Vonage Opentok mendix react pluggable widget (code)

1. Εισαγωγή

Το σύστημα Vonage Video Call είναι μία React-based εφαρμογή που αξιοποιεί το Vonage (OpenTok) API για τη διαχείριση video call sessions. Είναι πλήρως ενσωματωμένο με Mendix workflows, προσφέροντας διαχείριση video streams και παρακολούθηση καταστάσεων sessions σε πραγματικό χρόνο.

Κύριοι Στόχοι

1. **Διαχείριση Video Calls**:
   - Διαχείριση sessions, publishing video, και διαχείριση streams.
2. **Ενσωμάτωση με Mendix**:
   - Ενεργοποίηση workflows για offline καταστάσεις, διακοπές, κ.λπ.
3. **Καταγραφή Συμβάντων**:
   - Παρακολούθηση συμμετεχόντων, συνδέσεων, και αποσυνδέσεων.
4. **Custom Alerts**:
   - Δυναμικά UI notifications για καταστάσεις sessions.

2. Αρχιτεκτονική Συστήματος

Το σύστημα είναι modular, χωρισμένο σε **τέσσερα components**:

MODULE	ΡΟΛΟΣ	ΕΞΑΡΤΗΣΕΙΣ
VONAGE VIDEO CALL	Διαχειρίζεται τον κύκλο ζωής των video sessions.	Mendix Helpers, Event Listeners, Connection Manager
MENDIX HELPERS	Χειρίζεται Mendix workflows και αντικείμενα.	Mendix `mx` APIs
EVENT LISTENERS	Παρακολουθεί browser-level και session-level events.	Vonage Session, Mendix Helpers
CONNECTION MANAGER	Καταγράφει συνδέσεις/subscriptions και διαχειρίζεται alerts.	DOM Elements, Vonage Session

---

3. Επισκόπηση Κώδικα

3.1 Vonage Video Call Component (`VonageVideoCall.jsx`)

Κεντρικό component που:

- Ξεκινά, διαχειρίζεται, και καθαρίζει Vonage sessions.
- Ενσωματώνεται με Mendix workflows για custom events.

3.2 Mendix Helper Utilities (`mendixHelper.jsx`)

Προσφέρει utility functions για:

1. Ενεργοποίηση Mendix workflows.
2. Τροποποίηση attributes αντικειμένων στο Mendix.

3.3 Event Listeners (`eventListeners.jsx`)

Διαχειρίζεται browser-level και session-level events:

- Παρακολουθεί visibility changes, offline/online status. Ενημερώνει το UI
- και τα Mendix workflows ανάλογα.

3.4 Connection Manager (`connectionInfo.jsx`) Λογισμικό για:

1. Καταγραφή πληροφοριών συνδέσεων.
2. Custom alerts για δυναμικές ειδοποιήσεις κατά τη διάρκεια sessions.

---

4. Αναλυση Κώδικα (VonageVideoCall.jsx)

4.1. `logMessage`

```
const logMessage = (level, message, data = null) => {
    const logPrefix = "VRL_Main:";

    if (level === "error") console.error(`${logPrefix} ${message}`, data);
    else if (level === "warn") console.warn(`${logPrefix} ${message}`, data);
    else console.log(`${logPrefix} ${message}`, data);

};
```

Σκοπός:

Παρέχει μία κεντρική μέθοδο καταγραφής μηνυμάτων (logging) με χρήση prefix (`VRL_Main`).
Υποστηρίζει τρία επίπεδα logging:

- `info` (γενικές πληροφορίες),
- `warn` (προειδοποιήσεις),
- `error` (σφάλματα).

Λειτουργία:

1. Δέχεται τα παρακάτω ορίσματα:
    - `level`: Το επίπεδο καταγραφής (`info`, `warn`, `error`).
    - `message`: Το μήνυμα προς καταγραφή.
    - `data` (προαιρετικό): Πρόσθετα δεδομένα για το μήνυμα.
2. Χρησιμοποιεί το κατάλληλο method του `console` για την καταγραφή (π.χ., `console.log`).

Εξαρτήσεις:

Καμία. Αυτή η συνάρτηση είναι ανεξάρτητη.

4.2. `flowInterrupt`

```
const flowInterrupt = () => MF_Call(terminateFlow, guid, mxformOrigin);
```

Σκοπός:

Ενεργοποιεί το `terminateFlow` workflow του Mendix όταν το video session διακόπτεται.

Λειτουργία:

- Χρησιμοποιεί τη συνάρτηση `MF_Call` για να καλέσει το Mendix workflow, περνώντας τα εξής:
    - Το όνομα του workflow (`terminateFlow`),
    - Το session GUID (`guid`),
    - Το reference της Mendix φόρμας (`mxformOrigin`).

Εξαρτήσεις:

- `MF_Call`: Συνάρτηση που εκτελεί Mendix workflows.

4.3. `offlineFlow`

```
javascript


  const offlineFlow = () => MF_Call(workFlowOffline, guid, mxformOrigin);
```

Σκοπός:

Ενεργοποιεί το `workFlowOffline` workflow του Mendix όταν ο χρήστης βρεθεί offline.

Λειτουργία:

- Λειτουργεί παρόμοια με τη `flowInterrupt`, αλλά καλεί το workflow `workFlowOffline`.

Εξαρτήσεις:

- `MF_Call`.

4.4. `offline` και `online`

```
javascript


  const offline = () => showCustomAlert("Reconnecting", "Reconnecting to the video session. Pl
  ease wait.");
  const online = () => closeCustomAlert();
```

Σκοπός:

- `offline`: Εμφανίζει ένα custom alert όταν ο χρήστης είναι offline ή προσπαθεί να επανασυνδεθεί.
- `online`: Κλείνει το alert όταν ο χρήστης επανέλθει online.

Εξαρτήσεις:

- `showCustomAlert`: Εμφανίζει alerts.
- `closeCustomAlert`: Κλείνει τα alerts.

4.5. `initializeSession`

```javascript
const initializeSession = () => {
    if (typeof OT === "undefined") {
        logMessage("error", "OpenTok library is not available.");
        showCustomAlert("Error", "Video call features are currently unavailable.");
        return;
    }

    vonageSession = OT.initSession(vonageApiKeyValue, vonageSessionIDValue);
    logMessage("info", "Vonage session initialized.");

    setupSessionEvents();
    connectSession();
};
```

Σκοπός:

Αρχικοποιεί το Vonage session χρησιμοποιώντας τα παρεχόμενα API key και session ID.

Λειτουργία:

1. Ελέγχει αν το OpenTok library (`OT`) είναι διαθέσιμο.
2. Χρησιμοποιεί το `OT.initSession` για να δημιουργήσει ένα νέο session.
3. Καλεί τις συναρτήσεις `setupSessionEvents` και `connectSession` για να ρυθμίσει τα events και να συνδεθεί.

Εξαρτήσεις:

- `OT` (OpenTok SDK),
- `logMessage`, `showCustomAlert`, `setupSessionEvents`, `connectSession`.

4.6. `setupSessionEvents`

```
javascript



  const setupSessionEvents = () => {
      if (!vonageSession) return;


      const eventHandlers = {

          connectionCreated: () => logMessage("info", "Connection created."),
          connectionDestroyed: () => logMessage("info", "Connection destroyed."),
          streamCreated: handleStreamCreated,

          streamDestroyed: handleStreamDestroyed,
          sessionDisconnected: handleSessionDisconnected,

      };


      Object.entries(eventHandlers).forEach(([event, handler]) => {
          vonageSession.on(event, handler);

      });
  };
```

Σκοπός:

Δεσμεύει event listeners για τα βασικά events του Vonage session.

Λειτουργία:

1. Ελέγχει αν υπάρχει ενεργό session.
2. Ορίζει handlers για events όπως:
    - **`connectionCreated`**, **`streamCreated`**, **`sessionDisconnected`**.
3. Χρησιμοποιεί το **`vonageSession.on`** για να δεσμεύσει τους handlers.

Εξαρτήσεις:

- **`logMessage`**, **`handleStreamCreated`**, **`handleStreamDestroyed`**, **`handleSessionDisconnected`**.

4.7. `connectSession`

```
javascript



  const connectSession = () => {
      vonageSession.connect(vonageTokenValue, (error) => {

          if (error) {

              logMessage("error", "Error connecting to the session.", error);
          } else {

              logMessage("info", "Session connected successfully.");
              notifyConnections(vonageSession);
              startVideoPublishing();

          }
      });

  };
```

Σκοπός:

Συνδέει το session χρησιμοποιώντας το token.

Λειτουργία:

1. Χρησιμοποιεί το `vonageSession.connect` με το token.
2. Σε περίπτωση επιτυχίας:
    - Καταγράφει το μήνυμα επιτυχίας.
    - Καλεί τη `notifyConnections` για να καταγράψει τους συμμετέχοντες.
    - Ξεκινά το video publishing με τη `startVideoPublishing`.

Εξαρτήσεις:

- `vonageSession`, `logMessage`, `notifyConnections`, `startVideoPublishing`.

4.8. `startVideoPublishing`

```
javascript



  const startVideoPublishing = () => {
      const videoPublisherOptions = {

          insertMode: "append",

          width: "100%",

          height: "100%",
          publishAudio: true,
          publishVideo: true,

      };


      if (themeSelection === "blur") {

          videoPublisherOptions.videoFilter = { type: "backgroundBlur", blurStrength: "high"
  };

      } else if (themeSelection === "theme") {
          videoPublisherOptions.videoFilter = {

              type: "backgroundReplacement",
              backgroundImgUrl: "../images/theme.jpg"

          };
      }


      videoPublisher = OT.initPublisher("publisher-video", videoPublisherOptions, (error) => {
          if (error) logMessage("error", "Error initializing publisher.", error);

          else {

              logMessage("info", "Video publisher initialized successfully.");
              vonageSession.publish(videoPublisher);

          }
      });

  };
```

Σκοπός:

Αρχικοποιεί τον video publisher και δημοσιεύει το stream.

Λειτουργία:

1. Καθορίζει τις επιλογές για τον publisher (διαστάσεις, video/audio settings).
2. Εφαρμόζει themes εάν οριστούν (`blur`, `backgroundReplacement`).
3. Χρησιμοποιεί το `OT.initPublisher` για να δημιουργήσει τον publisher και δημοσιεύει το stream.

Εξαρτήσεις:

- `OT` (OpenTok SDK), `logMessage`, `vonageSession`.

4.9. Stream Events

`handleStreamCreated`

```
javascript


  const handleStreamCreated = (event) => {

      logMessage("info", `Stream created: ${event.stream.streamId}`);
      vonageSession.subscribe(event.stream, "subscriber-video", { insertMode: "append", ...

  });
  };
```

`handleStreamDestroyed`

```
javascript


  const handleStreamDestroyed = (event) => {

  logMessage("info", `Stream destroyed: ${event.stream.streamId}`);};
```

Σκοπός:

- `handleStreamCreated`: Εγγράφεται στα νέα streams.
- `handleStreamDestroyed`: Καταγράφει την καταστροφή ενός stream.

4.1.0. `cleanupResources`

```
javascript



  const cleanupResources = () => {

      if (videoPublisher) videoPublisher.destroy();
      if (vonageSession) vonageSession.disconnect();

  };
```

Σκοπός:

Καθαρίζει τα resources του session και του publisher.

5. Ανάλυση Κώδικα: (mendixHelper.jsx)

Utilities for Managing Mendix Microflows, Video Calls, and Offline Notifications

Επισκόπηση

Αυτό το module παρέχει δύο κύριες utility συναρτήσεις για την **επικοινωνία με το Mendix**:

1. `MF_Call`: Εκτέλεση Mendix microflows.
2. `updateMendixObject`: Ενημέρωση αντικειμένων στο Mendix.

Οι συναρτήσεις αυτές διευκολύνουν τη διαχείριση workflows και αντικειμένων στο Mendix, επιτρέποντας την εκτέλεση ενεργειών από React components.

5.1. Συνάρτηση `MF_Call`

```javascript


export const MF_Call = (microflowActionName, sessionEntityGUID, mxformOrigin) => {
    if (!microflowActionName || !sessionEntityGUID) {

        console.warn("VRL_(MX_Lib): Microflow action or session entity GUID is missing.");
        return;

    }


    mx.data.action({
        params: {

            applyto: "selection",
            actionname: microflowActionName,
            guids: [sessionEntityGUID]

        },

        origin: mxformOrigin,

        callback: (response) => console.log("VRL_(MX_Lib): Microflow executed successfull
y:", response),

        error: (error) => console.error("VRL_(MX_Lib): Error executing microflow:", JSON.str
ingify(error))

    });
};
```

Σκοπός

Η `MF_Call` εκτελεί ένα συγκεκριμένο **Mendix microflow** που εφαρμόζεται σε ένα αντικείμενο με βάση το GUID του.

Λειτουργία

1. **Validation**:
   - Ελέγχει αν έχουν δοθεί τα `**microflowActionName**` και `**sessionEntityGUID**`.
   - Αν λείπουν, εμφανίζεται προειδοποιητικό μήνυμα και η συνάρτηση τερματίζεται.
2. **Εκτέλεση Microflow**:
   - Καλεί το Mendix `**mx.data.action**` API.
   - Ορίζει τις παραμέτρους:
     - `**applyto**`: Ορίζει ότι η ενέργεια εφαρμόζεται σε ένα αντικείμενο (selection).
     - `**actionname**`: Το όνομα του microflow.
     - `**guids**`: Λίστα GUIDs των αντικειμένων στα οποία εφαρμόζεται το microflow.
   - Προσθέτει το `**origin**` για να δώσει το context της φόρμας Mendix.
3. **Callbacks**:
   - `**callback**`: Εκτελείται όταν το microflow ολοκληρωθεί επιτυχώς. Εμφανίζει μήνυμα επιτυχίας στο console.
   - `**error**`: Εκτελείται όταν υπάρχει σφάλμα. Εμφανίζει λεπτομέρειες του σφάλματος στο console.

Χρήση

```javascript
MF_Call("TerminateSession", "12345-67890-GUID", mxform);
```

- Εκτελεί το microflow `**TerminateSession**` στο αντικείμενο με GUID `**12345-67890-GUID**`.

Εξαρτήσεις

- `**mx.data.action**`: Mendix API για την εκτέλεση microflows.

5.2. Συνάρτηση `updateMendixObject`

```javascript


export const updateMendixObject = (guid, attributeName, attributeValue, onSuccess, onError)
=> {

    if (!guid || !attributeName || attributeValue === undefined) {
        console.error("VRL_(MX_Lib): updateMendixObject - Invalid parameters.");
        if (onError) onError("Invalid parameters");

        return;
    }


     mx.data.get({
         guid,

        callback: (retrievedObject) => {
            if (retrievedObject) {

                console.log("VRL_(MX_Lib): Object retrieved successfully:", retrievedObjec
t);


                // Set the attribute and its value
                retrievedObject.set(attributeName, attributeValue);

                console.log(`VRL_(MX_Lib): Updated attribute ${attributeName} with value:`,
attributeValue);


                // Commit the changes
                mx.data.commit({

                    mxobj: retrievedObject,
                    callback: () => {

                        console.log("VRL_(MX_Lib): Object committed successfully!");
                        if (onSuccess) onSuccess();

                    },

                    error: (commitError) => {

                        console.error("VRL_(MX_Lib): Error committing object:", commitErro
r);


                        if (onError) onError(commitError);
                    }

                });
            } else {

                console.warn("VRL_(MX_Lib): No object found with the specified GUID.");
                if (onError) onError("Object not found");

            }
        },

        error: (fetchError) => {

            console.error("VRL_(MX_Lib): Error retrieving object by GUID:", fetchError);
            if (onError) onError(fetchError);

        }
    });

};
```

Σκοπός

Η `updateMendixObject` επιτρέπει την **ενημέρωση ενός attribute** ενός Mendix object, καθώς και την αποθήκευση (commit) της αλλαγής.

Λειτουργία

1. Validation

- Ελέγχει αν έχουν δοθεί έγκυρες τιμές για:
  - `guid`: Το μοναδικό ID του αντικειμένου Mendix.
  - `attributeName`: Το όνομα του attribute.
  - `attributeValue`: Η νέα τιμή του attribute.
- Αν κάποια από τις παραμέτρους λείπει, εμφανίζει μήνυμα σφάλματος και καλεί το `onError`.

2. Ανάκτηση Αντικειμένου

- Καλεί τη συνάρτηση `mx.data.get` για να ανακτήσει το αντικείμενο με βάση το GUID.
- Αν το αντικείμενο βρεθεί:
  - Εμφανίζει μήνυμα επιτυχίας στο console.
  - Προχωρά στο επόμενο βήμα.

3. Ενημέρωση Attribute

- Με τη μέθοδο `set`, ενημερώνει το `attributeName` με την τιμή `attributeValue`.
- Εμφανίζει μήνυμα στο console για την αλλαγή.

4. Commit Αλλαγών

- Χρησιμοποιεί το `mx.data.commit` για να αποθηκεύσει την αλλαγή στη βάση δεδομένων.
- **Callbacks**:
  - `callback`: Εμφανίζει μήνυμα επιτυχίας και καλεί το `onSuccess`.
  - `error`: Εμφανίζει μήνυμα αποτυχίας και καλεί το `onError`.

5. Error Handling

- Αν το αντικείμενο δεν βρεθεί ή η ανάκτηση αποτύχει:
  - Εμφανίζει μήνυμα σφάλματος.
  - Καλεί το `onError`.

Χρήση

```
javascript



  updateMendixObject(
      "12345-67890-GUID",
      "status",
      "offline",

      () => console.log("Status updated successfully!"),
      (error) => console.error("Error updating status:", error)

  );
```

- Ενημερώνει το attribute `**status**` του αντικειμένου με GUID `**12345-67890-GUID**` στη νέα τιμή `**offline**`.

Εξαρτήσεις

- `**mx.data.get**`: Mendix API για την ανάκτηση αντικειμένων.
- `**mx.data.commit**`: Mendix API για την αποθήκευση αλλαγών.

Συνολική Ανάλυση

Συνάρτηση	Σκοπός	Κύρια Σημεία
`**MF_Call**`	Εκτελεί Mendix microflows.	Validation, εκτέλεση μέσω `**mx.data.action**`, callbacks για επιτυχία/σφάλμα.
`**updateMendixObject**`	Ενημερώνει attributes αντικειμένων Mendix και αποθηκεύει τις αλλαγές.	Validation, ανάκτηση αντικειμένου, ενημέρωση attribute, commit, error handling.

6. Ανάλυση Κώδικα: (eventListeners.jsx)

Event Listeners για Vonage Sessions

Επισκόπηση

Αυτό το module περιέχει:

1. Διαχείριση συνδρομητών (`activeSubscribers`).
2. Έλεγχος συνδέσεων (`getConnections`, `checkParticipantOnlineStatus`).
3. Ρύθμιση event listeners για:
    - Καταστάσεις offline/online.
    - Χειρισμό visibility changes.
    - Event handlers για το Vonage session.
4. Καθαρισμός listeners (cleanup).

Ο κώδικας είναι δομημένος ώστε να χειρίζεται τη ροή εργασιών και να ενημερώνει τη Mendix βάση δεδομένων, ενώ παρέχει feedback μέσω του UI.

6.1. Global Variables

`activeSubscribers`

```
javascript


let activeSubscribers = new Set();
```

- **Σκοπός**: Διατηρεί τις active συνδέσεις (subscribers).
- Χρησιμοποιεί `Set` για αποφυγή διπλοτύπων.

`listenersInitialized`

```
javascript


let listenersInitialized = false;
```

- **Σκοπός**: Αποτρέπει πολλαπλή αρχικοποίηση listeners.

6.2. Συνάρτηση `getConnections`

```javascript
const getConnections = (vonageSession) => {
    if (!vonageSession || !vonageSession.connections) { console.log("VRL_(Listeners): Vonage session is not properly initialized."); return [];
    }

    try {
        if (Array.isArray(vonageSession.connections)) {
            return vonageSession.connections;
        } else if (vonageSession.connections instanceof Map) {
            return Array.from(vonageSession.connections.values());
        } else if (typeof vonageSession.connections === "object") {
            return Object.values(vonageSession.connections);
        } else {
            console.log("VRL_(Listeners): Unknown connections format.");
            return [];
        }
    } catch (error) {
        console.error("VRL_(Listeners): Error normalizing connections:", error);
        return [];
    }
};
```

Σκοπός

Κανονικοποιεί το format των `connections` ενός Vonage session.

Λειτουργία

1. Ελέγχει αν το `vonageSession` και οι `connections` είναι έγκυρες.
2. Επιστρέφει τις `connections` με βάση το format:
    - `Array`: Επιστρέφεται αυτούσιο.
    - `Map`: Μετατρέπεται σε array μέσω `Array.from`.
    - `Object`: Μετατρέπεται σε array μέσω `Object.values`.

6.3. Συνάρτηση `checkParticipantOnlineStatus`

```
javascript



  const checkParticipantOnlineStatus = async (vonageSession) => {
      try {

          const connections = getConnections(vonageSession);
          if (connections.length === 0) {

              console.log("VRL_(Listeners): No active participants found.");
              activeSubscribers.clear();

              return false;
          }


          const nonPublisherConnections = connections.filter(

              (connection) => connection.role && connection.role !== "publisher"
          );


          if (nonPublisherConnections.length > 0) {

              console.log(`VRL_(Listeners): Active non-publisher participants found (${nonPubl
  isherConnections.length}).`);

              activeSubscribers = new Set(nonPublisherConnections.map((conn) => conn.connectio
  nId));


              nonPublisherConnections.forEach((connection) => {
                  console.log("VRL_(Listeners): Participant ID:", connection.connectionId);

              });


              return true;
          } else {

              console.log("VRL_(Listeners): No active non-publisher participants found.");
              activeSubscribers.clear();

              return false;
          }

      } catch (error) {

          console.error("VRL_(Listeners): Error checking participant status:", error);
          activeSubscribers.clear();

          return false;
      }

  };
```

Σκοπός

Ελέγχει αν υπάρχουν ενεργοί συμμετέχοντες (μη publishers) στο session.

Λειτουργία

1. Ανακτά τις `connections` μέσω της `getConnections`.
2. Ελέγχει:
    - Αν υπάρχουν συμμετέχοντες (`connections.length > 0`).
    - Αν οι συμμετέχοντες **δεν είναι publishers**.
3. Ενημερώνει το `activeSubscribers` με τα `connectionId` των μη publishers.

6.4. Συνάρτηση `setupEventListeners`

```
javascript



 export const setupEventListeners = ({
     enableInterruption,
     flowInterrupt,

     vonageSession,
     notifyDisconnection,
     updateMendixObject,
     sessionGuid,
     offlineAttributeName,
     offline,




     online,
 }) => { ... };
```

Σκοπός

Αρχικοποιεί event listeners για:

1. **Browser Events**:
    - Offline/Online αλλαγές.
    - `visibilitychange` για διακοπές ροής (flow interruptions).
2. **Vonage Session Events**:

`connectionCreated`, `connectionDestroyed`, κ.λπ.

Netcompany

Βασικά Events

1. `handleOffline`

```
javascript



  const handleOffline = () => {

      console.log("VRL_(Listeners): Network connection lost. User is offline.");
      notifyDisconnection(vonageSession);

      offline();



      updateMendixObject(
          sessionGuid,
          offlineAttributeName,
          true,

          () => {

              console.log(`VRL_(Listeners): Offline status updated successfully for GUID: ${se
  ssionGuid}`);

          },

          (error) => {

              console.error(`VRL_(Listeners): Failed to update offline status for GUID: ${sess
  ionGuid}`, error);

          }
      );


      document.body.classList.add("offline");

      const statusMessageElement = document.getElementById("status-message");
      if (statusMessageElement) {

          statusMessageElement.textContent = "You are currently offline.";
      }

  };
```

Σκοπός:

- Ενημερώνει τη Mendix βάση και το UI όταν ο χρήστης βρεθεί offline.

2. `handleOnline`

```
javascript


  const handleOnline = async () => {
      console.log("VRL_(Listeners): Network connection restored. Checking participant status...");
      document.body.classList.remove("offline");

      const statusMessageElement = document.getElementById("status-message");
      if (statusMessageElement) {
          statusMessageElement.textContent = "You are back online!";
      }

      flowInterrupt();
  };
```

Σκοπός:

- Αφαιρεί το offline state από το UI και ελέγχει την κατάσταση των συμμετεχόντων.

3. `handleVisibilityChange`

```
javascript


  const handleVisibilityChange = () => {
      if (document.hidden) {
          flowInterrupt();
          console.log("VRL_(Listeners): Visibility change detected. Flow interrupted.");
      }
  };
```

Σκοπός:

- Διακόπτει τη ροή όταν η σελίδα γίνει αόρατη (π.χ., αλλαγή tab).

Netcompany

Vonage Session Handlers

1. `connectionCreated`

```
javascript


  connectionCreated: (event) => {
      console.log(`VRL_(Listeners): Connection created: ${event.connection.connectionId}`);
      activeSubscribers.add(event.connection.connectionId);
  }
```

- Καταγράφει νέες συνδέσεις.

2. `connectionDestroyed`

```
javascript


  connectionDestroyed: (event) => {
      console.log(`VRL_(Listeners): Connection destroyed: ${event.connection.connectionId}`);
      activeSubscribers.delete(event.connection.connectionId);
  }
```

- Αφαιρεί τις συνδέσεις που τερματίστηκαν.

Visibility, Offline, Online Listeners

```
javascript


  window.addEventListener("offline", handleOffline);
  window.addEventListener("online", handleOnline);
  window.addEventListener("beforeunload", flowInterrupt);
  if (enableInterruption) {
  document.addEventListener("visibilitychange", handleVisibilityChange); }
```

6.5. Καθαρισμός Listeners

```
javascript



  return () => {
      if (!listenersInitialized) return;


      window.removeEventListener("offline", handleOffline);
      window.removeEventListener("online", handleOnline);
      window.removeEventListener("beforeunload", flowInterrupt);
      if (enableInterruption) {
          document.removeEventListener("visibilitychange", handleVisibilityChange);
      }


      listenersInitialized = false;
      console.log("VRL_(Listeners): Event listeners removed during cleanup.");
  };
```

Σκοπός:

- Αφαιρεί όλα τα event listeners και επαναφέρει το `**listenersInitialized**` σε `**false**`.

Συνολική Εικόνα

`**getConnections**` Επιστρέφει τις ενεργές συνδέσεις από το Vonage session.

`**checkParticipantOnlineStatus**` Ελέγχει την online κατάσταση των συμμετεχόντων.

`**setupEventListeners**` Ρυθμίζει event listeners για browser και session events.

`**handleOffline**` Ενημερώνει Mendix και UI όταν ο χρήστης είναι offline.

`**handleOnline**` Επαναφέρει το UI και ελέγχει συμμετέχοντες όταν ο χρήστης είναι online

7. Ανάλυση Κώδικα: (connectionInfo.jsx) Connection Manager

Επισκόπηση

Το `Connection Manager` παρέχει εργαλεία για:

1. Καταγραφή των ενεργών συνδέσεων (publishers, subscribers).
2. Καταγραφή πληροφοριών συνδρομητών.
3. Εμφάνιση και απόκρυψη custom alerts μέσω του DOM.

Ο κώδικας χρησιμοποιείται για debugging, monitoring, και ενημέρωση του UI σε πραγματικό χρόνο.

7.1. Συνάρτηση `parseConnectionData`

```
javascript



  const parseConnectionData = (data) => {
      if (!data) return {};

      return Object.fromEntries(new URLSearchParams(data).entries());
  };
```

Σκοπός

- Αναλύει δεδομένα σύνδεσης (`connection.data`) και επιστρέφει ένα αντικείμενο με τα key-value pairs.

Λειτουργία

1. Ελέγχει αν το `data` είναι `null` ή άδειο.
2. Χρησιμοποιεί το `URLSearchParams` για να αναλύσει το string σε key-value pairs.
3. Επιστρέφει τα αποτελέσματα ως αντικείμενο με χρήση `Object.fromEntries`.

Παράδειγμα

```javascript
const connectionData = "user=JohnDoe&role=publisher";
const parsedData = parseConnectionData(connectionData);

console.log(parsedData);

// { user: "JohnDoe", role: "publisher" }
```

7.2. Συνάρτηση `logConnectionInfo`

```
javascript
```

```javascript
const logConnectionInfo = (connections, logType = "Current Connections") => {
    if (!Array.isArray(connections) || connections.length === 0) {

        console.log(`VRL_(Connection Info): No ${logType.toLowerCase()} found.`);
        return;

    }


    connections.forEach((connection) => {

        const parsedData = parseConnectionData(connection.data);
        console.log(`VRL_(Connection Info): ${logType}`, {

            connectionId: connection.connectionId,
            data: parsedData,

            creationTime: connection.creationTime,
            destroyed: connection.destroyed || false,

            destroyedReason: connection.destroyedReason || "N/A",
            capabilities: connection.capabilities || {},
            permissions: connection.permissions || {},

            quality: connection.quality || "Unknown",
            allKeys: Object.keys(connection),

        });
    });

};
```

Σκοπός

- Καταγράφει λεπτομερείς πληροφορίες για όλες τις συνδέσεις (`connections`).

Λειτουργία

1. Ελέγχει αν οι `connections` είναι άδειες ή μη έγκυρες.
2. Για κάθε σύνδεση:
    - Αναλύει τα δεδομένα της σύνδεσης μέσω της `parseConnectionData`.
    - Εκτυπώνει πληροφορίες όπως:
        - `connectionId`
        - `creationTime`
        - `destroyed` (αν έχει καταστραφεί)
        - `capabilities` και `permissions`
        - Ποιότητα σύνδεσης (`quality`).
3. Επιστρέφει πλήρη εικόνα για debugging.

7.3. Συνάρτηση `notifyConnections`

```javascript
export const notifyConnections = (vonageSession, isDisconnected = false) => {
    if (!vonageSession || !vonageSession.isConnected()) {
        console.warn(`VRL_(Connection Info): Unable to fetch ${isDisconnected ? "disconnected" : "connected"} connections - session is not connected.`);
        return;
    }

    const connections = vonageSession.connections || [];
    logConnectionInfo(connections, isDisconnected ? "Disconnected Connections" : "Current Connections");
};
```

Σκοπός

- Ανακτά και καταγράφει τις ενεργές ή αποσυνδεδεμένες συνδέσεις του Vonage session.

Λειτουργία

1. Ελέγχει αν το `vonageSession` είναι ενεργό.
2. Αν όχι, εμφανίζει προειδοποίηση.
3. Ανακτά τις `connections` από το session.
4. Χρησιμοποιεί τη `logConnectionInfo` για να καταγράψει τις συνδέσεις.

Παράδειγμα

```javascript


  notifyConnections(vonageSession);
  // Καταγράφει όλες τις ενεργές συνδέσεις.
```

7.4. Συνάρτηση `notifySubscribers`

```javascript
export const notifySubscribers = (subscribers, vonageSession) => {
    if (!vonageSession || !vonageSession.isConnected()) {
        console.warn("VRL_(Connection Info): Unable to fetch subscribers - session is not connected.");
        return;
    }

    console.log("VRL_(Connection Info): Fetching current subscribers...");

    if (!Array.isArray(subscribers) || subscribers.length === 0) {
        console.log("VRL_(Connection Info): No active subscribers found.");
        return;
    }

    subscribers.forEach((subscriber) => {
        console.log("VRL_(Connection Info): Subscriber Info", {
            subscriberId: subscriber.id,
            streamId: subscriber.stream?.streamId || "Unknown",
            connectionId: subscriber.stream?.connection?.connectionId || "Unknown",
            streamName: subscriber.stream?.name || "Unnamed Stream",
            subscriberProperties: subscriber.properties || {},
        });
    });
};
```

Σκοπός

- Καταγράφει πληροφορίες για όλους τους ενεργούς συνδρομητές (subscribers) ενός session.

Λειτουργία

1. Ελέγχει αν το `vonageSession` είναι συνδεδεμένο.
2. Αν οι `subscribers` είναι κενές ή μη έγκυρες, εκτυπώνει μήνυμα.
3. Για κάθε subscriber:
    - Εκτυπώνει πληροφορίες όπως `streamId`, `connectionId`, `streamName`.

Παράδειγμα

javascript

```
notifySubscribers(vonageSession.subscribers, vonageSession);
// Καταγράφει όλους τους ενεργούς συνδρομητές.
```

7.5. Συνάρτηση `getAlertElements`

```
javascript



  export const getAlertElements = () => ({

      alertTitle: document.getElementById('alertTitle'),
      alertMessage: document.getElementById('alertMessage'),
      customAlert: document.getElementById('customAlert'),




      alertOverlay: document.getElementById('alertOverlay'),
  });
```

Σκοπός

- Εντοπίζει DOM elements που σχετίζονται με τα custom alerts.

Λειτουργία

- Επιστρέφει τα DOM στοιχεία:
  - `alertTitle`: Τίτλος του alert.
  - `alertMessage`: Κείμενο μηνύματος.
  - `customAlert`: Container του alert.

- `alertOverlay`: Overlay που εμφανίζεται μαζί με το alert.

7.6. Συνάρτηση `showCustomAlert`

```
javascript



  export const showCustomAlert = (title = '', message = '') => {
      const { alertTitle, alertMessage, customAlert, alertOverlay } = getAlertElements();

      if (customAlert && alertOverlay) {
          if (alertTitle) alertTitle.textContent = title;
          if (alertMessage) alertMessage.textContent = message;

          customAlert.style.display = 'block';
          alertOverlay.style.display = 'block';
      } else {
          console.error('VRL_(Connection Info): Custom alert elements not found in the DOM.');
      }
  };
```

Σκοπός

- Εμφανίζει ένα custom alert με τίτλο και μήνυμα.

Λειτουργία

1. Ανακτά τα DOM στοιχεία μέσω της `getAlertElements`.
2. Αν υπάρχουν, ενημερώνει τα περιεχόμενα (`title`, `message`) και τα εμφανίζει.

3. Αν όχι, εμφανίζει μήνυμα σφάλματος.

7.7. Συνάρτηση `closeCustomAlert`

```
javascript



  export const closeCustomAlert = () => {
      const { customAlert, alertOverlay } = getAlertElements();

      if (customAlert && alertOverlay) {
          customAlert.style.display = 'none';
          alertOverlay.style.display = 'none';
      } else {
          console.error('VRL_(Connection Info): Custom alert elements not found in the DOM.');
      }
  };
```

Σκοπός

- Κλείνει το custom alert και το overlay.

Λειτουργία

1. Ανακτά τα DOM στοιχεία μέσω της `getAlertElements`.
2. Αν υπάρχουν, αλλάζει το `style.display` σε `none`.

3. Αν όχι, εμφανίζει μήνυμα σφάλματος.

Συνολική Εικόνα

Συνάρτηση	Σκοπός
`parseConnectionData`	Αναλύει δεδομένα σύνδεσης και τα μετατρέπει σε αντικείμενο.
`logConnectionInfo`	Καταγράφει λεπτομέρειες για τις συνδέσεις (publishers, subscribers).
`notifyConnections`	Καταγράφει ενεργές ή αποσυνδεδεμένες συνδέσεις.
`notifySubscribers`	Καταγράφει πληροφορίες για τους ενεργούς συνδρομητές.
`showCustomAlert`	Εμφανίζει custom alerts.
`closeCustomAlert`	Κλείνει τα custom alerts.