MedDream Viewer Communication API

Version 1.0.12 (2023-03-10)

Add component to your project

Import and create new Viewer Communication component in your project:

```
const viewerCommunication = new ViewerCommunication(targetURL, integration);
```

Parameters:

- `targetURL` - MedDream Viewer URL.
- `integration` (Optional) - Integration type: `study` or `token`. Default value: `study`.

Window reference functions

Get current Viewer window reference

```
const windowReference = viewerCommunication.getWindowReference();
```

Find available Viewer window reference

```
const windowReference = viewerCommunication.findWindowReference();
```

Focus available window

```
viewerCommunication.focusWindow();
```

Post action message to MedDream Viewer

```
viewerCommunication.postActionMessage(actionType, actionData);
```

Parameters:

- `actionType` - Action message command type.
- `actionData` - Data needed for action message.

For more details about available action messages check: `MedDream communication documentation`.

Viewer communication integration functions

Get current integration type

```
const integrationType = viewerCommunication.getIntegrationType();
```

Update integration type

```
viewerCommunication.updateIntegrationType(newIntegrationType);
```

Parameter:

- `newIntegrationType` - Integration type: `study` or `token`.

Functions to open MedDream Viewer

Open studies in MedDream Viewer window

```
viewerCommunication.openInMedDreamWindow(studies/token);
```

Parameters:

- `studies` (For `study` integration) - Study uid's list separated with `,`.
- `token` (For `token` integration) - Token with study information.

Add studies to MedDream Viewer window

```
viewerCommunication.addToMedDreamWindow(studies/token);
```

Parameters:

- `studies` (For `study` integration) - Study uid's list separated with `,`.
- `token` (For `token` integration) - Token with study information.

Replace studies in MedDream Viewer window

```
viewerCommunication.replaceInMedDreamWindow(studies/token);
```

Parameters:

- `studies` (For `study` integration) - Study uid's list separated with `,`.
- `token` (For `token` integration) - Token with study information.

Open MedDream with studies to iframe

```
viewerCommunication.openMedDreamToIframe(iframeId, studies/token);
```

Parameters:

- `iframeId` - Iframe element id.
- `studies` (For `study` integration) - Study uid's list separated with `,`.
- `token` (For `token` integration) - Token with study information.

Communication functions

Functions only for Study integration

Open study

```
viewerCommunication.openStudy(study);
```

Parameter:

- `study` - Study uid.

Open studies

```
viewerCommunication.openStudies(studies);
```

Parameter:

- `studies` - Array of study uid's.

Replace studies

```
viewerCommunication.replaceStudies(studies);
```

Parameter:

- `studies` - Array of study uid's.

Preload studies

```
viewerCommunication.preloadStudies(studies);
```

Parameter:

- `studies` - Array of study uid's.

Cache studies

```
viewerCommunication.cacheStudies(studies);
```

Parameter:

- `studies` - Array of study objects. Each study object has ***studyUid*** and ***storageId*** parameters.

Array example:

```
const studies = [
    {
        studyUid: 'study-uid-1',
        storageId: 'storage-id'
    },
    {
        studyUid: 'study-uid-2',
        storageId: 'storage-id'
    }
];
```

Close studies

```
viewerCommunication.closeStudies(studies);
```

Parameter:

- `studies` - Array of study objects. Each study object has ***studyUid*** and ***storageId*** parameters.

Array example:

```
const studies = [
    {
        studyUid: 'study-uid-1',
        storageId: 'storage-id'
    }
];
```

Functions only for Token integration

Open studies

```
viewerCommunication.openStudies(token);
```

Parameter:

- `token` - Generated token with studies information.

Replace studies

```
viewerCommunication.replaceStudies(token);
```

Parameter:

- `token` - Generated token with studies information.

Preload studies

```
viewerCommunication.preloadStudies(token);
```

Parameter:

- `token` - Generated token with studies information.

Cache studies

```
viewerCommunication.cacheStudies(token);
```

Parameter:

- `token` - Generated token with studies information.

Close studies

```
viewerCommunication.closeStudies(token);
```

Parameter:

- `token` - Generated token with studies information.

Common functions

Cache all studies

```
viewerCommunication.cacheAllStudies();
```

Close all studies

```
viewerCommunication.closeAllStudies();
```

Set layout

```
viewerCommunication.setLayout(columns, rows);
```

Parameters:

- `columns` - Number of columns.
- `rows` - Number of rows.

Open instance

```
viewerCommunication.openInstance(instanceUid, viewportColumn, viewportRow, viewportActions);
```

Parameters:

- `instanceUid` - Unique instance uid which has to be opened to viewport.
- `viewportColumn` - Column number of desired viewport.
- `viewportRow` - Row number of desired viewport.
- `viewportActions` - Object of actions which have to be performed on viewport after instance is loaded.

Available viewport actions:

- `windowing` - Windowing level. Available options: ***"DEFAULT"***, ***"AUTO"***, ***"CUSTOM"***. If ***"CUSTOM"*** windowing is selected, ***customWindowing*** parameter has to be defined in ***viewportActions*** object.
- `customWindowing` - Custom windowing level. This parameter allows to set custom windowing ***width*** and ***center*** levels. ***customWindowing*** has to be defined only when ***"CUSTOM"*** windowing is selected.
- `rotation` - Instance rotation by defined number of degrees.
- `verticalFlip` - Vertical instance flip. Available options: ***true***/***false***.
- `horizontalFlip` - Horizontal instance flip. Available options: ***true***/***false***.
- `scale` - Instance scaling option. Available options: ***"ORIGINAL"***, ***"FIT_TO_SCREEN"***, ***"CUSTOM"***. If ***"CUSTOM"*** scale is selected, ***customScale*** parameter has to be defined in ***viewportActions*** object.
- `customScale` - Custom scale number.
- `alignment` - Instance alignment in viewport. Available options: ***"RIGHT"***, ***"LEFT"***, ***"CENTER"***.

Viewport actions object example:

```
const viewportActions = {
    windowing: 'CUSTOM',                    //DEFAULT, AUTO, CUSTOM
    customWindowing: {width: 2, center: 2}, //Use if custom windowing
    rotation: 45,
    verticalFlip: true,
    horizontalFlip: true,
    scale: 'CUSTOM',                        //ORIGINAL, FIT_TO_SCREEN, CUSTOM
    customScale: 0.5,                       //Use if custom scale
    alignment: 'CENTER'                     //RIGHT, LEFT, CENTER
};
```

Export instance

```
viewerCommunication.exportInstance(viewportColumn, viewportRow);
```

Parameters:

- `viewportColumn` (Optional) - Column number of desired viewport.
- `viewportRow` (Optional) - Row number of desired viewport.

Currently active viewport instance is exported, if `viewportColumn` and `viewportRow` are not provided.

Update segmentation tool permissions

```
viewerCommunication.updateSegmentationToolPermissions(permissions);
```

Parameter:

- `permissions` - Object with segmentation permissions.

Available segmentation permissions:

- `boundingBoxView` - Permission to see bounding box tab. Default value: ***false***.
- `boundingBox2dEdit` - Permission to edit 2d bounding box tab. Default value: ***false***.
- `boundingBox3dEdit` - Permission to edit 3d bounding box tab. Default value: ***false***.
- `boundingBoxInfo` - Permission to see bounding box information button and panel. Default value: ***false***.
- `freeDrawView` - Permission to see free draw tab. Default value: ***false***.
- `freeDrawEdit` - Permission to edit free draw tab. Default value: ***false***.
- `smartPaintView` - Permission to see smart paint tab. Default value: ***false***.
- `smartPaint2dEdit` - Permission to use 2d smart paint tool. Default value: ***false***.
- `smartPaint3dEdit` - Permission to use 3d smart paint tool. Default value: ***false***.
- `smartPaintInfo` - Permission to see smart paint information button and panel. Default value: ***false***.

Usage example:

```
const permissions = {
    boundingBoxView: true,
    boundingBox2dEdit: true,
    boundingBox3dEdit: true,
    boundingBoxInfo: false,
    freeDrawView: true,
    freeDrawEdit: true,
    smartPaintView: true,
    smartPaint2dEdit: true,
    smartPaint3dEdit: true,
    smartPaintInfo: false
};
viewerCommunication.updateSegmentationToolPermissions(permissions);
```

Get opened studies

```
 const callback = (studies) => console.log(studies);
viewerCommunication.subscribeGetOpenedStudiesEvent(callback);
viewerCommunication.getOpenedStudies();
```

Usage:

- Register ***subscribeGetOpenedStudiesEvent callback*** function.

- Call ***getOpenedStudies*** function to request opened studies data in callback function.
- Once message is processed, ***callback*** function will be triggered with opened studies array.

Get viewport data

```
const callback = (viewportData) => console.log(viewportData);
viewerCommunication.subscribeGetViewportDataEvent(callback);
viewerCommunication.getViewportData();
```

Usage:

- Register ***subscribeGetViewportDataEvent callback*** function.
- Call ***getViewportData*** function to request active viewport data in callback function.
- Once message is processed, ***callback*** function will be triggered with viewport data object.

Get snapshot

```
const callback = (snapshot) => console.log(snapshot);
viewerCommunication.subscribeGetSnapshotEvent(callback);
viewerCommunication.getSnapshot();
```

Usage:

- Register ***subscribeGetSnapshotEvent callback*** function.
- Call ***getSnapshot*** function to generate current layout with viewports snapshot and return it to ***callback*** function.
- Once message is processed, ***callback*** function will be triggered with snapshot data.

Set snapshot

```
viewerCommunication.setSnapshot(layoutSnapshot);
```

Parameter:

- `layoutSnapshot` - layout and viewports snapshot which was requested by ***getSnapshot*** function and returned to ***callback*** function.

Events

Subscribe communication service ready event

```
const callback = (annotations) => console.log(annotations);
viewerCommunication.subscribeCommunicationServiceReadyEvent(callback);
```

Parameter:

- `callback` - Callback function which is called when event is triggered.

Unsubscribe communication service ready event

```
viewerCommunication.unsubscribeCommunicationServiceReadyEvent();
```

Subscribe get opened studies event

```
const callback = (studies) => console.log(studies);
viewerCommunication.subscribeGetOpenedStudiesEvent(callback);
```

Parameter:

- `callback` - Callback function which is called when event is triggered.

Unsubscribe get opened studies event

```
viewerCommunication.unsubscribeGetOpenedStudiesEvent();
```

Subscribe get viewport data event

```
const callback = (viewportData) => console.log(viewportData);
viewerCommunication.subscribeGetViewportDataEvent(callback);
```

Parameter:

- `callback` - Callback function which is called when event is triggered.

Unsubscribe get viewport data event

```
viewerCommunication.unsubscribeGetViewportDataEvent();
```

Subscribe get snapshot event

```
const callback = (snapshot) => console.log(snapshot);
viewerCommunication.subscribeGetSnapshotEvent(callback);
```

Parameter:

- `callback` - Callback function which is called when event is triggered with generated snapshot information.

Unsubscribe get snapshot event

```
viewerCommunication.unsubscribeGetSnapshotEvent();
```

Subscribe study loaded event

```
const callback = (study) => console.log(study);
viewerCommunication.subscribeStudyLoadedEvent(callback);
```

Parameter:

- `callback` - Callback function which is called when event is triggered.

Unsubscribe study loaded event

```
viewerCommunication.unsubscribeStudyLoadedEvent();
```

Subscribe annotations saved event

```
const callback = (annotations) => console.log(annotations);
viewerCommunication.subscribeAnnotationsSavedEvent(callback);
```

Parameter:

- `callback` - Callback function which is called when event is triggered.

Unsubscribe annotations saved event

```
viewerCommunication.unsubscribeAnnotationsSavedEvent();
```

Change log

1.0.12 (2023-03-10)

Changes

- Updated old `getWindowReference` function name to `findWindowReference`.
- Added new `getWindowReference` function which returns last received window reference.

1.0.11 (2023-03-07)

Changes

- Removed information about not used `freeDrawInfo` permission from example and `Update segmentation tool permissions` documentation section.

Change log

1.0.10 (2022-12-19)

Changes

- Added `getViewportData` function to get active viewport data.
- Added `subscribeGetViewportDataEvent` function to subscribe of get viewport data event callback.
- Added `unsubscribeGetViewportDataEvent` function to unsubscribe of get viewport data event callback.

1.0.9 (2022-11-08)

Changes

- Added `subscribeStudyLoadedEvent` function to subscribe of study loaded event callback.
- Added `unsubscribeStudyLoadedEvent` function to unsubscribe of study loaded event callback.

1.0.8 (2022-10-06)

Changes

- Added `getIntegrationType` function to return current integration type.
- Added `updateIntegrationType` function to update integration type.
- Updated integration type dropdown in example to actually update integration type in library when new integration type is selected.

1.0.7 (2022-03-14)

Changes

- Added `getSnapshot` function to generate viewer layout and viewports snapshot.
- Added `setSnapshot` function to set previously generated snapshot back to the viewer.
- Added `subscribeGetSnapshotEvent` function to subscribe of get snapshot event callback.
- Added `unsubscribeGetSnapshotEvent` function to unsubscribe of get snapshot event callback.

1.0.6 (2021-12-15)

Changes

- Updated `updateSegmentationToolPermissions` function to support new permissions: `smartPaintView`, `smartPaint2dEdit`, `smartPaint3dEdit`, `smartPaintInfo`.

1.0.4 (2021-11-11)

Breaking changes

- Updated segmentation permission `boundingBoxEdit` to `boundingBox2dEdit` and `boundingBox3dEdit` for 2d and 3d bounding box permissions control.

1.0.3 (2021-09-28)

Changes

- Added `updateSegmentationToolPermissions` function to update segmentation tool permissions.
- Added `subscribeCommunicationServiceReadyEvent` function to subscribe communication service ready event.
- Added `unsubscribeCommunicationServiceReadyEvent` function to unsubscribe communication service ready event.
- Added `unsubscribeGetOpenedStudiesEvent` function to unsubscribe get opened studies event.
- Added `subscribeAnnotationsSavedEvent` function to subscribe annotation saved event.
- Added `unsubscribeAnnotationsSavedEvent` function to unsubscribe annotation saved event.

Breaking changes

- Renamed `onGetOpenedStudies` function to `subscribeGetOpenedStudiesEvent`.

1.0.2 (2021-09-22)

Changes

- Added `openMedDreamToIframe` function to open studies in iframe.

Breaking changes

- Renamed `openInMedDream` function to `openInMedDreamWindow`.
- Renamed `addToMedDream` function to `addToMedDreamWindow`.
- Renamed `replaceInMedDream` function to `replaceInMedDreamWindow`.