – stores current msg.payload and msg.getsession * VARegionsLog – stores current conversation log and conversation containment score |
| 135.9 |  | MS Sql Server Node which executes the query against SQL server database. |
| 135.12 |  | Restores msg.payload |

# Enabling Splunk Logging

## Overview

Regions uses Splunk to monitor and analyze application logs. SOE logging is enabled @Docker/Container level to redirect logs generated by SOE as well as any exceptions those may be generated @Docker/Container level. During Sprint-1 & Sprint-2, the SOE log was being captured in “status.log” file which resides on the host. With an enabling of Splunk Logging, the usage of “status.log” has been disabled.

## How to enable Splunk Logging?

Splunk logging is enabled when the container is created. The following example shows parameters to be passed to enable Splunk Logging.

Docker run  -it

   -v /Users/nitinkul/Documents/nitinkulData-Dev/Docker/Dev/supportingFiles:**/usr/src/node-red/keyfiles**

   -v /Users/nitinkul/Documents/nitinkulData-Dev/Docker/Dev/logs:/usr/src/node-red/logs

**--log-driver=splunk**

**--log-opt splunk-token=<splunk http event collector token>**

**--log-opt splunk-url=<splunk http event collector end-point>**

**--log-opt splunk-insecureskipverify=true.       <Use this option if SSL is enabled>**

**--log-opt tag="{{.Name}}/{{.FullID}}"**

    -p 1880:1880   --name <container name>  <repository>/image name

## Test Result- “Successful”



Attached Splunk Extract.



# Error handling and Time-out Handling Strategy for SOE

IBM error handling and time-out handling strategy for SOE addresses the following types of exceptions:

* Business Exceptions: In this case SOE will notify WCS about the exceptions and WCS will take action and intimate the caller/customer.
* Technical Exceptions: In this case, the technical error will be caught by the ‘Catch’ node and will write exceptions to the Splunk and send generic message to vgw to intimate the caller/customer. The technical errors are:

**SOE Error Handling Table:**

| **Error Source** | **Scenario** | **Error Handler** | **Resolution** |
| --- | --- | --- | --- |
| 1st Fee API | 1st Fee API returns no rows. | SOE will mark customer is not eligible for fee refund and will set context variables for WCS. | WCS will direct the conversation based on fee refund eligibility flag. |
| 1st Fee Refund post fails | User is eligible for refund but refund process fails | SOE will mark flag the case ‘sys-Decline’ | SOE will write the error to Splunk.  SOE will play standard message to customer and will initiate the call transfer to the live agent. |
| CVP call fails | CVP call fails and unable to retrieve user Info | SOE Catch Node | SOE will write the error to Splunk.  SOE will play standard message to customer and will initiate the call transfer to the live agent. |
| Cust Decline fails | CustDecline call fails to post the customer decline action for reporting | SOE Catch Node | SOE will write error to Splunk.  SOE will play standard message to customer and will initiate the call transfer to the live agent. |
| MSSQL server call fails | SOE unable to retrieve session information | SOE Catch Node | ***SOE will write the error to Splunk. And will initiate the call transfer to the live agent.*** |
| MSSQL server call fails | SOE unable to write to Regions log or session table | SOE Catch Node | ***SOE will write the error to Splunk.***  ***Catch node will restore the msg.payload object and restore the flow so call continues*** |
| WCS | Watson Service call fails | SOE Catch Node | SOE will write the error to Splunk.  SOE will play standard message to customer and will initiate the call transfer to the live agent. |

## Time-outs within SOE:

Time-outs values SOE integration points will be kept with SOE environment files which can be configured as per Region’s standard.

|  |  |
| --- | --- |
| **SOE environment file Topic** | **Value (in Millisecond)** |
| cvp\_timeOut | 30000 |
| watson\_timeOut | 30000 |
| mf\_timeOut | 30000 |
| mssql\_timeOut | 30000 |
| vgw\_timeOut\_default | 30000 |

## Time-outs within vgw:

**Call Flow1:** Inbound Caller from SIP PSTN Toll Free Trunk to Cisco Voice Portal (CVP), redirected to Watson:

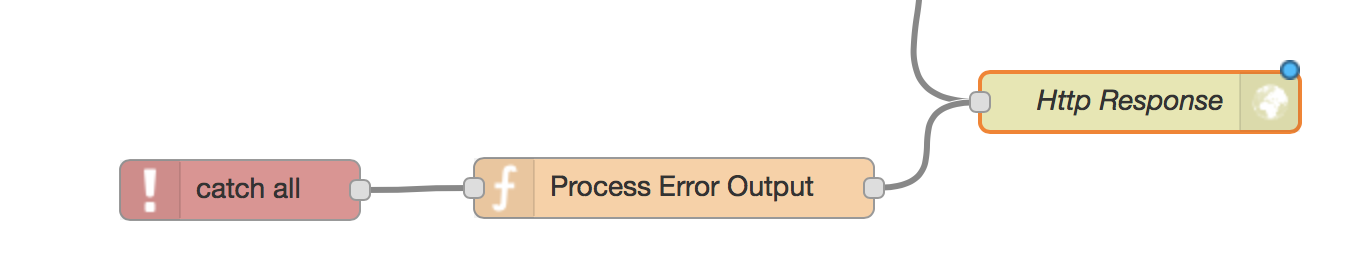
* SBC Sends INVITE to Cisco Unified Proxy (CUSP)
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure at 32s
* CUSP Sends INVITE to CVP
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure > 3 min
* Refer from CVP to SBC
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure at 32s
* INVITE from SBC to CUSP
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure at 64 \* 500ms
* CUSP to VGW
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure > 3 min

**Call Flow2**: Transfer from Watson back to UCCE Agent

* Refer from VGW to SBC
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure at 32s
* INVITE from SBC to CVP
  + 500ms
  + Retrans at 500ms
  + Transaction Timeout Failure at 64 \* 500ms

## SOE flow for catch node

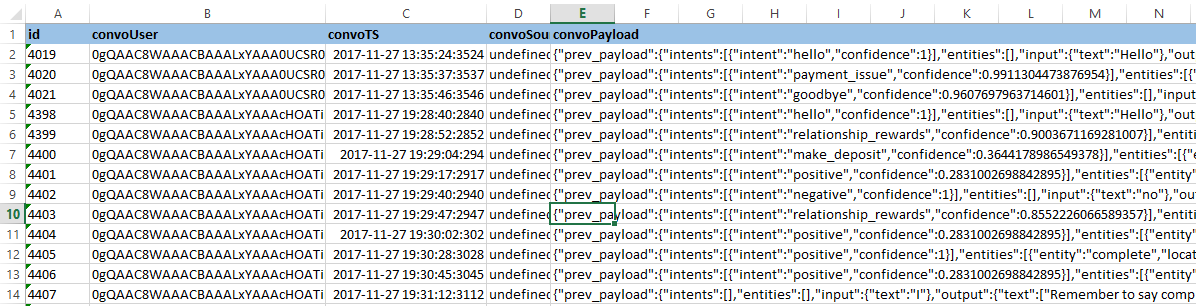
**Sample flow for Catch Node:**



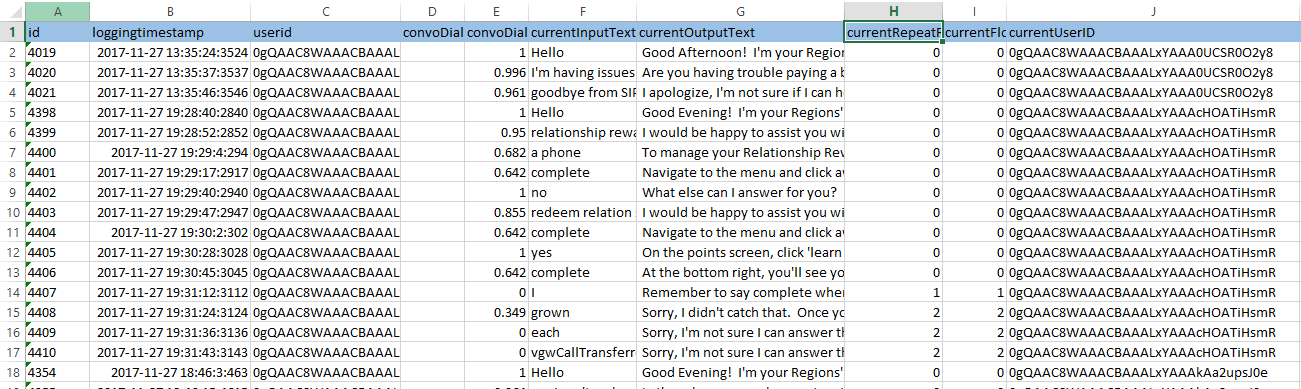
|  |  |
| --- | --- |
| Node | Comments |
|  | Node will catch technical exceptions in the flow |
|  | ***If the issue has occurred due to SQL server connection then-***  ***SOE will write the error to Splunk.***  ***Catch node will restore the msg.payload object and restore the flow so call continues.***  ***If the issue occurred due to some other tech issue then***  ***Function node will write exceptions to Splunk as well as prepare message to send to vgw to intimate to the caller and will initiate the call transfer.*** |
|  | HTTP Response node |
|  | ***SOE will write the error to Splunk.***  ***Catch node will restore the msg.payload object and restore the flow so call continues*** |

# Test Result – vgwLog and vgwSession output captured in SQL server database

**VgwSessionlog**



**vgwLog (conversation log)**

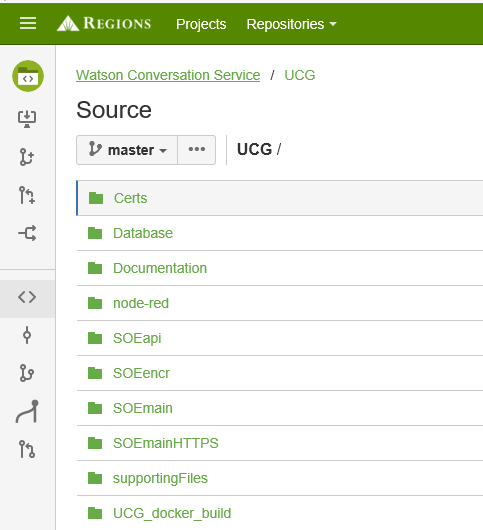


**Note**: Did not attach excel files or screenshot of Refund getting posted as it will have sensitive information.

# Instructions to create Sprint5 docker images

Please download following packages from the ALM sites-

<https://alm.rgbk.com/source/projects/WASOE/repos/ucg/browse>



|  |  |
| --- | --- |
| **File Name** | **Comments** |
| setting.js | This file contains various settings, parameters for the node-red UI. |
| Dockerfile | File to generate Docker Images; this file also install npm packages (which are required by the flow) when docker image gets created. |
| Soe\_main\_flow.json | Node-Red flow for SOE main flow |
| soe\_api\_flow.json | Node-Red flow for CVP API flow |
| soe\_envfile\_encryption\_flow.json | Node-Red flow for Environment file encryption |
| httpBasicAuth.js | This file includes a function that serves as the middleware for checking for basic authentication for http requests  The functions looks in the environment file for "httpNodeBasicAuthUsername" and "httpNodeBasicAuthPassword".  If password is encrypted, it will decrypt the password and compare it with the password that's attached to the http header |
| environmentFileConfig.js | This file includes a function that reads the SOE environment file and looks for "disableNodeRedEditor"  if disableNodeRedEditor is "true", we will not be able to access the node-red-ui  if disableNodeRedEditor is "false" or the line doesn't exist , we will be able to access the node-red-ui |
| fileProcessUtility.js | This file includes supporting functions required by httpBasicAuth.js & environmentFileConfig.js. |
| SOE\_environment\_file | SOE environment file – already covered earlier in the document |
| Password\_Phrase.txt | Password phrase file – already covered earlier in the document |
| Load Call Transfer Number.txt | SQL Script to load call transfer numbers to SQL server table |

|  |  |  |  |
| --- | --- | --- | --- |
| **Flow & Package Name** | **Includes** | **Not Included** | **HTTPS Port to expose** |
| **SOE Main Flow 🡺**  SOE\_Main\_Flow\_Package.zip | Dockerfile  settings.js  soe\_main\_flow.json  httpBasicAuth.js  environmentFileConfig.js  fileProcessUtility.js  SOE\_environment\_file  Password\_Phrase.txt | SSL Certificates | 1881 |
| **CVP Flow 🡺**  CVP\_Package.zip | Dockerfile  settings.js  soe\_api\_flow.json  environmentFileConfig.js  fileProcessUtility.js  SOE\_environment\_file  Password\_Phrase.txt  (***No basic authentication for AP***I) | SSL Certificates | 2881 |
| **SOE Encryption Flow 🡺**  SOE\_Encryption\_Flow\_Package.zip | Dockerfile  settings.js  httpBasicAuth.js  environmentFileConfig.js  fileProcessUtility.js  soe\_envfile\_encryption\_flow.json  SOE\_environment\_file  SOE\_environment\_file\_input  Password\_Phrase.txt | SSL Certificates | 3881 |

## Build Docker Images for each Flow

### Folders on Host server (NFS Share)

1. **Folder to Store Supporting Files**

**e.g.** /Users/nitinkul/Documents/nitinkulData-Dev/Docker/**Dev**/supportingFiles

**Pull the following files from the Regions Repository and copy to supportingFiles Folder on NFS**

* Password\_phrase.txt
* SOE\_environment\_file
* SSL Certificates

Note: Folders are already setup, hence no action needed for this step.

1. **Folder to Store log files**

      e.g. /Users/nitinkul/Documents/nitinkulData-Dev/Docker/Dev/logs

Note: ***This folder is no longer needed with Splunk enablement.***

1. **Create folder on host to build the image instead of pulling from repository and copy following files in the folder.**

* Create build Docker folder –
  + for main SOE Flow, Folder Name:  “**build\_docker\_main**”
  + for CVP flow, Folder Name: “**buid\_docker\_cvp**”
  + for SOE environment encryption, flow Name: “**build\_docker\_encrypt**”
* Download and extract files from each package and save following files in respective Docker build folder-

|  |  |  |
| --- | --- | --- |
| **SOE Main flow** | **CVP Flow** | **SOE Environment file Encryption Flow** |
| DockerFile  soe\_main\_flow.json  Settings.js  httpBasicAuth.js  environmentFileConfig.js  fileProcessUtility.js  ***HTTPS Port: 1881*** | DockerFile  Soe\_api\_flow.json  Settings.js  environmentFileConfig  fileProcessUtility.js  ***HTTPS Port: 2881***. | DockerFile  soe\_envfile\_encryption\_flow.json  settings.js  httpBasicAuth.js  environmentFileConfig  fileProcessUtility.js  ***HTTPS Port:3881*** |

***Note****: Pl. make sure https port number is correctly defined in respective settings.js and DockerFile of each flow*

1. **Generate the Image from the Docker file-**

**Docker build** --no-cache -t  <repository>/image name .

**e.g.** Docker build --no-cache -t  nitinkul/nk2\_20170820\_01

### Build Container with Volume Mapping

**Note**:

* SupportingFiles folder from the host must be mapped to ‘/usr/src/node-red/keyfiles’

  Docker run  -it

   -v /Users/nitinkul/Documents/nitinkulData-Dev/Docker/Dev/supportingFiles:**/usr/src/node-red/keyfiles**

**--log-driver=splunk**

**--log-opt splunk-token=<splunk http event collector token>**

**--log-opt splunk-url=<splunk http event collector end-point>**

**--log-opt splunk-insecureskipverify=true.  <Use this option if SSL is enabled>**

**--log-opt tag="{{.Name}}/{{.FullID}}"**

 -p **1881:1881** --name <container name>  <repository>/image name

**Note**: ***Please ensure Port Number while building container is matching with corresponding port number in for each image as per the Docker file.***

# Deployment Tasks and Dependencies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr#** | **Task** | **Owner/ Point of contact** | **Date** | **Status** |
| 1 | Deploy Sprint-5 SOE Environment flow in SOE-1 / SOE-2 with https and https basic authentication enabled | Brian W/Jim/Nitin | 11/27 | Complete |
| 2 | Deploy Sprint-5 SOE Main flow in SOE-1 / SOE-2 with https and https basic authentication enabled | Brian W/Jim/Nitin | 11/27 | Complete |
| 3 | Deploy Sprint-5 CVP flow for SOE-1 & SOE-2 with https and https basic authentication enabled | Brian W | 11/27 | Complete |
| 4 | Setup Session Management SQL Tables | Brian W | 11/27 | Complete |
| 5 | Load Call Transfer Numbers | Brian W/Nitin | 11/27 | Complete |
| 5 | SOE Environment file updates in SOE-1 / SOE-2 | Brian W / Regions Security team | 11/27 | Complete |
| 6 | Configure vgw for SOE1/SOE2 (enable https and basic authentication) | Courtney B | 11/27 | Complete |
| 7 | Perform Functional Testing | Cat | 11/27 | Complete |
| 8 | Prepare Test server for SOE Flows | Brian W | 11/02 – 11/03 | Complete |
| 9 | Prepare SQL Server to support SOE Test instance | Brian W | 11/02 – 11/03 | Complete |
| 10 | Generate https certificates for SOE flows for UAT | Brian M | 11/30 | Complete |
| 11 | Generate https certificates for vgw | Courtney | 11/30 | Complete |
| 12 | Configure IVR to use https certificates | IVR team | 11/30 | Complete |
| 13 | Configure vgw for UAT (enable https and basic authentication) | Courtney B | 11/30 | Complete |
| 14 | Deploy SOE Environment flow image on Test server (with https and https basic authentication enabled) | Brian W | 11/30 | Complete |
| 15 | Deploy SOE Main flow image on UAT Server (with https and https basic authentication enabled) | Brian W | 11/30 | Complete |
| 16 | Deploy CVP flow image on UAT Server (with https and https basic authentication enabled) | Brian W | 11/30 | Complete |
| 17 | Configure SOE env file for UAT SOE | Brian W / Regions Security team | 11/30 | Complete |
| 18 | Update WCS workspace in UAT for changes | Cat | 11/30 | Complete |
| 19 | Perform shakedown testing | Brian W/Cat/Jim/Nitin | 11/30 | Complete |
| 20 | Released SOE UAT instance for Functional Testing | Brian W/Cat/Jim/Nitin | 11/30 | Complete |