



## CLO API

Jaden

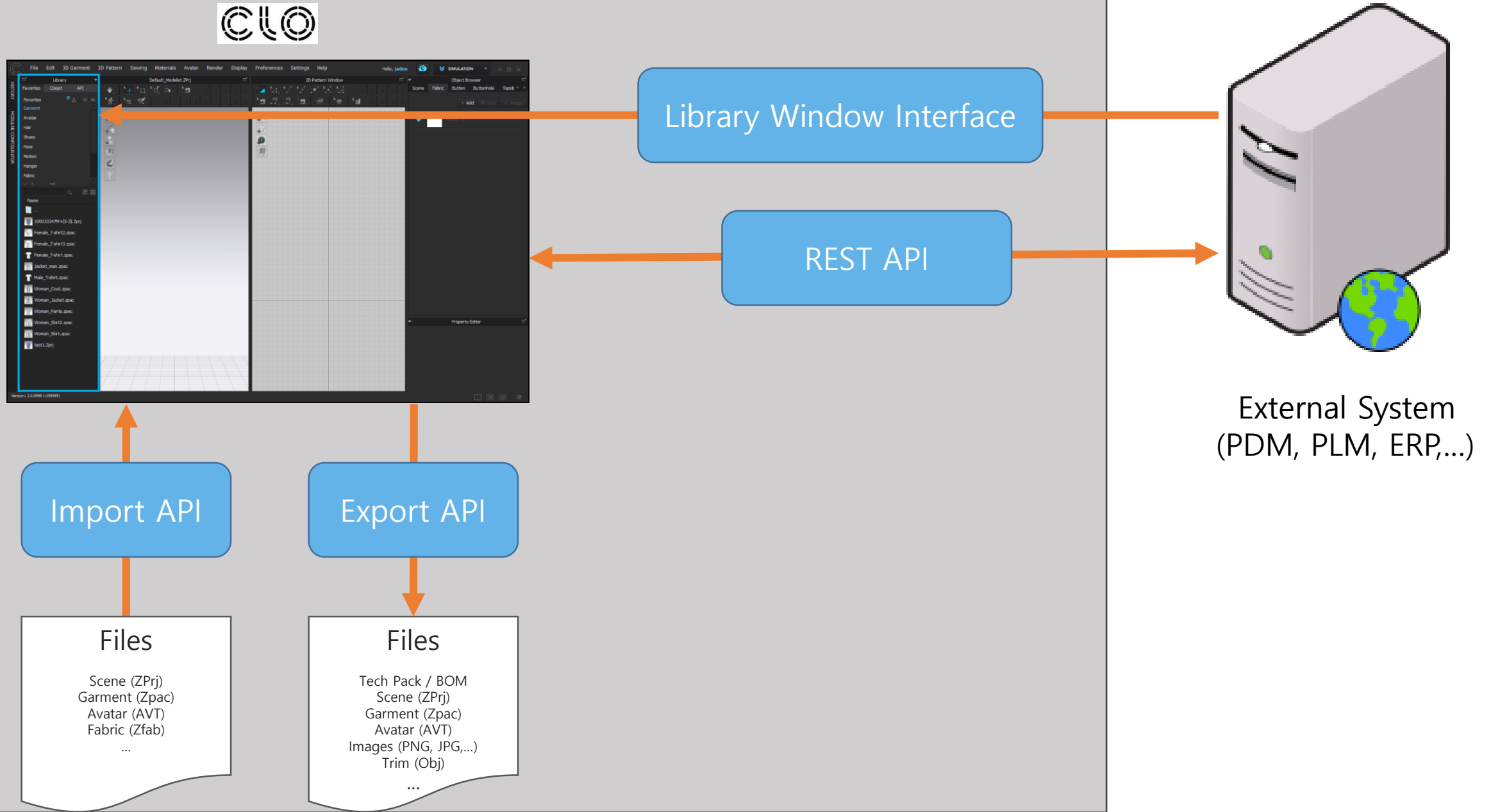
## ■ About CLO API

- For Both Windows and Mac
- C++
- The first version has been released with CLO v5.0
- Use cases
  1. For creating **PLUG-INS** to extend the functionality of CLO
  2. CLO Library Window Customization to transfer data from external systems like PDM/PLM
- Link to download API/SDK Package and Manual  
(<https://support.clo3d.com/hc/en-us/articles/360017616633-CLO-API-SDK-Guide>)



Overview

# CLO APIs



# Inbound – PLM to CLO



- File
- Thumbnail
- Meta data like Folder Hierarchy, Style No., Item ID,...

## Library Window Interface

API Developer needs to implement the body of each interface

```
virtual string GetItemList(const string& itemId, unsigned int pageNo, unsigned int pageSize, CLO_FINDER_SORT sort, bool bAscending, const string& searchText)
{
    return "";
}

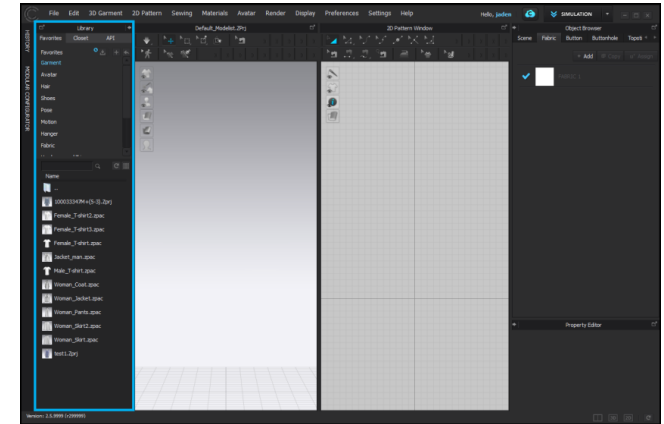
/*
*/
virtual string GetSearchItemList(unsigned int pageNo, unsigned int pageSize, CLO_FINDER_SORT sort, bool bAscending, const string& searchText)
{
    return "";
}

/*
@param parentFolderID The ID of the parent folder.
@return If any exceptions happen, this function should return "false". If the parent folder is "root", it should return "" as parentFolderID.
*/
virtual bool GetParentFolderID(const string& itemId, string& parentFolderID)
{
    return false;
}

/*
This function is called when starting paging from CLO SW.

@param itemId The ID of a file item
@param sizeInByte The size of returned thumbnail image data in bytes
@return Thumbnail image data in byte array. The data should be given in PNG format.
*/
virtual CLO_BYTE* GetPNGThumbnail(const string& itemId, unsigned int& sizeInByte)
{
    return NULL;
}

/*
will be supported in v5.1 or above
*/
virtual string GetMetaData(const string& itemId)
```



Add DLL

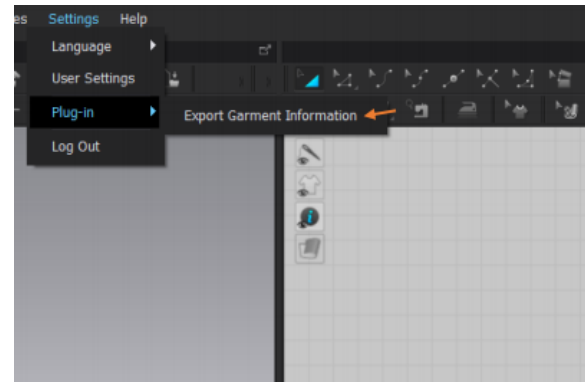
# Outbound – CLO to PLM

API Developer needs to make a **PLUG-IN**.

```
/*  
Export DXF  
@return Output file path. If an error occurs, return empty string. If filePath parameter is not given, output files will be created  
*/  
MV_CLO_SCENE_API string ExportDXF();  
MV_CLO_SCENE_API string ExportDXF(const string& filePath);  
  
/*  
Export Tech Pack data in json file and associated image files.  
@param filePath Should be given in ".json" format  
@param bWithCaptureImage Set as true to save with associated images.  
*/  
MV_CLO_SCENE_API void ExportTechPack(const string& filePath, bool bWithCaptureImage);  
  
/*  
Export thumbnail of the current scene  
@return Output file path. If an error occurs, return empty string. If filePath parameter is not given, output files will be created  
*/  
MV_CLO_SCENE_API string ExportThumbnail3D();  
MV_CLO_SCENE_API string ExportThumbnail3D(const string& filePath);  
  
/*  
Export snapshot images. This function displays the same dialog as CLO so that users can configure the snapshots.  
If user turns on the option "Save Separate Images", then series of images will be saved with the name followed by the postfix "_01".  
@return Return the list of the path of output files per colorway. The files are saved in the format of "filePath_colorwayIndex_imageIndex".  
*/  
MV_CLO_SCENE_API vector< vector< string>>> ExportSnapshot3D(const string& filePath);  
MV_CLO_SCENE_API vector< vector< string>>> ExportSnapshot3D();  
  
/*  
Export Rendering Image.  
@param bRenderAllColorways If true, output the images for all colorways. Otherwise, it returns the images for the current colorway.  
@param savedFilePathList Return the list of the path of output files per colorway. The files are saved in the format of "filePath_colorwayIndex_imageIndex".  
@param colorwayIndex It starts with 0 index. It must not be over the total colorway count.  
*/  
MV_CLO_SCENE_API vector< vector< string>>> ExportRenderingImage(const string& filePath, bool bRenderAllColorways, vector< string>& savedFilePathList, int colorwayIndex);  
MV_CLO_SCENE_API vector< vector< string>>> ExportRenderingImage(const string& filePath, bool bRenderAllColorways);  
MV_CLO_SCENE_API vector< string> ExportSingleColorwayRenderingImage(const string& filePath, unsigned int colorwayIndex);  
MV_CLO_SCENE_API vector< string> ExportSingleColorwayRenderingImage(unsigned int colorwayIndex);
```

Export API

REST API



Add DLL

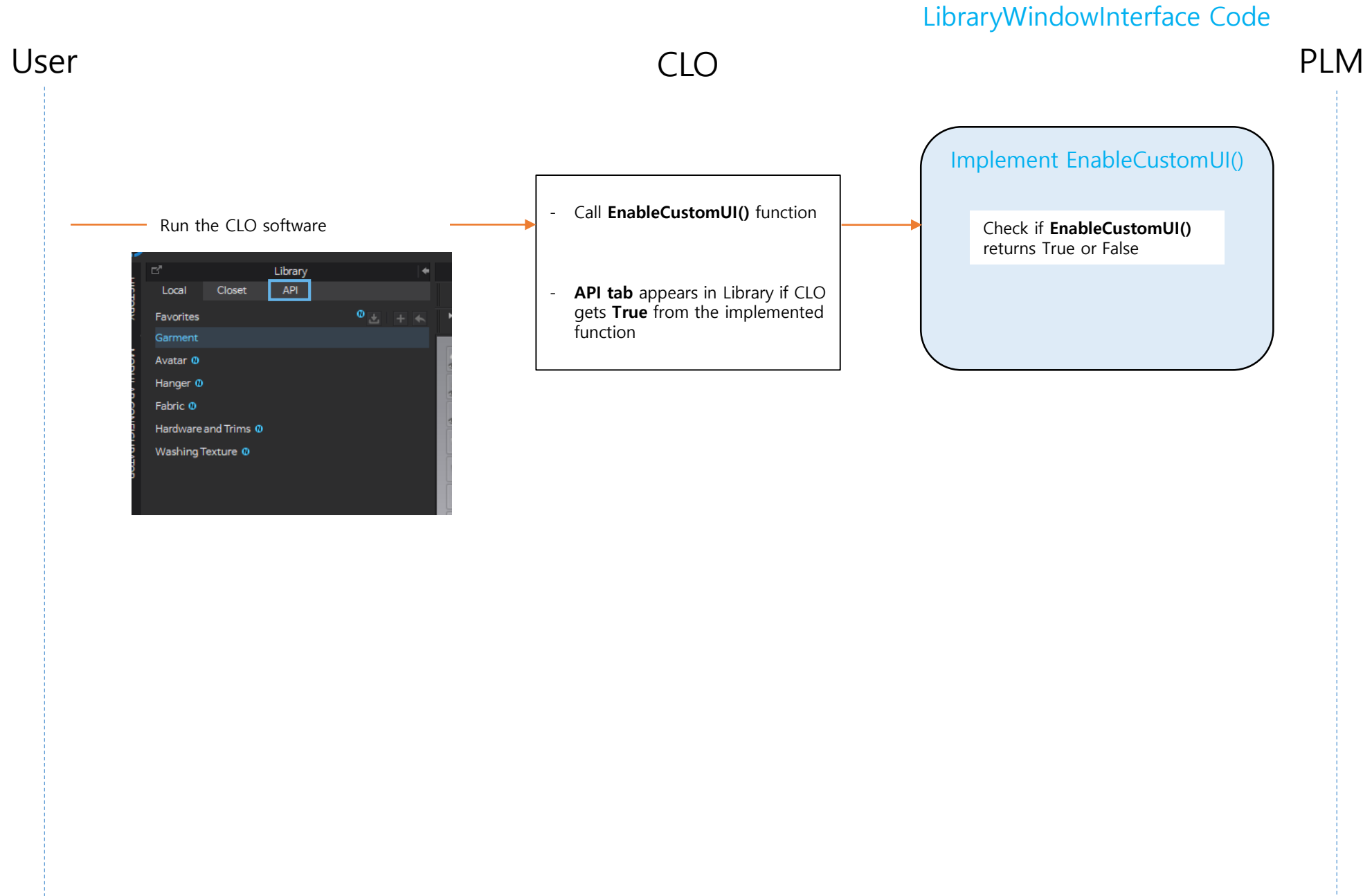
- Snapshots
- Rendering Images
- Tech Pack /BOM
- Files (Zpac, OBJ, ...)
- ...





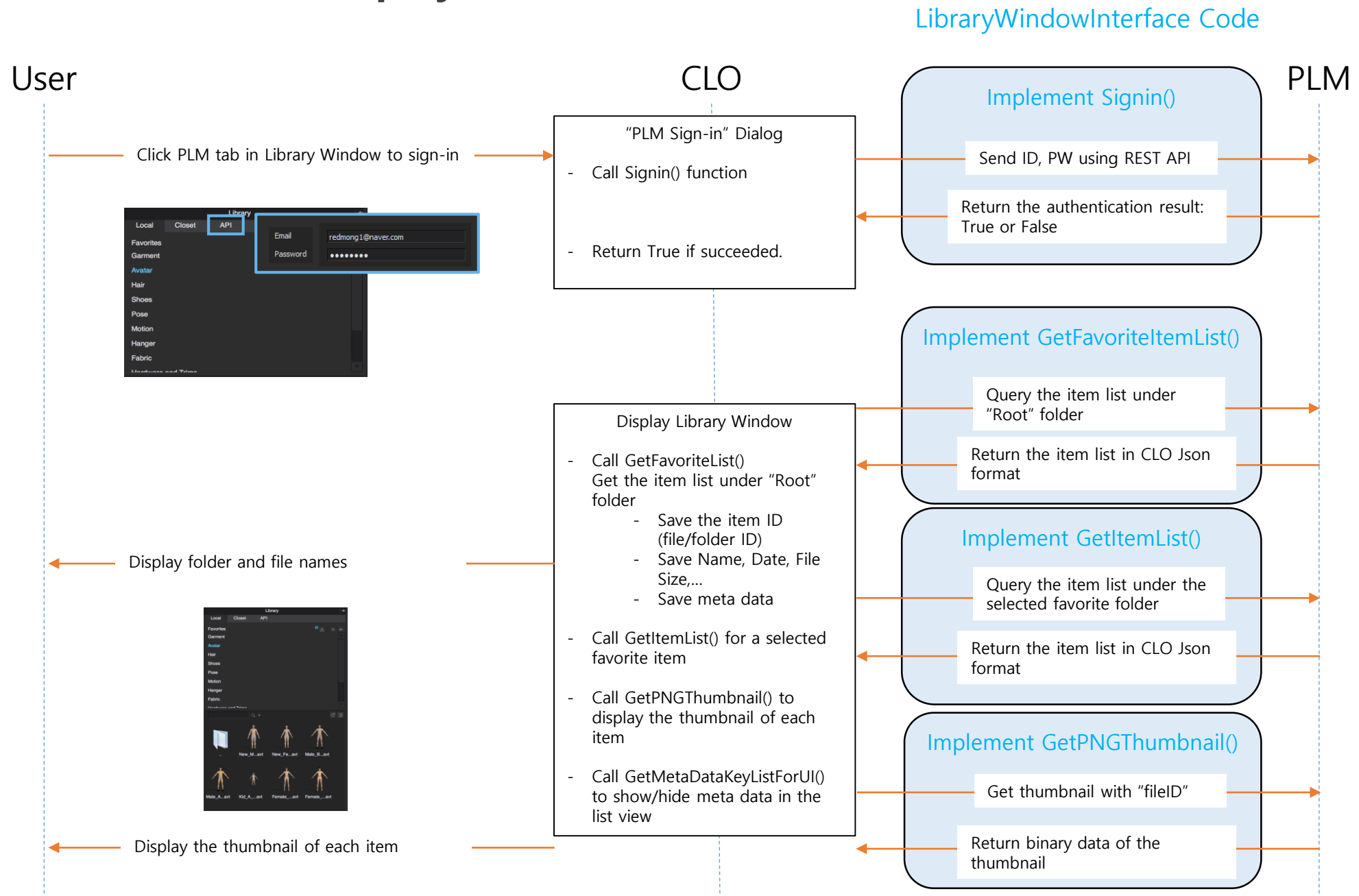
Workflow

# 0. Enable "API" tab in Library Window

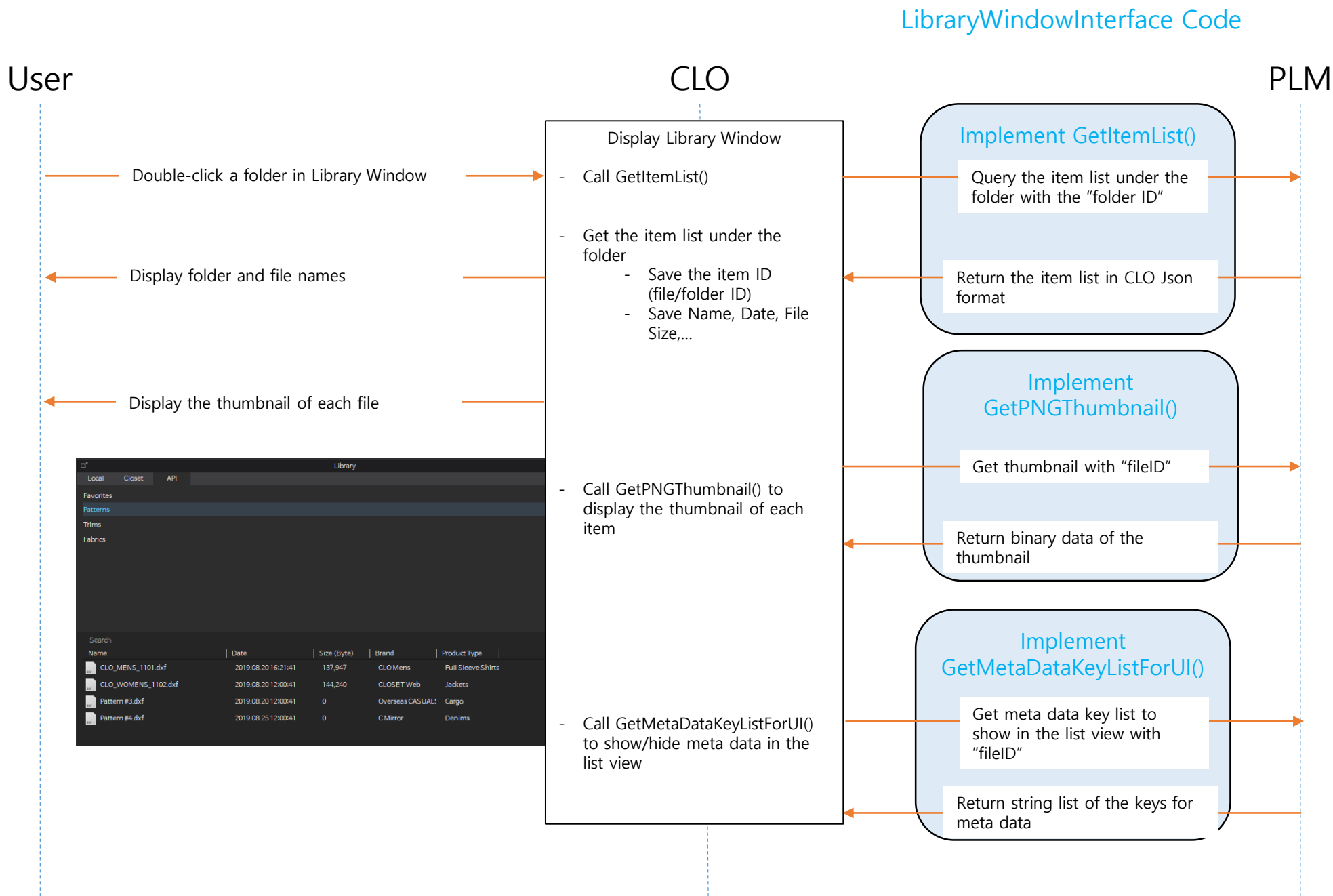




# 1. Sign-in to PLM and Display "Root" folder



# 2. Browse PLM folder



# 3. Customizing Preview Dialog

LibraryWindowInterface Code

User

CLO

PLM

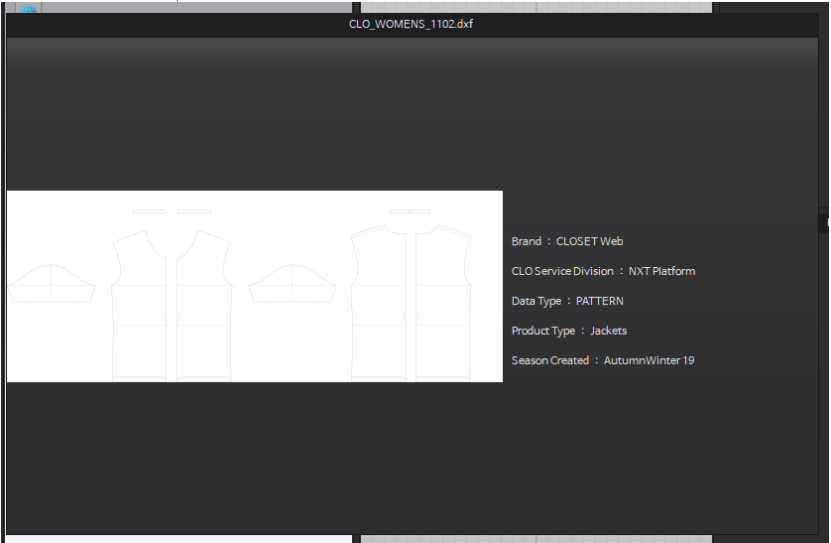
- Mouse on-over an item on the Library Window
- Display the preview dialog for the item with the preview image and meta data list

Construct Preview Dialog

- Create FinderViewPreView dialog
- Parse meta data list from the item which were saved via GetItemList()
- Call GetPreviewImage() to get the preview image on the preview dialog.
- Display the FinderViewPreview dialog

Implement GetPreviewImage()

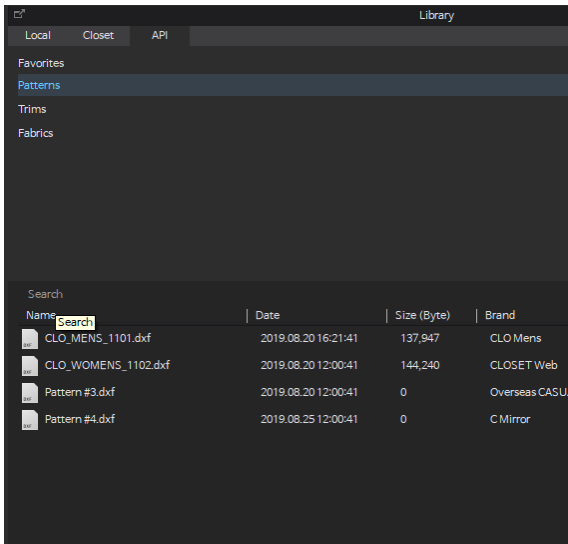
- Get preview image with "fileID"
- Return binary data of the thumbnail



# 4. Search Items

User

Click the 'Search' button on the middle of Library Window



Display the result item

CLO

Display Library Window

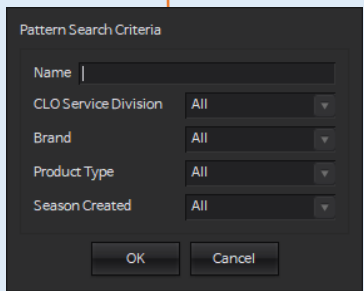
- Call GetSearchItemList()
- Get the item list under the folder
  - Save the item ID (file/folder ID)
  - Save Name, Date, File Size,...
- Call GetPNGThumbnail() to display the thumbnail of each item

LibraryWindowInterface Code

PLM

Implement GetSearchItemList()

Display search dialog and get the search keywords



Query the search keywords to PLM

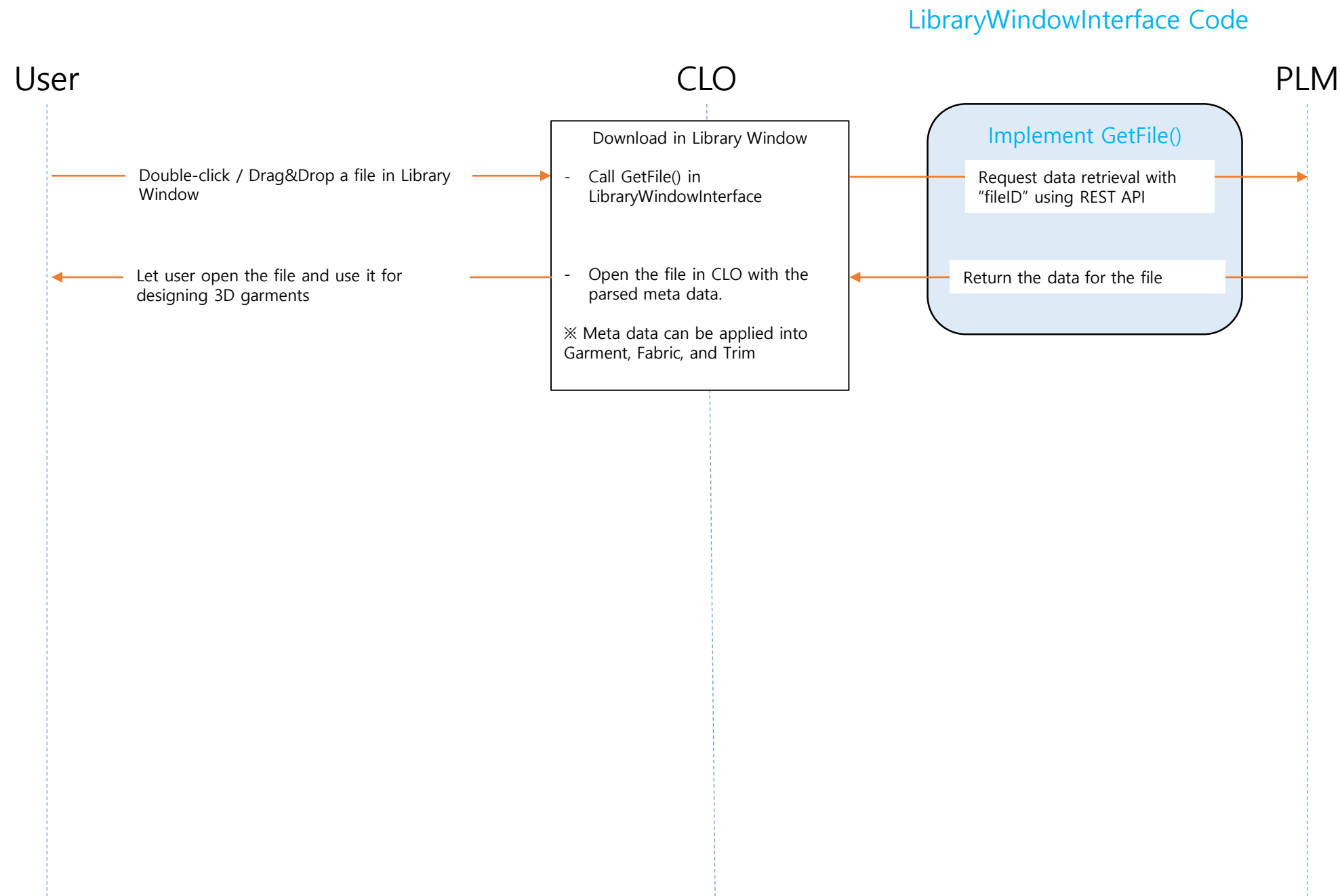
Return the item list in CLO Json format

Implement GetPNGThumbnail()

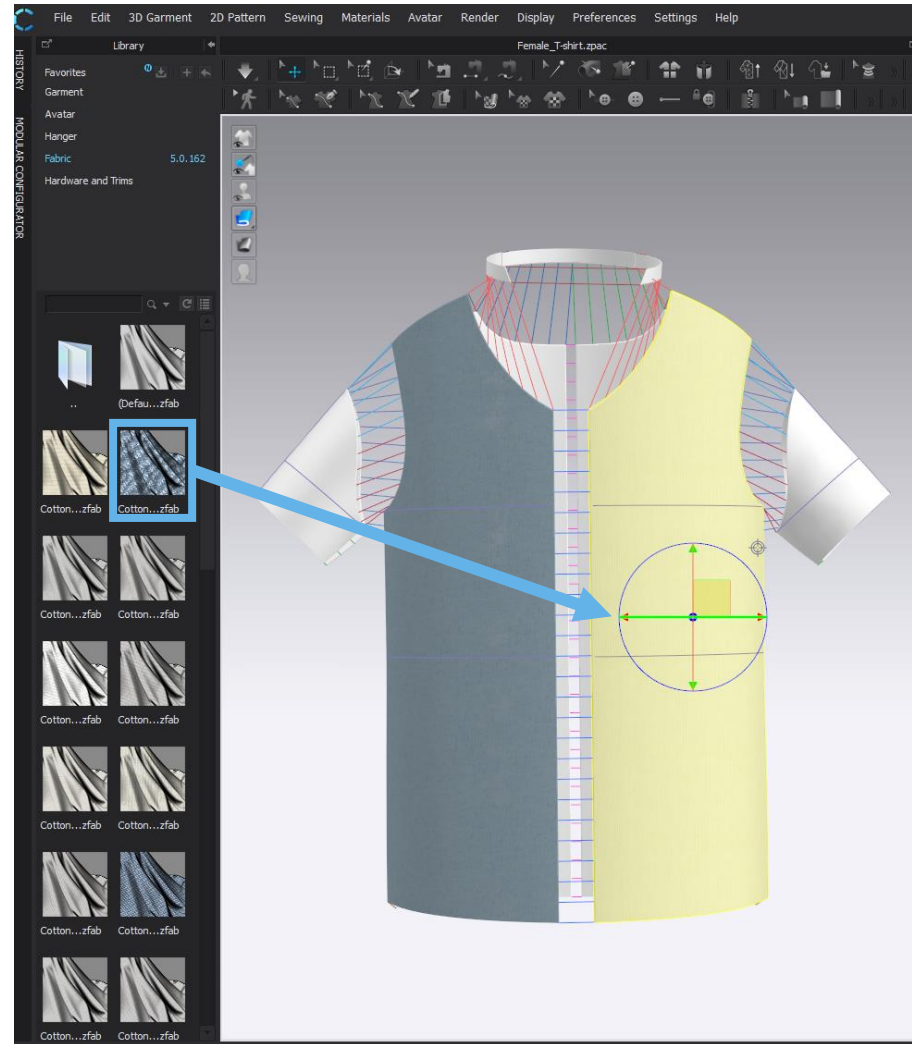
Get thumbnail with "fileID"

Return binary data of the thumbnail

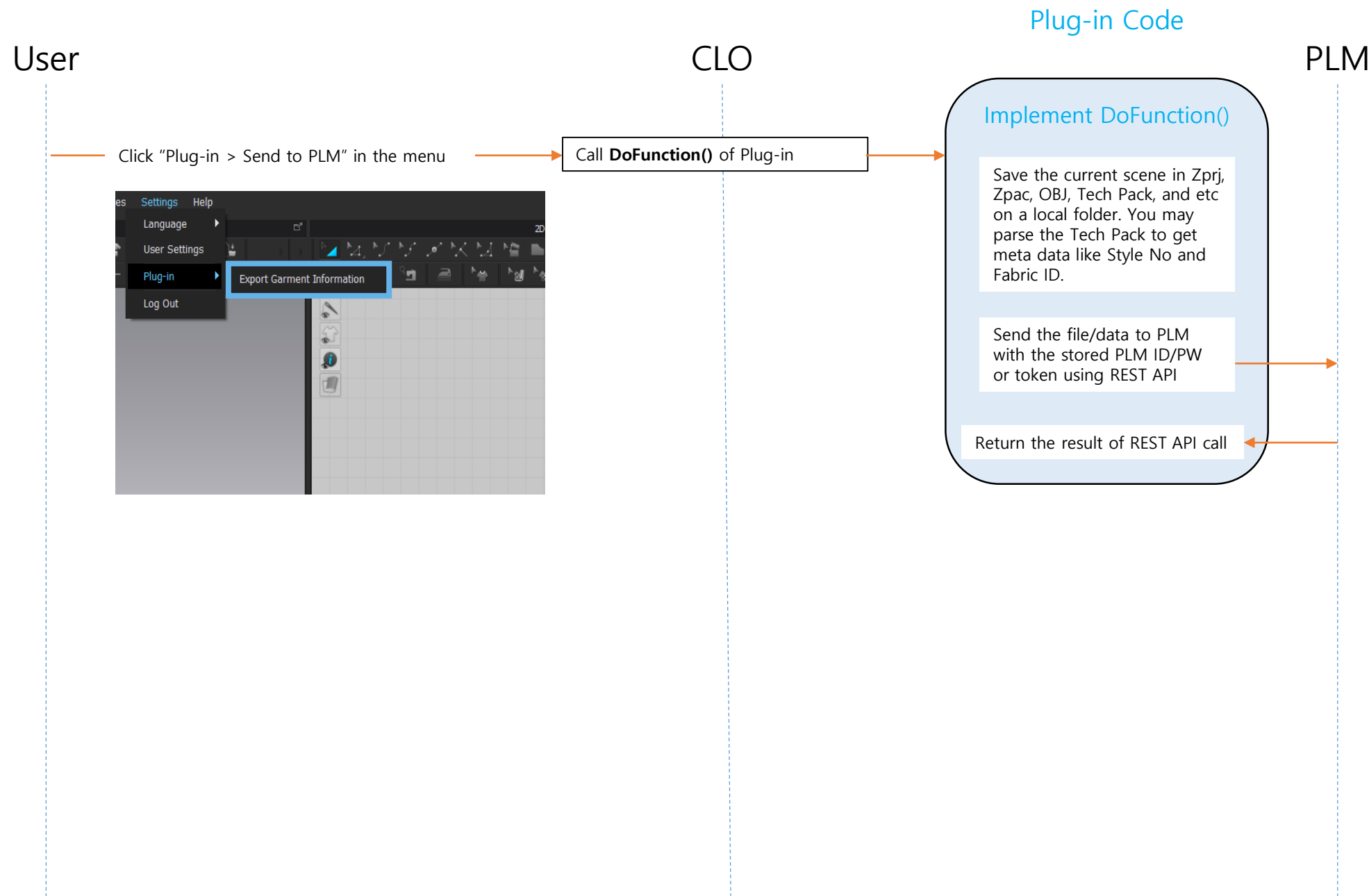
# 5. Get PLM file/data



## 6. Design garment



# 7. Send CLO file/data to PLM



## 8. Analyze and Save data on PLM

JSON Editor Online

```
1 {
2   "version": "130", // means v1.3
3
4   "zpacPath": "output.ZPac", // [v1.1] zpac file path (this file is created only via CLO API)
5   "zprjPath": "output.ZPrj", // [v1.1] zprj file path (this file is created only via CLO API) -> [v1.2] "projectFile"
6   "zrestPath": "output.zrest", // [v1.0] zrest file path (this file is created only for CLOSET). -> [v1.1] absolute
  relative path
7
8   "lengthOfSeamLines": 355.535, // [v1.0] unit is inch. Total length of seam lines to estimate how much sewing thread
  consumed.
9   "numberOfPatterns": 30, // [v1.0] total number of patterns
10  "areaOfPatterns": 0.89348, // [v1.0] unit is m^2. The consumed area of whole patterns.
11  "fitting": 0, // [v1.0] for BenefitByCLO API (0: impossible, 1: possible)
12  "thumbnail": [ // [v1.0]
13    "thumbnail_o.png", // front thumbnail
14    "thumbnail_l.png", // left thumbnail
15    "thumbnail_b.png" // back thumbnail
16  ],
17  "patternLayoutThumbnail": "patternLayoutThumbnail.png", // [v1.3]
18  // "colorway": [], // [v1.0] deprecated since v1.2
19  "currentColorwayIndex": 0, // [v1.2]
20  "colorwayList": [ // [v1.2]
21    {
22      "id": 3814,
23      "name": "colorway 0",
24      "thumbnail": [ // colorway thumbnail front, side, back in order.
25        "colorway0_thumbnail0.png", // front
26        "colorway0_thumbnail1.png", // side
27        "colorway0_thumbnail2.png" // back
28      ],
29      "thumbnailWithAvatar": [ // colorway thumbnail with avatar
30        "colorway0_thumbnailWithAvatar0.png", // front
31        "colorway0_thumbnailWithAvatar1.png", // side
32        "colorway0_thumbnailWithAvatar2.png" // back
33      ]
34    },
35  ],
36  "modularMark": 0, // [v1.2] to show if modular is used or not (0: not used, 1: used) (for CLO API)
37  "modularTemplateFileName": "Jacket.Double.zmdr", // [v1.2] modular structure file name (for CLO API), optional
38  "modularBlockList": [ // [v1.2], optional.
39    {
40      "id": 3813,
41      "name": "Pattern2D_4127611",
42      "type": "Body",
43      "position": "Back",
44      "singleDual": "Single",
45      "tag": "",
46      "zblcName": "Body_B.DoubleVent",
47    },
48  ],
49  // todo : sewingList
50  "patternList": [ // [v1.2]
```

Tech Pack (.json) sent to PLM contains all information

### - Data brought from PLM

- Style No
- Fabric, Trim ID
- Organization data (Brand, Season, Line,...)

\* Based on this information, you can decide on where to store the file (send from CLO) in PLM

\* Meta Data sections for Garment(Style), Fabric, Trim in Tech Pack (.json) file ( **New** )

### - Data created in CLO

- Thumbnail of each item(pattern, fabric, trim, graphic, colorway)
- The number of each item
- Information about each item
  - Fabric consumption, button weight
  - How many each fabric/trim/graphic is used
  - ...



**Blue part** : should be implemented by your developers – Plug-in,  
LibraryWindowInterface

