

# Getting Started

## What is *INVOX Medical*?

*INVOX Medical* is a medical report dictation platform that has been developed for use in various medical specialties. Thanks to *INVOX Medical*, healthcare professionals can write reports with the help of voice, improving their efficiency and obtaining reports that are richer in content.

*INVOX Medical* can be integrated with any medical record and hospital management system, as well as with departmental care applications and other general-purpose applications. This allows healthcare professionals to dictate directly into patient records, **reducing reporting times and improving productivity**.

The *INVOX Medical* platform **includes specific dictionaries for a multitude of medical specialties**. These dictionaries have been developed from thousands of selected medical reports from each specialty and with the help of professionals in each area to ensure that the system is able to recognise the terms used in each medical specialty.

The benefits of *INVOX Medical* include:

- **Reduced reporting time and cost.**
- **Vocabulary and resources adapted to each medical speciality and to the specific service or centre in which it is used.**
- **Integration with any medical record system, medical management system or general purpose application.**
- **Continuous adaptation to the voice characteristics of each user.**
- **Ability to define templates for the most frequent types of reports.**
- **Mechanisms for the creation and management of own dictionaries of medical words and acronyms.**
- **Ability to scroll through reports using voice commands as well as to define keyboard shortcuts to interact by voice with the reporting application.**

## INVOX Medical SDK

*INVOX Medical* has an SDK that allows integration into different applications.

**The SDK provides the integrator with services to receive the users' voice corresponding to dictation or command invocation, process it with INVOX Medical and return the transcribed text or other events associated with the commands used.**

This SDK consists of a specification, a JavaScript API for web integration and the implementation of speech recognition services (ASR) for free and/or structured grammar-based text dictation. These services include:

- **Speech-to-text transcription** from free text language models created for each medical specialty.
- **Centralised storage** and automatic updating of each user's speech profiles, dictionaries and templates.

## Integration mechanisms







Two *web integration scenarios* are offered via JavaScript:

- **Web integration with Local Dictation Service** The service required for speech recognition tasks will be provided and must be installed on the client machines.
- **Web integration with Remote Dictation Service** The service that will be in charge of the voice recognition tasks will be located in an external machine, thus avoiding installations in the clients.

**IMPORTANT:** For full functionality of the INVOX Medical SDK it is necessary to install a Local Dictation Service in the local machine or to install a Remote Dictation Service in a server. Without these, the integration can't be completed correctly. If you don't have them, please contact Vócali's team and we will provide it to you.

## Supported browsers

INVOX Medical SDK officially supports last year's versions of the following browsers:

Browsers	Local Dictation	Remote Dictation
 Chrome	:heavy_check_mark:	:heavy_check_mark:
 Firefox	:heavy_check_mark:	:heavy_check_mark:
 Edge	:heavy_check_mark:	:heavy_check_mark:
 IExplorer	:x:	:x:
 Vivaldi	:heavy_check_mark:	:heavy_check_mark:
 Opera	:heavy_check_mark:	:heavy_check_mark:

## Download

Download the **INVOX Medical SDK** code together with the provided examples and start integrating it into your web application in simple steps.

- Download INVOX Medical SDK

## Content

Once downloaded, unzip the compressed file and you will see the following folder structure:

```
.
|-- libs/
|   |-- integrations/
|   |   |-- invox-bar/
|   |   |-- invox-dialogs/
|   |-- opus/
|   |-- invox.min.js
|
|-- samples/
|
|-- index.html
|
|-- README.txt
|
|-- VERSION.txt
```

### libs/

The `libs/` folder contains the minified **INVOX Medical SDK library** (`invox.min.js`), along with the `opus/` folder which contains **audio management resources** for Remote Dictation Service and the `integrations/` folder which contains **modules for quick and easy deploy**.

- `integrations/`: two high-level integration examples.
  - `invox-bar/`: INVOX Medical Bar implementation.
- `opus/`: resources required for audio management when using the Remote Dictation Service.
- `invox.min.js`: core of INVOX Medical (JavaScript API).

### samples/

The `samples/` folder contains a set of examples to understand INVOX Medical SDK.

- `common/`: common resources for all pages, such as functions and styles.
- `invox-bar-multiple_textareas/`: INVOX Medical Bar with multiple TextAreas.
- `invox-components/`: presentation of the INVOX Medical components and how they work.
- `index.html`: main page with links to all example pages.

## Deploying examples in *Docker*

This guide demonstrates how to deploy the implementation of the integration examples provided in the zip package using *Docker*.

More information can be found in the Download section.

## Extract the contents of the package

Locate the downloaded .zip package, and proceed to extract its contents.

Once the process is finished, you will see C:\INVOX\_SDK-2.8.X folder.

## Deploy SDK sample pages into a *Docker* container.

### Using *Docker*

1. Make sure *Docker* is installed on your computer.
2. Make sure *Docker* is running on your computer.
3. Open a terminal cmd or powershell.
4. Execute command, replacing the variable {PATH\_TO\_INVOX\_SDK-2.8.X\_FOLDER} for C:\INVOX\_SDK-2.8.X folder:

```
docker run --name INVOX_SDK_SAMPLES -v {PATH_TO_INVOX_SDK-2.8.X_FOLDER}:/usr/local/apache2/htdocs/ -p 8080:80 -d httpd:2.4
```

5. Container INVOX\_SDK\_SAMPLES must be created and running.

### Using *Docker-compose*

1. Make sure *Docker* and *Docker-compose* are installed on your computer.
2. Make sure *Docker* is running on your computer.
3. Open a terminal cmd or powershell.
4. Move to C:\INVOX\_SDK-2.8.X\docker folder.
5. Execute command:

```
docker-compose up
```

6. Container INVOX\_SDK\_SAMPLES must be created and running.

## Next steps

You can check the website accessing ***http://127.0.0.1:8080/*** or ***http://localhost:8080/*** from a browser. The result should look like the following image.

From here, you can refer to the Samples pages section for a more in-depth look at each of them.

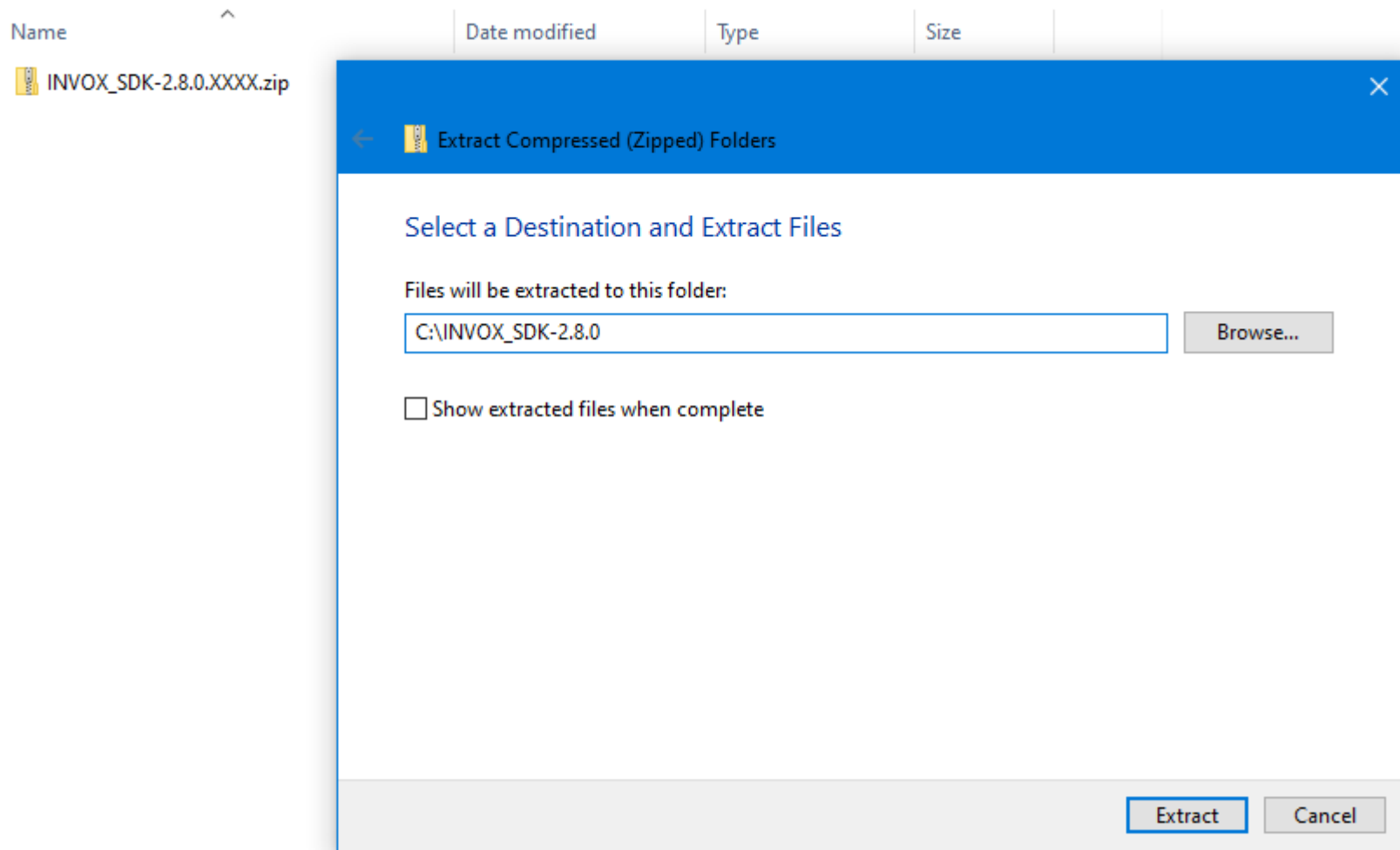


Figure 1: Extract package content



# SDK Integration Examples

## INVOX Medical Bar & Components

### INVOX Medical Bar Integration with multiple text editors

On this page you will find the INVOX Medical Bar, which is an example of integration with all the components. You can also see how it works with multiple writing destinations and how to change the focus between them.

[Open example](#) →

### INVOX Medical Components

On this page you will find all the INVOX Medical components. You can see how they work and play with them.

[Open example](#) →

Figure 2: Next steps

## Deploying examples in IIS

This guide demonstrates how to configure a new website in *Internet Information Services (IIS)* and implement the integration examples provided in the zip package.

More information can be found in the Download section.

### Extract the contents of the package

Locate the downloaded .zip package, and proceed to extract its contents.

Once the process is finished, you will see the following content in the C:\INVOX\_SDK-2.8.X folder.

Now we need to copy this content to the following path C:\inetpub\wwwroot\INVOX\_SDK-2.8.X, so that we can link it to our website when we have created it.

### Open the IIS Manager

Open *Internet Information Services Manager (IIS)*. You can find it by typing “IIS” in the search field.

### Enable mimetype wasm

This step is only required with Remote Dictation Service

This mimetype is necessary to send audio from browser with HTML5.

Open MIME Types.

Right click, select **Add...** and set the values:

Field	Value
File name extension	<b><i>.wasm</i></b>
MIME type	<b><i>application/wasm</i></b>

### Add new website for SDK examples.

On the root of the sites, right click and select the **Add Website...** option.

### Create site

Fill in the necessary data:

- **Site name:** The name of the site should be something relevant and easy to identify. In this case, we write ***INVOX-SDK***.

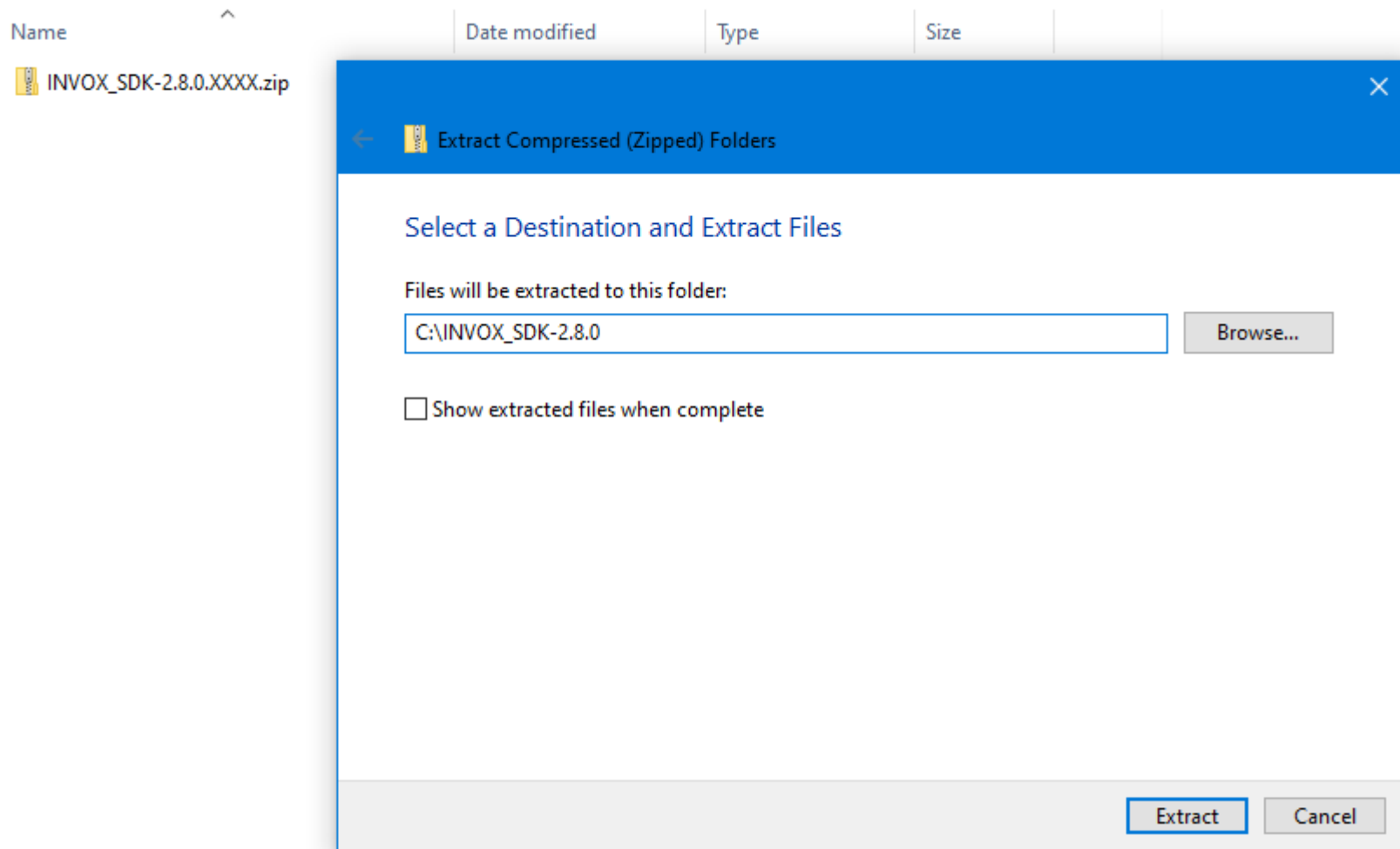


Figure 3: Extract package content



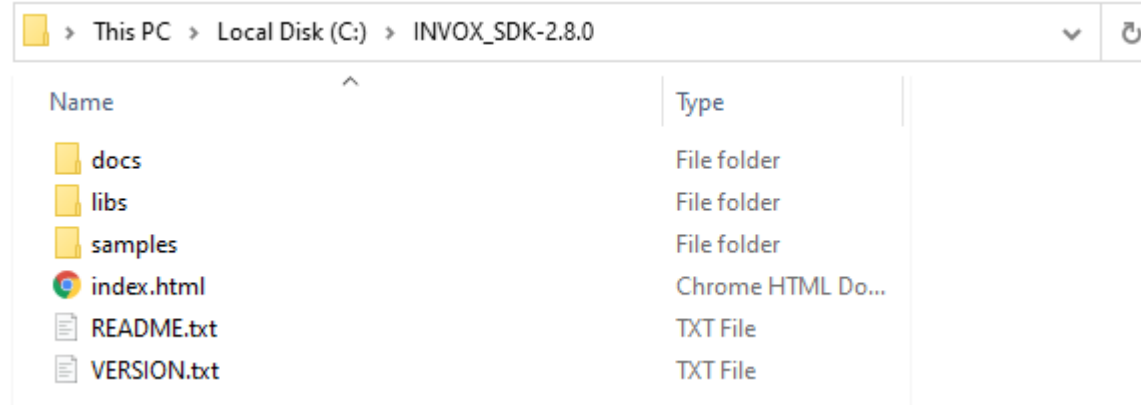


Figure 4: Package content

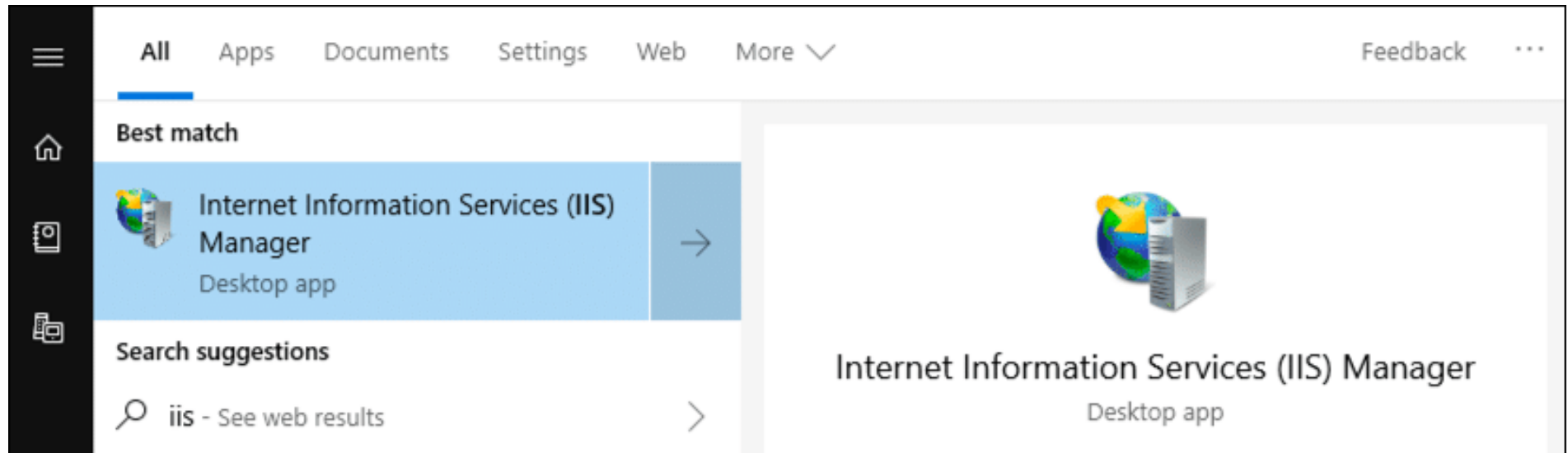


Figure 5: Search

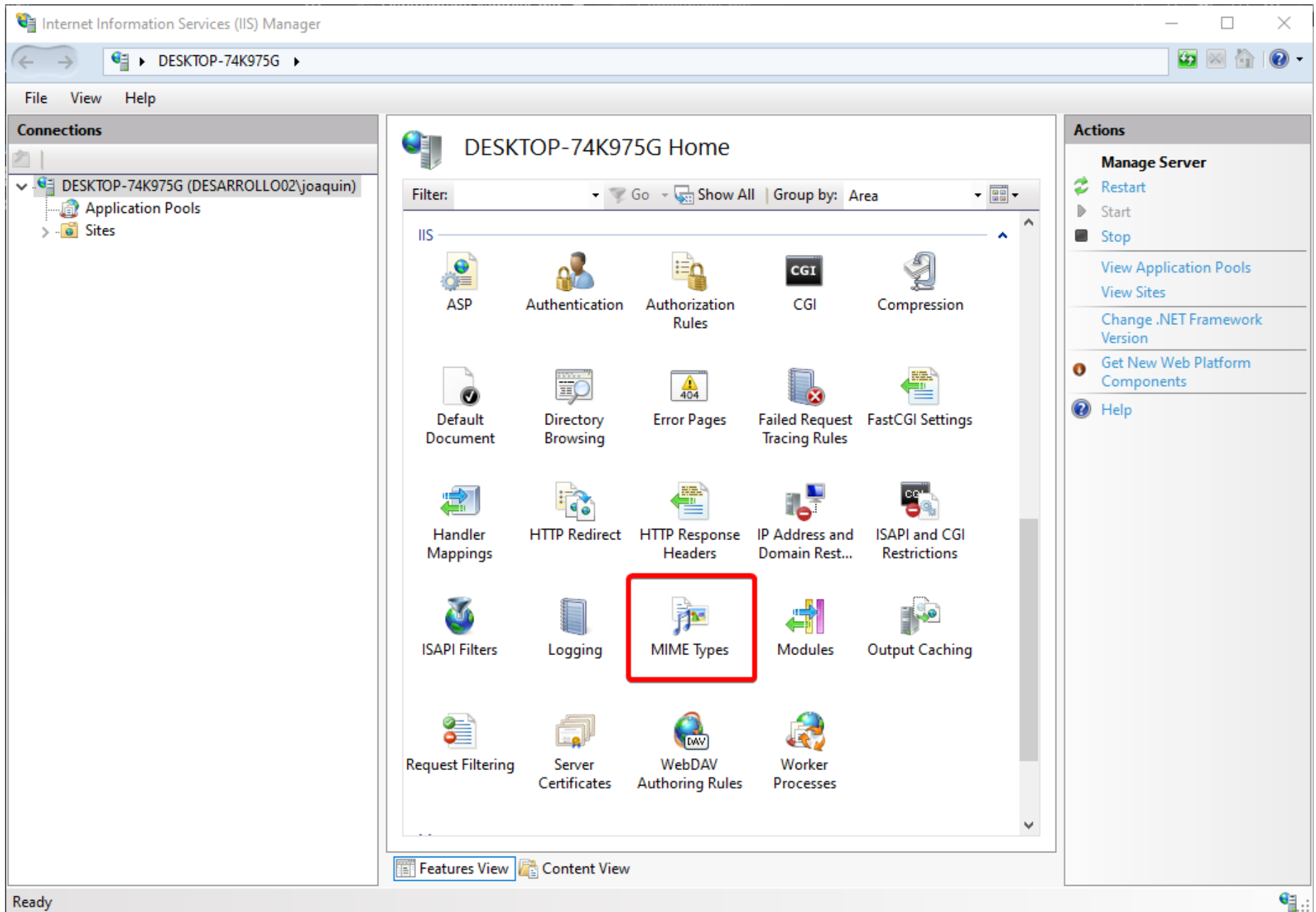


Figure 6: Search  
10

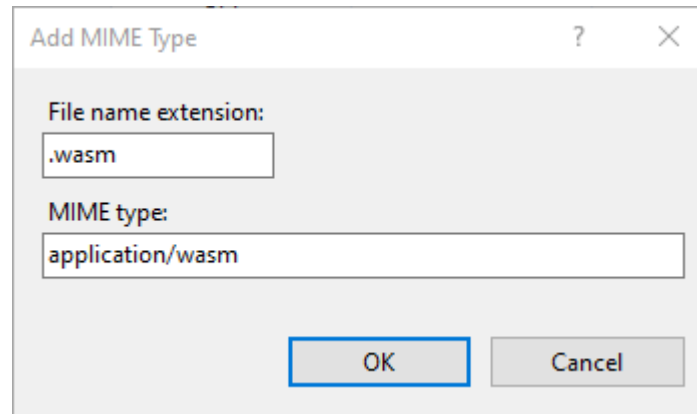


Figure 7: Search

- **Content directory > Physical path:** You must specify the path to the directory where you have extracted the content of the package in point 1 of this guide. To do this, use the button on the right to navigate to it, as shown in the following image.

**Attention:** we must select the root folder that contains the subfolders *libs* and *samples*.

- **Creation of links:** You must specify the host name you want to set up, and the protocol to use. In this case, we use HTTP, for all unallocated IPS, on port 80. As hostname, we write *mysdk.localhost*.

Accept the changes and the website will be successfully created.

## Next steps

You can check the website accessing *mysdk.localhost* from a browser. The result should look like the following image.

From here, you can refer to the Samples pages section for a more in-depth look at each of them.

## Quick start

### Introduction

To perform an **agile and effortless** deployment, check out the Quick Integration Components. These web components integrate the full functionality of the API along with a user interface that enables interaction with the system. Visit this section to explore the web components and their capabilities.

The INVOX Medical SDK offers essential functionalities for user authentication, speech recognition initiation and termination, and configuring the editor to display the results of speech recognition. For detailed information on these functions and more, please refer to the API documentation.

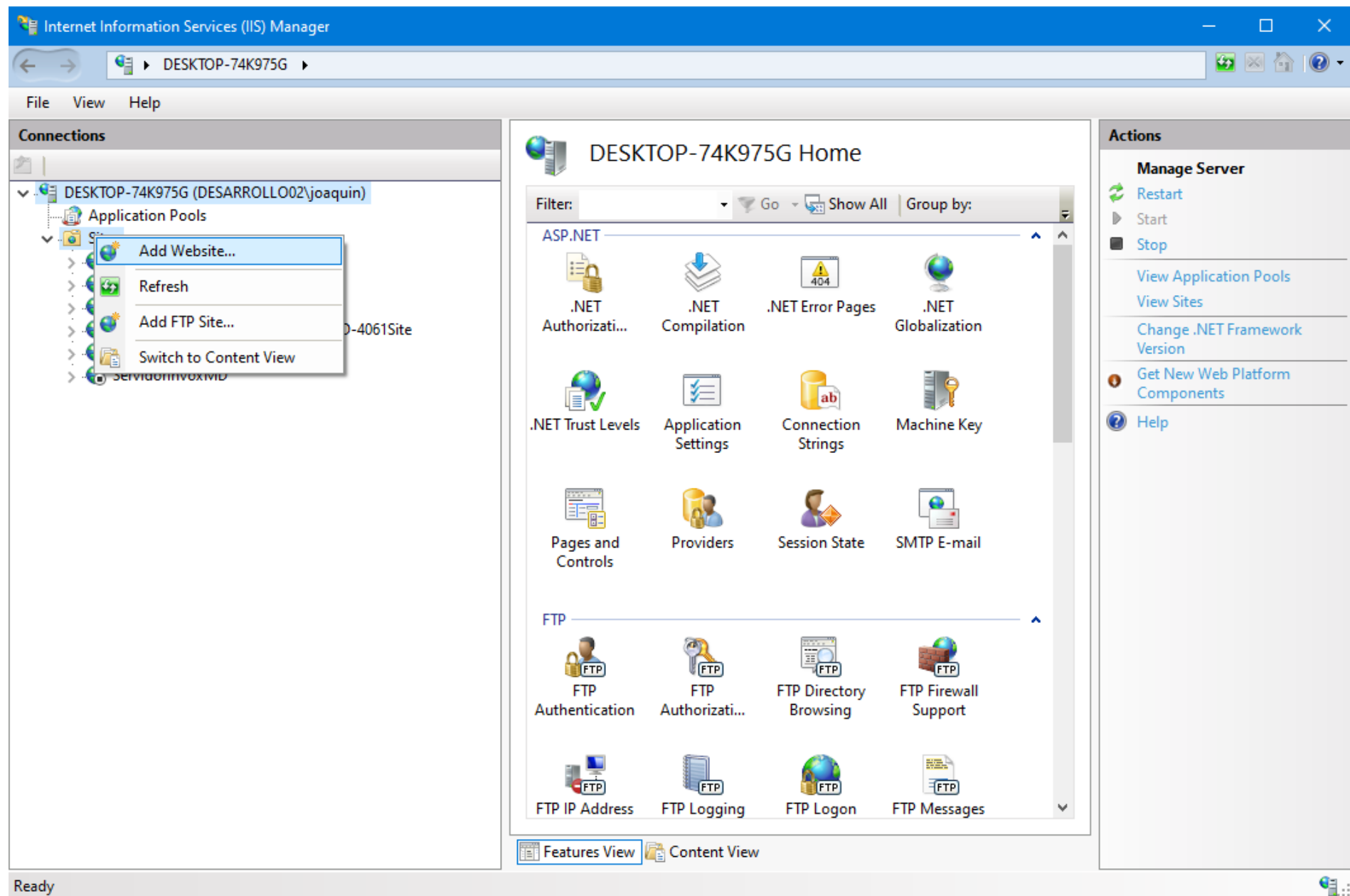


Figure 8: Create site name

Add Website

Site name: INVOX-SDK Application pool: INVOX-SDK Select...

Content Directory

Physical path: ...

Pass-through authentication

Connect as... Test Settings...

Binding

Type: http IP address: All Unassigned Port: 80

Host name:

Example: www.contoso.com or marketing.contoso.com

☒ Start Website immediately

OK Cancel

Figure 9: Create site 1

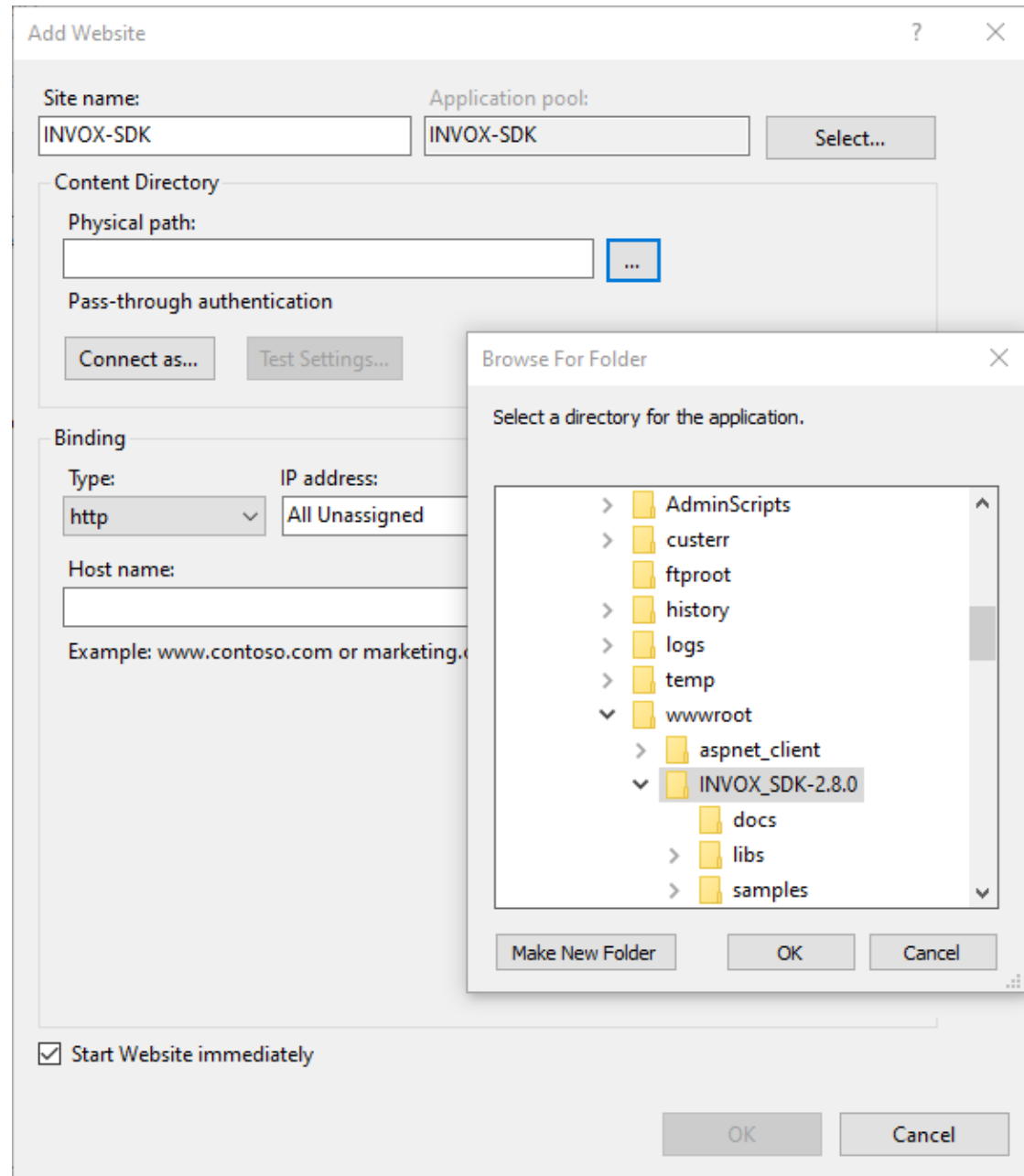


Figure 10: Create site 2

Add Website

Site name: INVOX-SDK Application pool: INVOX-SDK Select...

Content Directory

Physical path: C:\inetpub\wwwroot\INVOX\_SDK-2.8.0 ...

Pass-through authentication

Connect as... Test Settings...

Binding

Type: http IP address: All Unassigned Port: 80

Host name: mysdk.localhost

Example: www.contoso.com or marketing.contoso.com

☒ Start Website immediately

OK Cancel

Figure 11: Create site 2



# SDK Integration Examples

## INVOX Medical Bar & Components

### INVOX Medical Bar Integration with multiple text editors

On this page you will find the INVOX Medical Bar, which is an example of integration with all the components. You can also see how it works with multiple writing destinations and how to change the focus between them.

[Open example](#) →

### INVOX Medical Components

On this page you will find all the INVOX Medical components. You can see how they work and play with them.

[Open example](#) →

Figure 12: Next steps



This section will guide you through the development of the essential functionality required to establish a connection with the dictation service, perform dictation, and display the results in a text editor.

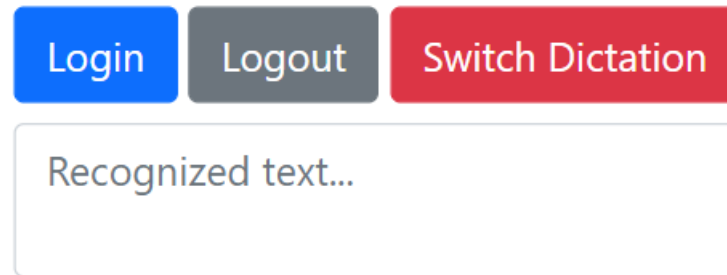


Figure 13: Quick start example basic page

## Integration Scenarios

Depending on your specific deployment needs, you have the flexibility to use either one or both services for testing and developing your integration. For more information on the available options, please refer to the Web Integration Scenarios section.

## Folder Structure

The folder and file structure for this example is straightforward. Create a root folder named `inbox-quick-start` that includes the following files:

```
.
|-- inbox-quick-start/
|   |-- opus/                <- Opus Audio Library required from Remote Dictation Service
|   |-- inbox.min.js
|   |-- index.html
```

- The crucial aspect is to have the `opus/` folder and its content at the same level as the `inbox.min.js` file. Please remember to include this audio library in order to connect to the Remote Dictation Service.
- The full code of this example will be located in the `index.html` file.

## Full Code

The following code includes all the necessary components to execute the example, and the subsequent sections will provide a step-by-step explanation of its implementation.

```
<!DOCTYPE html>
```

```

<html lang="EN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
  <meta http-equiv='cache-control' content='no-cache'>
  <meta http-equiv='expires' content='0'>
  <meta http-equiv='pragma' content='no-cache'>

  <!-- INVOX Medical SDK-->
  <script type="text/javascript" src="invox.min.js" charset="UTF-8"></script>

  <!-- Bootstrap -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" >
</head>
<body>
  <main class="m-2">
    <button class="btn btn-primary" onclick="DoLogin()">Login</button>
    <button class="btn btn-secondary" onclick="DoLogout()">Logout</button>
    <button class="btn btn-danger" onclick="SwitchDictation()">Switch Dictation</button>
    <textarea id="invox-textarea-1" class="form-control mt-2" placeholder="Recognized text..."></textarea>
  </main>
  <script>
    async function DoLogin() {
      const credentials = {
        user: "provided_username",
        password: "provided_password"
      }

      const connectToLocal = {
        host: "localhost",    // Default host when "useDictationService" is false
        port: "8443",        // Default port
        useDictationService: false
      }

      const connectToRemote = {
        host: "dev.invoxmedical.com",
        port: "8443",
        useDictationService: true
      }
    }
  </script>

```

```

    try {
        await INVOX.Login(credentials, connectToLocal);
        INVOX.LogInfo("Local Dictation Service connected successfully.");
    } catch (error) {
        INVOX.LogWarn("Local Dictation Service not found. Trying to connect to Remote Dictation Service...");

        try {
            await INVOX.Login(credentials, connectToRemote);
            INVOX.LogInfo("Remote Dictation Service connected successfully.");
        } catch (error) {
            INVOX.LogError("Remote Dictation Service not found. Make sure that connection parameters are correct.");
        }
    }
}

function DoLogout() {
    INVOX.Logout()
        .then(() => {
            INVOX.LogInfo("Session closed successfully")
        })
        .catch((error) => {
            INVOX.LogError(error.message)
        })
}

function SwitchDictation() {
    try {
        INVOX.SwitchDictation()
    } catch(error) {
        INVOX.LogError(error.message)
    }
}

document.addEventListener(INVOX.eventTypeReport.LOGIN_ERROR, (event) => {
    INVOX.LogError(event.detail)
    alert(event.detail)
})

document.addEventListener(INVOX.eventTypeReport.LOGIN_SUCCESS, () => {
    INVOX.LogInfo(event.detail)
    try {
        INVOX.SetTextWriter(INVOX.TextAreaTextWriter)

        const textareaElement = document.getElementById("inbox-textarea-1")
    }
}

```

```

        INVOX.SetWriterTarget(textareaElement)
    } catch(error) {
        INVOX.LogError(error.message)
    }
})
document.addEventListener(INVOX.eventTypeReport.NOT_CUSTOMIZED_COMPONENTS, (event) => {
    INVOX.LogWarn(event.detail)
})
</script>
</body>
</html>

```

## Import resources

To start with, the INVOX Medical SDK library must be imported into the header, and Bootstrap will be used for some styles (you can skip this).

```

<head>
    ...

    <!-- INVOX Medical SDK-->
    <script type="text/javascript" src="invoy.min.js" charset="UTF-8"></script>

    <!-- Bootstrap -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" >
</head>

```

## Log in

To initiate the login process, you need to pass user credentials and connection data to the Login function.

Please refer to the Essential functions section for a comprehensive list of available methods and functionalities.



Figure 14: Login Button

## HTML

```
<button class="btn btn-primary" onclick="DoLogin()">Login</button>
```

**JavaScript** The following code is implemented to attempt connection to the Local Dictation Service first. If no service is found locally, it then tries to connect to the Remote Dictation Service. **This approach allows for automated integration with both services.**

```
async function DoLogin() {

    const credentials = {
        user: "provided_username",
        password: "provided_password"
    }

    const connectToLocal = {
        host: "localhost",    // Default host when "useDictationService" is false
        port: "8443",        // Default port
        useDictationService: false
    }

    const connectToRemote = {
        host: "dev.invoxmedical.com",
        port: "8443",
        useDictationService: true
    }

    try {
        await INVOX.Login(credentials, connectToLocal);
        INVOX.LogInfo("Local Dictation Service connected successfully.");
    } catch (error) {
        INVOX.LogWarn("Local Dictation Service not found. Trying to connect to Remote Dictation Service...");

        try {
            await INVOX.Login(credentials, connectToRemote);
            INVOX.LogInfo("Remote Dictation Service connected successfully.");
        } catch (error) {
            INVOX.LogError("Remote Dictation Service not found. Make sure that connection parameters are correct.");
        }
    }
}
```

## Subscribe Login Events

INVOX Medical allows to subscribe to a certain events related to the authentication process.

Please refer to the Capture Internal Events section for a comprehensive list of available methods and functionalities.

### JavaScript

```
document.addEventListener(INVOX.eventTypeReport.LOGIN_ERROR, (event) => {
    INVOX.LogError(event.detail)
    alert(event.detail)
})

document.addEventListener(INVOX.eventTypeReport.LOGIN_SUCCESS, (event) => {
    INVOX.LogInfo(event.detail)
})

document.addEventListener(INVOX.eventTypeReport.NOT_CUSTOMIZED_COMPONENTS, (event) => {
    INVOX.LogWarn(event.detail)
})
```

## Set writing location

It is crucial to configure the Writer after a successful login. This can be done by utilizing the promise returned by the Login method or by handling the Login Success event. Ensuring proper configuration of the Writer ensures that the dictation results are accurately written to the designated output component.

After successfully logging in and prior to starting the dictation, the writing area must be configured to display the recognition results. This will be achieved by creating a simple text box using the HTML element TextArea, with the identifier `invoy-textarea-1`.

Please refer to the Writer functions section for a comprehensive list of available methods and functionalities.

### HTML

```
<textarea id="invoy-textarea-1" class="form-control" placeholder="Recognized text..."></textarea>
```

**JavaScript** To enable writing of dictation results in a text box, you need to configure a TextWriter. The INVOX Medical SDK provides the TextAreaTextWriter for this purpose. To specify the TextAreaTextWriter, use the provided method and set the target element as the HTML element representing the text box where the output will be displayed.

Please refer to the Writing in other components section for a comprehensive list of available methods and functionalities.

```
// Subscribe to INVOX Login Success event and set behaviour
```



Figure 15: TextArea Target

```
document.addEventListener(INVOX.eventTypeReport.LOGIN_SUCCESS, () => {  
  
    INVOX.LogInfo(event.detail)  
  
    // Add following lines  
    try {  
        INVOX.SetTextWriter(INVOX.TextAreaTextWriter)  
  
        const textareaElement = document.getElementById("inbox-textarea-1")  
        INVOX.SetWriterTarget(textareaElement)  
    } catch(error) {  
        INVOX.LogError(error.message)  
    }  
})
```

## Start or stop speech recognition

We will create a button that executes *INVOX.SwitchDictation*, which will start or stop the dictation depending on its internal state.

Please refer to the Dictation functions section for a comprehensive list of available methods and functionalities.

### HTML

```
<button class="btn btn-danger" onclick="SwitchDictation()">Switch Dictation</button>
```

### JavaScript

```
function SwitchDictation() {  
    try {
```



Figure 16: Switch Dictation Button

```
        INVOX.SwitchDictation()
    } catch(error) {
        INVOX.LogError(error.message)
    }
}
```

## Log out

Finally, we will implement a logout button that allows the user to end the session. Upon logging out, the start dictation button will become inactive and no longer trigger any actions. This ensures that the user is logged out and no further dictation actions can be performed.



Figure 17: Logout Button

## HTML

```
<button class="btn btn-secondary" onclick="DoLogout()">Logout</button>
```

## JavaScript

```
function DoLogout() {
    INVOX.Logout()
        .then(() => {
            INVOX.LogInfo("Session closed successfully")
        })
}
```



```
    .catch((error) => {  
        INVOX.LogError(error.message)  
    })  
}
```

## Next Steps

- Take a look at the Quick Integration Components for a streamlined integration experience. :link:
- Explore the Test Scenarios to better understand various usage scenarios. :link:
- Refer to the JavaScript API documentation to explore all the available methods. :link:
- Learn how to build your own Text Writer for customizing the text output in your editor. :link: