# MedDream Viewer Communication API

Version 1.0.31 (2024-07-29)

## 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 viewports information

```
const callback = (viewportsInformation) => console.log(viewportsInformation);
viewerCommunication.subscribeGetViewportsInformationEvent(callback);
viewerCommunication.getViewportsInformation();
```

Usage:

- Register ***subscribeGetViewportsInformationEvent callback*** function.
- Call ***getViewportsInformation*** function to request viewports information in callback function.
- Once message is processed, ***callback*** function will be triggered with viewports information array.

### Get instance metadata

```
const callback = (instanceMetadata) => console.log(instanceMetadata);
viewerCommunication.subscribeGetInstanceMetadataEvent(callback);
viewerCommunication.getInstanceMetadata(instanceUid);
```

Parameter:

- `instanceUid` - Unique instance uid which metadata you want.

Usage:

- Register ***subscribeGetInstanceMetadataEvent callback*** function.
- Call ***getInstanceMetadata*** function with instanceUid to request instance metadata in callback function.
- Once message is processed, ***callback*** function will be triggered with instance metadata 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.

### Update button visibility

```
viewerCommunication.updateButtonVisibility(buttonsVisibility);
```

Parameter:

- `buttonsVisibility` - Object consistent of toolbar button names(keys) and their visibility value - true(hidden), false(shown).

Object example:

```
const buttonsVisibility = {
    'dicom-tag-list': true,
    'mpr-mist-oblique': true,
    'key-object-selection': true
}
```

### Show info labels

```
viewerCommunication.showInfoLabels(value);
```

Parameter:

- `value` - boolean to show or hide viewports labels.

### Set additional info labels

```
viewerCommunication.setAdditionalInfoLabels(labels);
```

Parameter:

- `labels` - Object of top left or right labels and parameters to hide original dicom labels.

Object example:

```
const labels = {
    topLeftLabels: [
        {
            // At which level labels are displayed. Level can be 'series' or 'study'.
            // If both are provided then series will be displayed.
            level: 'series',
            // Series uid because it is series level.
            uid: '1.2.840.113619.2.55.3.4271045733.996.1449464144.601',
            labels: ['series', 'label'],
            hideDicomLabels: false
        }
    ],
    topRightLabels: [
        {
            level: 'study',
            // Study uid because it is study level.
            uid: '1.2.840.113619.2.55.3.4271045733.996.1449464144.601',
            labels: ['patient', 'right'],
            hideDicomLabels: false
        }
    ]
}
```

### Set custom study label

```
viewerCommunication.setCustomStudyLabel(studyUid, label);
```

Parameters:

- `studyUid` - Study UID for which the label should be applied to.
- `label` - String to show in study header.

### Set additional custom study and series tags

```
viewerCommunication.setCustomTags(tags);
```

Parameter:

- `tags` - Array of objects of tags to add to study and series. Each object has level, uid and tags.

Object example:

```
const tags = [
    {
        // At which level labels are displayed. Level can be 'series' or 'study'.
        level: 'study',
        // Study uid because it is study level.
        uid: '1.2.840.113619.2.55.3.4271045733.996.1449464144.595',
        // Tags to add to study. Each tag has text and color properties
        tags: [{text: 'Leg', color: '#f6ff00'}, {text: 'Right knee', color: '#0d00ff'}, {text: 'Leg', color: '#d000ff'}]
    },
    {
        level: 'series',
        // Series uid because it is series level.
        uid: '1.2.840.113619.2.55.3.4271045733.996.1449464144.598',
        tags: [{text: 'Abdomen', color: '#40f616'}, {text: 'Right knee', color: '#0d00ff'}]
    }
]
```

### Generate instance MPR

```
viewerCommunication.generateInstanceMpr(containerId);
```

Parameter:

- `containerId` - viewport container id. If no container id is provided then active container is used.

### Change viewport orientation

```
viewerCommunication.changeViewportOrientation(containerId, orientation);
```

Parameters:

- `containerId` - viewport container id. If no container id is provided then active container is used.
- `orientation` - orientations: `CORONAL`, `AXIAL`, `SAGITTAL`.

### Create new measurement

```
viewerCommunication.createNewMeasurement(containerId, measurementData);
```

Parameters:

- `containerId` - Viewport container id in which measurement will be created.
- `measurementData` - Object with measurement data which will be created in viewport.

Measurement data object example:

```
const measurementData = {
    id: 'closed-polygon-id-1',
    type: 'closed-polygon',
    containerId: 'viewport-container-1-1',
    studyUid: 'study-uid-1',
    seriesUid: 'series-uid-1',
    instanceUid: 'instance-uid-1',
    colors: {
        regularColor: '#FFA500',
        activeColor: '#33CCFF',
        markedColor: '#009BFF',
        activeLabelColor: '#FFF'
    },
    data: {
        points: [
            [80, 113, 42],
            [221, 1, 42],
            [335, 132, 42],
            [178, 224, 42],
            [80, 113, 42]
        ],
        disabled: false
    }
};
```

### Delete measurement by id

```
viewerCommunication.deleteMeasurementById(measurementId);
```

Parameter:

- `measurementId` - Measurement id that has to be deleted.

### Update measurement by id

```
viewerCommunication.updateMeasurement(containerId, measurementData);
```

Parameters:

- `containerId` - Viewport container id in which measurement will be created.
- `measurementData` - Object with measurement data to use updating existing measurement.

### Select measurement to edit

```
viewerCommunication.selectMeasurementToEdit(containerId, measurementId, opts);
```

Parameters:

- `containerId` - Viewport container id in which measurement should be edited.
- `measurementId` - An ID of measurement to be selected.
- `opts` - Additional parameters depending on type of measurement being selected.

Opts data object example for `myocardium-roi-annotation`:

```
const opts = {
    toolId: 'repulsor',
    hoveredRegionIndex: 1
};
```

Parameter `toolId` can be one of three possible values: `repulsor`, `draw-region`, `fill-brush`. Parameter `hoveredRegionIndex` indicates which region from myocardium ROI is intended to be edited with repulsor. Parameter is ignored if other tools are activated.

### Change measurement display by id

```
viewerCommunication.changeMeasurementDisplayById(containerId, measurementId, opts);
```

Allows to alter the way how myocardium ROI is displayed (changing display properties for other measurement types is not supported). Parameters:

- `containerId` - Viewport container id in which measurement is visible.
- `measurementId` - An ID of measurement to be updated.
- `opts` - Additional parameters depending on type of measurement being updated.

Opts data object example for `myocardium-roi-annotation`:

```
const opts = {
    show: false,
    sendToFront: false
};
```

Parameter `show` is of boolean type. If true - respective myocardium ROI should be made visible. If false - it should be hidden. Parameter `sendToFront` is optional. If provided and its value is true, then respective myocardium ROI should be brought to top in Z-order.

### Initiate create measurement

```
viewerCommunication.initiateCreateMeasurement(containerId, measurementType);
```

Parameters:

- `containerId` - Viewport container id in which measurement will be created.
- `measurementType` - Identifies the type of measurement to initiate via this method. Currently is limited to `myocardium-roi-annotation` type only.

### Get list of available HP for study

```
viewerCommunication.getListOfAvailableHpForStudy();
```

### Apply hanging protocol

```
viewerCommunication.applyHangingProtocol(groupId, categoryId);
```

Parameters:

- `groupId` - HP group id.
- `categoryId` - HP category id from selected group.

### Apply previous hanging protocol category

```
viewerCommunication.applyPreviousHangingProtocolCategory();
```

### Apply next hanging protocol category

```
viewerCommunication.applyNextHangingProtocolCategory();
```

### Apply next hanging protocol comparison study (CP)

```
viewerCommunication.applyNextHangingProtocolCP();
```

## 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 viewports information event

```
const callback = (viewportsInformation) => console.log(viewportsInformation);
viewerCommunication.subscribeGetViewportsInformationEvent(callback);
```

Parameter:

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

### Unsubscribe get viewports information event

```
viewerCommunication.unsubscribeGetViewportsInformationEvent();
```

### Subscribe get instance metadata event

```
const callback = (viewportsInformation) => console.log(viewportsInformation);
viewerCommunication.subscribeGetInstanceMetadataEvent(callback);
```

Parameter:

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

### Unsubscribe get instance metadata event

```
viewerCommunication.unsubscribeGetInstanceMetadataEvent();
```

### 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 save started event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeAnnotationsSaveStartedEvent(callback);
```

Parameter:

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

### Unsubscribe annotations save started event

```
viewerCommunication.unsubscribeAnnotationsSaveStartedEvent();
```

### 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();
```

### Subscribe structure set edited event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeStructureSetEditedEvent(callback);
```

Parameter:

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

### Unsubscribe structure set edited event

```
viewerCommunication.unsubscribeStructureSetEditedEvent();
```

### Subscribe instance changed event

```
const callback = (viewportsInformation) => console.log(viewportsInformation);
viewerCommunication.subscribeInstanceChangedEvent(callback);
```

Parameter:

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

#### Unsubscribe instance changed event

```
viewerCommunication.unsubscribeInstanceChangedEvent();
```

#### Subscribe active container changed event

```
const callback = (activeContainerInformation) => console.log(activeContainerInformation);
viewerCommunication.subscribeActiveContainerChangedEvent(callback);
```

Parameter:

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

#### Unsubscribe active container changed event

```
viewerCommunication.unsubscribeActiveContainerChangedEvent();
```

#### Subscribe measurement created event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeMeasurementCreatedEvent(callback);
```

Parameter:

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

#### Unsubscribe measurement created event

```
viewerCommunication.unsubscribeMeasurementCreatedEvent();
```

#### Subscribe measurement updated event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeMeasurementUpdatedEvent(callback);
```

Parameter:

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

#### Unsubscribe measurement updated event

```
viewerCommunication.unsubscribeMeasurementUpdatedEvent();
```

#### Subscribe measurement deleted event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeMeasurementDeletedEvent(callback);
```

Parameter:

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

#### Unsubscribe measurement deleted event

```
viewerCommunication.unsubscribeMeasurementDeletedEvent();
```

#### Subscribe virtual series create event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeVirtualSeriesCreatedEvent(callback);
```

Parameter:

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

### Unsubscribe virtual series create event

```
viewerCommunication.unsubscribeVirtualSeriesCreatedEvent();
```

### Subscribe measurement marked event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeMeasurementMarkedEvent(callback);
```

Parameter:

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

### Unsubscribe measurement marked event

```
viewerCommunication.unsubscribeMeasurementMarkedEvent();
```

### Subscribe measurement unmarked event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeMeasurementUnmarkedEvent(callback);
```

Parameter:

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

### Unsubscribe measurement unmarked event

```
viewerCommunication.unsubscribeMeasurementUnmarkedEvent();
```

### Subscribe get list of available HP for study event

```
const callback = (data) => console.log(data);
viewerCommunication.subscribeGetListOfAvailableHpForStudy(callback);
```

Parameter:

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

### Unsubscribe get list of available HP for study event

```
viewerCommunication.unsubscribeGetListOfAvailableHpForStudy();
```

## Measurement coordinates conversion

To ensure correct measurement recreation from data, all our measurement related functions and events work or provide 3D coordinates in patient coordinate system. If you need to convert received 3D coordinate to instance 2D coordinate, you can use following function:

```
function get2DImagePositionFrom3D (position3d) {
    const imagePosition = this.getImagePosition();
    const imageOrientation = this.getImageOrientation();
    const pixelSpacing = this.getPixelSpacing();
    const x = ((position3d[0] - imagePosition[0]) * imageOrientation[0] + (position3d[1] - imagePosition[1]) * imageOrientation[1]
        + (position3d[2] - imagePosition[2]) * imageOrientation[2]) / pixelSpacing.x;
    const y = ((position3d[0] - imagePosition[0]) * imageOrientation[3] + (position3d[1] - imagePosition[1]) * imageOrientation[4]
        + (position3d[2] - imagePosition[2]) * imageOrientation[5]) / pixelSpacing.y;
    return {x, y};
}
```

If you need to convert 2D coordinate back to 3D coordinate, then you can use this function:

```
function get3DImagePositionFrom2D (position2d) {
    const imagePosition = this.getImagePosition();
    const imageOrientation = this.getImageOrientation();
    const pixelSpacing = this.getPixelSpacing();
    const x = imagePosition[0] + imageOrientation[0] * position2d.x * pixelSpacing.x
        + imageOrientation[3] * position2d.y * pixelSpacing.y;
    const y = imagePosition[1] + imageOrientation[1] * position2d.x * pixelSpacing.x
        + imageOrientation[4] * position2d.y * pixelSpacing.y;
    const z = imagePosition[2] + imageOrientation[2] * position2d.x * pixelSpacing.x
        + imageOrientation[5] * position2d.y * pixelSpacing.y;
    return [x, y, z];
}
```

# Change log

## 1.0.31 (2024-07-29)

### Changes

- Added functions `subscribeMeasurementMarkedEvent`, `subscribeMeasurementUnmarkedEvent` to listen on measurement mark events.
- Added function `subscribeVirtualSeriesCreatedEvent` to listen when virtual series are created.
- Added function `updateMeasurement` to enable update of existing measurement without destroying underlying object.
- Added function `initiateCreateMeasurement` to support creating myocardium ROI and activate default editing tool.
- Added function `selectMeasurementToEdit` to support either selecting closed_polygon or myocardium ROI objects for editing.
- Added function `changeMeasurementDisplayById` to allow show/hide myocardium ROI or rearrange myocardium ROIs on Z-axis.

## 1.0.30 (2024-04-16)

### Changes

- Fixed integration example issues related to latest MedDream Viewer changes.

## 1.0.29 (2024-03-06)

### Changes

- Added `applyHangingProtocol` function to set hanging protocol by group and category ids.

## 1.0.28 (2024-03-15)

### Changes

- Fix unsubscribe get list of available HP for study event description.

# Change log

## 1.0.27 (2024-03-06)

### Changes

- Added `getListOfAvailableHpForStudy` function to get available hanging protocol list for active study.

- Added `applyPreviousHangingProtocolCategory` function to apply previous available hanging protocol category.
- Added `applyNextHangingProtocolCategory` function to apply next available hanging protocol category.
- Added `applyNextHangingProtocolCP` function to apply next available hanging protocol comparison study (CP).

### 1.0.26 (2024-01-23)

#### Changes

- Updated `setCustomStudyLabel` function to set additional dicom tag labels for study or series level.
- Added `dicomTagLabels` argument to array of dicom tags.

#### Breaking changes

- Renamed `hideDicomLabels` argument to `hideOriginalLabels`.

## Change log

### 1.0.25 (2023-12-19)

#### Changes

- Updated `measurementData` object with `disabled` parameter which allows to disable measurement from editing.

### 1.0.24 (2023-12-15)

#### Changes

- Change log update.

### 1.0.23 (2023-12-14)

#### Changes

- Updated `updateButtonVisibility` example to include mpr mist oblique and key objects buttons.

### 1.0.22 (2023-10-19)

#### Changes

- Added `updateButtonVisibility` function to set which toolbar buttons are hidden.

### 1.0.21 (2023-10-04)

#### Changes

- Added `setCustomTags` function to set custom tags with tag text and color for study or series.

### 1.0.20 (2023-07-27)

#### Changes

- Added `setCustomStudyLabel` function to set additional custom label for study side panel by selected study uid.

### 1.0.19 (2023-07-21)

#### Changes

- Added `subscribeMeasurementDeletedEvent` function to subscribe of measurement deleted event callback.
- Added `unsubscribeMeasurementDeletedEvent` function to unsubscribe of measurement deleted event callback.

### 1.0.18 (2023-06-16)

#### Changes

- Added `getInstanceMetadata` function to get available instance metadata.
- Added `subscribeGetInstanceMetadataEvent` function to subscribe of get instance metadata event callback.
- Added `unsubscribeGetInstanceMetadataEvent` function to unsubscribe of get instance metadata event callback.

## 1.0.17 (2023-06-06)

### Changes

- Added `subscribeAnnotationsSaveStartedEvent` function to subscribe of annotations save started event callback.
- Added `unsubscribeAnnotationsSaveStartedEvent` function to unsubscribe of annotations save started event callback.
- Added `subscribeStructureSetEditedEvent` function to subscribe of structure set edited event callback.
- Added `unsubscribeStructureSetEditedEvent` function to unsubscribe of structure set edited event callback.

## 1.0.16 (2023-06-06)

### Changes

- Updated `setAdditionalInfoLabels` function to set additional info labels for study or series level.

## 1.0.15 (2023-06-05)

### Changes

- Added `createNewMeasurement` function to create new measurement in to viewport container.
- Added `deleteMeasurementById` function to delete requested measurement by container id.
- Added `subscribeInstanceChangedEvent` function to subscribe of instance changed event callback.
- Added `unsubscribeInstanceChangedEvent` function to unsubscribe of instance changed event callback.
- Added `subscribeActiveContainerChangedEvent` function to subscribe of active container changed event callback.
- Added `unsubscribeActiveContainerChangedEvent` function to unsubscribe of active container changed event callback.
- Added `subscribeMeasurementCreatedEvent` function to subscribe of measurement created event callback.
- Added `unsubscribeMeasurementCreatedEvent` function to unsubscribe of measurement created event callback.
- Added `subscribeMeasurementUpdatedEvent` function to subscribe of measurement updated event callback.
- Added `unsubscribeMeasurementUpdatedEvent` function to unsubscribe of measurement updated event callback.
- Added `getViewportsInformation` function to get available viewport's information.
- Added `subscribeGetViewportsInformationEvent` function to subscribe of get viewports information event callback.
- Added `unsubscribeGetViewportsInformationEvent` function to unsubscribe of get viewports information event callback.
- Added new `Measurement coordinates conversion` documentation section with information about coordinates conversion.

## 1.0.14 (2023-05-20)

### Changes

- Added `generateInstanceMpr` function to generate instance MPR.
- Added `changeViewportOrientation` function to change viewport orientation.

## 1.0.13 (2023-05-24)

### Changes

- Added `showInfoLabels` function to show/hide viewports info labels.
- Added `setAdditionalInfoLabels` function to set additional info labels for viewports.

## 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.

## 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`.