Platform Event:

Platform Events exchange event data in real-time within the Salesforce platform, and between Salesforce and external platforms. Platform Events are a Salesforce-native feature that work like this:

1.Event producer: An event producer creates an event.

2.Event bus: The event gets added onto the event bus (aka channel), which operates as a queue, with a strict chronological order, and executes each event one after the other.

3.Event consumers: Event consumers subscribe to an event. The moment that event gets put onto the event bus, the event consumer will be notified.

Limitation:

Platform events cannot be queried through SOQL or SOSL.

Platform events cannot be used in reports, list views, and search.

Platform events do not have a related tab.

Distributed Platform events cannot be rolled back.

All platform event fields are read-only by default.

Fully not working:

import { LightningElement , track, wire, api} from 'lwc';

import {subscribe,unsubscribe,onError,setDebugFlag,isEmpEnabled,} from 'lightning/empApi';

import getStatus from '@salesforce/apex/ivrLeg.getStatus';

import {getRecord} from 'lightning/uiRecordApi';

import Voice\_Call from '@salesforce/schema/VoiceCall';

import VOICECALL\_IVR\_Leg from '@salesforce/schema/VoiceCall.CallDisposition';

const FIELDS= ['VoiceCall.CallDisposition','VoiceCall.IVR\_Leg\_Customer\_Sentiment\_\_c', 'VoiceCall.VendorCallKey'] ;

export default class LiveSentimentSP extends LightningElement {

//@wire(getStatus) wiredStatus;

//console.log(wiredStatus);

@api recordId; //current Account's Id

// fetching data from object

@wire(getRecord, { recordId:'$recordId',fields:FIELDS})

VoiceCallStatus({data,error}){

if(data){

console.log(data);

const call\_disposition = data.fields.CallDisposition.value;

if(call\_disposition == 'in-progress' || call\_disposition == 'new')

{

this.ILCS = data.fields.IVR\_Leg\_Customer\_Sentiment\_\_c.value;

}

}

else{

console.error(error)

}

}

channelName = '/event/Live\_Sentiment\_\_e';

ALCS = ' ' //Agent\_Leg\_Customer\_Sentiment\_\_c

ILCS= ' ' //IVR\_Leg\_Customer\_Sentiment\_\_c

showRes = false

@track

subscription = {};

renderedCallback(){

console.log('in render callback');

console.log(this.VoiceCallStatus.data);

}

//initializes the component

connectedCallback() {

console.log('in connected callback');

this.handleSubscribe();

// Register error listener

this.registerErrorLisener();

}

get ALCSMood(){

return this.ALCS

}

get ILCSMood(){

return this.ILCS

}

/\*get ILCSMood(){

return this.ILCS

}\*/

handleSubscribe(){

console.log('in handleSubscribe')

const messageCallback = (response)=> {

console.log('New Message received:', JSON.stringify(response));

this.showRes = true

console.log('Data used');

this.vendorCallKey = response['data']['payload']['VendorCallKey\_\_c'];

console.log(vendorCallKey);

if(vendorCallKey == this.data.fields.VoiceCall.VendorCallKey.value){

this.ALCS = response['data']['payload']['Agent\_Leg\_Customer\_Sentiment\_\_c']

}

};

//Invoke subscribe method of empApi. Passreference to messageCallback

subscribe(this.channelName, -1, messageCallback ).then((response) => {

//Response contains the subscription information on sub

console.log('Subscription request sent to:',

JSON.stringify(response.channel));

this.subscription = response;

});

}

registerErrorLisener(){

//Invoke onError empApi method

onError((error)=> {

console.log('Receive error from server:', JSON.stringify(error));

//Error contains the server-side error

});

}

}

Working Code:

Js Code:

import { LightningElement , track, wire, api} from 'lwc';

import { getRecord } from 'lightning/uiRecordApi';

import { subscribe } from 'lightning/empApi'; // for use of cometD

const FIELDS= ['VoiceCall.CallDisposition','VoiceCall.VendorCallKey','VoiceCall.IVR\_Leg\_Customer\_Sentiment\_\_c'] ;

export default class RealTimeSentiment extends LightningElement {

@track agentLegCustomerSentimentFromEvent; //Agent\_Leg\_Customer\_Sentiment\_\_c to be populated from Platform Event

@track agentLegEmoji; // track agent emoji

@track ivrLegCustomerSentimentFromVCR; //IVR\_Leg\_Customer\_Sentiment\_\_c from Voice Call Record

@track vendorCallKeyFromVCR; // VendorCallKey from Voice Call Record

@track statusFromVCR; // Status from Voice Call Record

@track ivrLegEmoji; // track customer emoji

@track subscription = {}; // track who is subscribing this method

@api channelName = '/event/Live\_Sentiment\_\_e';

@api recordId; //current VoiceCall Id

@wire(getRecord, { recordId:'$recordId', fields:FIELDS})

voiceCall({data,error}){

if(data){

console.log('BG data ', data);

this.ivrLegCustomerSentimentFromVCR = data.fields.IVR\_Leg\_Customer\_Sentiment\_\_c.value;

this.vendorCallKeyFromVCR = data.fields.VendorCallKey.value;

this.statusFromVCR = data.fields.CallDisposition.value;

if(this.ivrLegCustomerSentimentFromVCR != null){

if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'POSITIVE'){

this.ivrLegEmoji = '🙂';

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEGATIVE'){

this.ivrLegEmoji = '☹️';

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'MIXED'){

this.ivrLegEmoji = '😵‍💫';

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEUTRAL'){

this.ivrLegEmoji = '😐';

}else{

this.ivrLegEmoji ='';

}

}

else{

this.ivrLegEmoji ='';

this.ivrLegCustomerSentimentFromVCR='';

}

}

else{

console.error(error)

}

}

connectedCallback() {

console.log('BG in connected callback');

this.handleSubscribe();

}

handleSubscribe() {

const messageCallback = (response)=> {

console.log('BG New Message received:', JSON.stringify(response));

console.log('BG this.vendorCallKeyFromVCR: ', this.vendorCallKeyFromVCR);

if(this.vendorCallKeyFromVCR == response['data']['payload']['VendorCallKey\_\_c']) {

this.agentLegCustomerSentimentFromEvent = response['data']['payload']['Agent\_Leg\_Customer\_Sentiment\_\_c']

if(this.agentLegCustomerSentimentFromEvent != null){

if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'POSITIVE'){

this.agentLegEmoji = '🙂';

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEGATIVE'){

this.agentLegEmoji = '☹️';

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'MIXED'){

this.agentLegEmoji = '😵‍💫';

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEUTRAL'){

this.agentLegEmoji = '😐';

}else{

this.agentLegEmoji ='';

}

}

else{

this.agentLegEmoji ='';

this.agentLegCustomerSentimentFromEvent='';

}

}

};

subscribe(this.channelName, -1, messageCallback).then(response => {

console.log('BG Subscription request sent to: ', JSON.stringify(response.channel));

this.subscription = response;

});

}

}

Html code:

<template>

<lightning-card >

<div class="slds-page-header slds-page-header\_record-home slds-card\_\_header slds-grid">

<header class="slds-media slds-media\_center slds-has-flexi-truncate">

<div class="slds-media\_\_body">

<h2 class="slds-card\_\_header-title">

Customer Sentiment (Bijoyini Copy)

</h2>

</div>

</header>

</div>

<div class="slds-card\_\_body slds-card\_\_body\_inner">

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc">IVR Leg: <span class="slds-text-heading\_large">{ivrLegEmoji}</span> ({ivrLegCustomerSentimentFromVCR})</p>

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc">Agent Leg: <span class="slds-text-heading\_large">{agentLegEmoji}</span> ({agentLegCustomerSentimentFromEvent})</p>

</div>

</lightning-card>

</template>

Css Code:

.dummy{

color : black;

}

.slds-card\_\_header {

background-color: #f3f3f3;

}

Final Code

Html:

<template>

<lightning-card >

<div class="slds-page-header slds-page-header\_record-home slds-card\_\_header" style="height: 69px">

<header class="slds-media slds-media\_center slds-has-flexi-truncate">

<div class="slds-media\_\_body">

<h2 class="slds-card\_\_header-title slds-m-top\_x-small slds-m-left\_x-small">

Customer Sentiment

</h2>

</div>

</header>

</div>

<div class="slds-card\_\_body slds-card\_\_body\_inner">

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc"><strong>IVR Leg: </strong> <span class="slds-text-heading\_large">{ivrLegEmoji}</span> ({ivrLegCustomerSentimentFromVCR})</p>

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc"><strong>Agent Leg: </strong> <span class="slds-text-heading\_large">{agentLegEmoji}</span> ({agentLegCustomerSentimentFromEvent})</p>

</div>

</lightning-card>

</template>

CSS:

.dummy{

color : black;

}

.slds-card\_\_header {

background-color: #f3f3f3;

top: -25px;

margin-top: -30px;

border-style: outset;

font-weight: bold;

}

.slds-card\_\_body{

position: relative;

top: -8px;

}

.slds-p-horizontal\_large{

margin-left: -20px;

}

.slds-page-header{

position: relative;

top: 5px;

}

.slds-card\_\_body{

position: relative;

top: -1px;

}

Js:

import { LightningElement , track, wire, api} from 'lwc';

import { getRecord } from 'lightning/uiRecordApi';

import { subscribe } from 'lightning/empApi';

const FIELDS= ['VoiceCall.CallDisposition','VoiceCall.VendorCallKey','VoiceCall.IVR\_Leg\_Customer\_Sentiment\_\_c'] ;

export default class RealTimeSentiment extends LightningElement {

@track agentLegCustomerSentimentFromEvent; //Agent\_Leg\_Customer\_Sentiment\_\_c to be populated from Platform Event

@track agentLegEmoji;

@track ivrLegCustomerSentimentFromVCR; //IVR\_Leg\_Customer\_Sentiment\_\_c from Voice Call Record

@track vendorCallKeyFromVCR; // VendorCallKey from Voice Call Record

@track statusFromVCR; // Status from Voice Call Record

@track ivrLegEmoji;

@track subscription = {};

@api channelName = '/event/Live\_Sentiment\_\_e';

@api recordId; //current VoiceCall Id

@wire(getRecord, { recordId:'$recordId', fields:FIELDS})

voiceCall({data,error}){

if(data){

console.log('BG data ', data);

this.ivrLegCustomerSentimentFromVCR = data.fields.IVR\_Leg\_Customer\_Sentiment\_\_c.value;

this.vendorCallKeyFromVCR = data.fields.VendorCallKey.value;

this.statusFromVCR = data.fields.CallDisposition.value;

if(this.ivrLegCustomerSentimentFromVCR != null){

if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'POSITIVE'){

this.ivrLegEmoji = '🙂';

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEGATIVE'){

this.ivrLegEmoji = '☹️';

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'MIXED'){

this.ivrLegEmoji = '😵‍💫';

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEUTRAL'){

this.ivrLegEmoji = '😐';

}else{

this.ivrLegEmoji ='';

}

}

else{

this.ivrLegEmoji ='';

this.ivrLegCustomerSentimentFromVCR='';

}

}

else{

console.error(error)

}

}

connectedCallback() {

console.log('BG in connected callback');

this.handleSubscribe();

}

handleSubscribe() {

const messageCallback = (response)=> {

console.log('BG New Message received:', JSON.stringify(response));

console.log('BG this.vendorCallKeyFromVCR: ', this.vendorCallKeyFromVCR);

if(this.vendorCallKeyFromVCR == response['data']['payload']['VendorCallKey\_\_c']) {

this.agentLegCustomerSentimentFromEvent = response['data']['payload']['Agent\_Leg\_Customer\_Sentiment\_\_c']

if(this.agentLegCustomerSentimentFromEvent != null){

if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'POSITIVE'){

this.agentLegEmoji = '🙂';

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEGATIVE'){

this.agentLegEmoji = '☹️';

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'MIXED'){

this.agentLegEmoji = '😵‍💫';

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEUTRAL'){

this.agentLegEmoji = '😐';

}else{

this.agentLegEmoji ='';

}

}

else{

this.agentLegEmoji ='';

this.agentLegCustomerSentimentFromEvent='';

}

}

};

subscribe(this.channelName, -1, messageCallback).then(response => {

console.log('BG Subscription request sent to: ', JSON.stringify(response.channel));

this.subscription = response;

});

}

}

Final Code for sentiment.(where we remove hard Coded emoji with Picture)

Html:

<template>

<lightning-card >

<div class="slds-page-header slds-page-header\_record-home slds-card\_\_header" style="height: 69px">

<header class="slds-media slds-media\_center slds-has-flexi-truncate">

<div class="slds-media\_\_body">

<h2 class="slds-card\_\_header-title slds-m-top\_x-small slds-m-left\_x-small">

Customer Sentiment

</h2>

</div>

</header>

</div>

<div class="slds-card\_\_body slds-card\_\_body\_inner">

<template if:true ={handleChangeIVR}>

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc"><strong>IVR Leg &nbsp;&nbsp; :&thinsp;&thinsp;</strong> <span class="slds-text-heading\_medium"><img src={ivrLegEmoji} height="25px" width="25px"/></span> ({ivrLegCustomerSentimentFromVCR})</p>

</template>

<template if:false ={handleChangeIVR}>

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc"><strong>IVR Leg &nbsp;&nbsp;&thinsp; :&thinsp;&thinsp;</strong> <span class="slds-text-heading\_medium"></span>({ivrLegCustomerSentimentFromVCR})</p>

</template>

</div>

<div class="slds-card\_\_body slds-card\_\_body\_inner">

<template if:true ={agentLegData}>

<template if:true ={handleChangeAgent}>

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc"><strong>Agent Leg: </strong> <span class="slds-text-heading\_medium"><img src={agentLegEmoji} height="25px" width="25px"/></span> ({agentLegCustomerSentimentFromEvent})</p>

</template>

<template if:false ={handleChangeAgent}>

<p class="slds-p-horizontal\_large slds-p-left\_medium changepc"><strong>Agent Leg: </strong> <span class="slds-text-heading\_medium"></span>&thinsp;({agentLegCustomerSentimentFromEvent})</p>

</template>

</template>

<template if:false ={agentLegData}>

<p><br><br></p>

</template>

</div>

</lightning-card>

</template>

Js

import { LightningElement , track, wire, api} from 'lwc';

import { getRecord } from 'lightning/uiRecordApi';

import { subscribe } from 'lightning/empApi';

import positiveEmoji from '@salesforce/resourceUrl/PositiveFace';

import neutralEmoji from '@salesforce/resourceUrl/NeutralFace';

import mixedEmoji from '@salesforce/resourceUrl/MixedFace';

import negativeEmoji from '@salesforce/resourceUrl/NegativeFace';

import noDataEmoji from '@salesforce/resourceUrl/NoDataFace'

const FIELDS= ['VoiceCall.CallDisposition','VoiceCall.VendorCallKey','VoiceCall.IVR\_Leg\_Customer\_Sentiment\_\_c'] ;

export default class RealTimeSentiment extends LightningElement {

@track agentLegCustomerSentimentFromEvent = 'No Data Yet'; //Agent\_Leg\_Customer\_Sentiment\_\_c to be populated from Platform Event

@track ivrLegCustomerSentimentFromVCR = 'No Data'; //IVR\_Leg\_Customer\_Sentiment\_\_c from Voice Call Record

@track agentLegEmoji = noDataEmoji ; // Agent Emoji

@track ivrLegEmoji =noDataEmoji ; // Customer Emoji

@track vendorCallKeyFromVCR; // VendorCallKey from Voice Call Record

@track statusFromVCR; // Status from Voice Call Record

@track handleChangeIVR = true; // Conditional Rendaring for Emoji used in HTML

@track handleChangeAgent =true; // Conditional Rendaring for Emoji used in HTML

@track agentLegData = false; // Hinding Agent Data

@track subscription = {};

@api channelName = '/event/Live\_Sentiment\_\_e';

@api recordId; //current VoiceCall Id

@wire(getRecord, { recordId:'$recordId', fields:FIELDS})

voiceCall({data,error}){

if(data){

console.log('BG data ', data);

this.ivrLegCustomerSentimentFromVCR = data.fields.IVR\_Leg\_Customer\_Sentiment\_\_c.value;

this.vendorCallKeyFromVCR = data.fields.VendorCallKey.value;

this.statusFromVCR = data.fields.CallDisposition.value;

if(this.statusFromVCR !== 'completed' || this.statusFromVCR ==='in-progress'){ // Hiding Condition for Agent

this.agentLegData =true;

this.handleChangeAgent =true;

this.agentLegEmoji =noDataEmoji;

this.agentLegCustomerSentimentFromEvent ='No Data Yet';

}

if(this.ivrLegCustomerSentimentFromVCR !== null ){

if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'POSITIVE'){

this.ivrLegEmoji = positiveEmoji;

this.handleChangeIVR = true;

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEGATIVE'){

this.ivrLegEmoji = negativeEmoji;

this.handleChangeIVR = true;

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'MIXED'){

this.ivrLegEmoji = mixedEmoji;

this.handleChangeIVR = true;

}else if(this.ivrLegCustomerSentimentFromVCR.toUpperCase() === 'NEUTRAL'){

this.ivrLegEmoji = neutralEmoji;

this.handleChangeIVR = true;

}else{

this.ivrLegEmoji ='';

this.handleChangeIVR = false;

}

}

else{

this.ivrLegEmoji = noDataEmoji;

this.ivrLegCustomerSentimentFromVCR='No Data';

this.handleChangeIVR = true;

}

}

else{

console.error(error)

}

}

connectedCallback() {

console.log('BG in connected callback');

this.handleSubscribe();

}

handleSubscribe() {

const messageCallback = (response)=> {

console.log('BG New Message received:', JSON.stringify(response));

console.log('BG this.vendorCallKeyFromVCR: ', this.vendorCallKeyFromVCR);

if(this.statusFromVCR !== 'completed'){

this.agentLegData =true;

this.handleChangeAgent =true;

if(this.vendorCallKeyFromVCR == response['data']['payload']['VendorCallKey\_\_c'] && this.statusFromVCR !== 'completed') {

this.agentLegCustomerSentimentFromEvent = response['data']['payload']['Agent\_Leg\_Customer\_Sentiment\_\_c']

if(this.agentLegCustomerSentimentFromEvent !== null){

if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'POSITIVE'){

this.agentLegEmoji = positiveEmoji;

this.handleChangeAgent =true;

this.agentLegData =true;

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEGATIVE'){

this.agentLegEmoji = negativeEmoji;

this.handleChangeAgent =true;

this.agentLegData =true;

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'MIXED'){

this.agentLegEmoji = mixedEmoji;

this.handleChangeAgent =true;

this.agentLegData =true;

}else if(this.agentLegCustomerSentimentFromEvent.toUpperCase() === 'NEUTRAL'){

this.agentLegEmoji = neutralEmoji;

this.handleChangeAgent =true;

this.agentLegData =true;

}else{

this.agentLegEmoji ='';

this.handleChangeAgent =false;

this.agentLegData =true;

}

}

else{

this.handleChangeAgent =true;

this.agentLegData =true;

this.agentLegEmoji =noDataEmoji;

this.agentLegCustomerSentimentFromEvent ='No Data Yet';

}

}

}

else{

this.agentLegData = false;

}

};

subscribe(this.channelName, -1, messageCallback).then(response => {

console.log('BG Subscription request sent to: ', JSON.stringify(response.channel));

this.subscription = response;

});

}

}

Css

.dummy{

color : black;

}

.slds-card\_\_header {

background-color: #f3f3f3;

top: -25px;

margin-top: -30px;

border-style: outset;

font-weight: bold;

}

.slds-card\_\_body{

position: relative;

top: -8px;

}

.slds-p-horizontal\_large{

margin-left: -20px;

}

.slds-page-header{

position: relative;

top: 5px;

}

.slds-card\_\_body{

position: relative;

top: -1px;

}