3D Viewer

Dependencies:

Required:

- 3D Viewer Field (component plugin)
    - Required for the field to be available in the UI.

Optional:

- 3D Viewer Connected System (connected system plugin)
    - Optional, allows for loading models from Appian.
    - Without this plugin, the field can only load models by drag-and-drop or picking a local file.
- 3D Viewer Function (function plugin)
    - Optional, allows for loading models from Appian.
    - Without this plugin, the field can only load models by drag-and-drop or picking a local file.

Configuration:

1. Add the 3D Viewer Field (component plugin) to your environment.
2. Call the field in an interface: `threeDViewerField()`.
3. If you only need to load models by drag-and-drop or picking a local file, you're done. **If you want to be able to load models directly from appian**, continue to the next section.

Loading models from Appian

Loading models from appian requires that you have the 3D Viewer Connected System (connected system plugin) and the 3D Viewer Function (function plugin) installed in your environment. You will also need to set up a third-party credential in Appian. For an explanation on why, see the "Additional Comments" section below.

1. Add the 3D Viewer Connected System (connected system plugin) to your environment.
2. Add the 3D Viewer Function (function plugin) to your environment.
3. Set up the Encryption Key:
    1. In the *Admin Console*, go to *Third-Party Credentials*.
    2. Click *Create* to add a new credential.
    3. Under the *Identity* section, enter a name for the credential. Record the "Key" that is generated below the name for later use. This is typically an all lower-case version of the name.
    4. Under the *Credentials* section, click *+Add Field*.
    5. Enter a *Field name* and *Value* for the credential. Record the generated *Key* field.
        - Note that the *Value* will be used as the encryption key. It is highly recommended that you use a strong password and *Mask* the value.
    6. Under the *Plug-ins List* section, add the following plugins:
        - 3D Viewer Connected System
        - 3D Viewer Function
4. Set up the Connected System:
    1. Create a new Connected System.
        1. In the list of Connected System types, choose **3D Viewer Connected System**.
        2. Enter the *Value* of the encryption key you created (and hopefully masked) in step 3.v. above.
        3. Save the connected system.
5. Create SAIL Constants:
    1. Create a constant of type *Text*, containing the Third-Party Credential key from step 3.iii. above.
        - Note that this is the Key under the *Identity* section, **not** the Key under the *Credentials* section.
    2. Create a constant of type *Text*, containing the key for the field that holds the encryption key value.
        - This is the *generated Key field* from step 3.v. above. This is **not** the masked/unmasked *value* of that field.
    3. Create a constant of type *Connected System*, containing the connected system you created in step 4.
6. Configure the `threeDViewerField()` to load documents from Appian:
    1. In your interface, call the `threeDViewerField()`.

2. Add your docId as the `docId` parameter.
3. Add your connected system constant as the `threeDViewerConnectedSystem` parameter.
4. Add loggedInUser() as the `plainTextUsername` parameter.
    1. Add the following function call for the `encryptedUsername` parameter:

        ```
        encryptusernamewithhmac(
             username: user(loggedInUser(), "username"),
             scsKey: ,
             scsField:
        )
        ```

That's it! You should now be able to load models from Appian. Here is an example code snippet:

```
threeDViewerField(
  label: if(
    a!isNotNullOrEmpty(ri!docId),
    document(ri!docId, "name") & "." & document(ri!docId, "extension"),
    null
  ),
  labelPosition: if(
    a!isNotNullOrEmpty(ri!docId),
    "ABOVE",
    "COLLAPSED"
  ),
  threeDViewerConnectedSystem: cons!PD_CS_THREE_D_VIEWER,
  docId: ri!docId,
  plainTextUsername: user(loggedInUser(), "username"),
  encryptedUsername: encryptusernamewithhmac(
    username: user(loggedInUser(), "username"),
    scsKey: cons!PD_SCS_THREE_D_VIEWER_KEY,
    scsField: cons!PD_SCS_THREE_D_VIEWER_FIELD_NAME
  ),
  onMeshSelected: ri!selectedMesh
)
```

Additional Comments:

Appian currently does not have a way to get the user's context (or their session) from a connected system via the java api. Passing the username via loggedInUser() from SAIL to the component plugin allows the component plugin to provide this username to the connected system and in turn allows the connected system to enforce folder and document security. However, an advanced user could modify this username in their browser since component plugins reside client-side. To account for this, the `encryptusernamewithhmac()` function plugin provides a way to encrypt the username with a secret key that is stored in a third-party credential. The connected system uses the same secret key. The connected system re-encrypts the plain text username server-side and verifies that it matches the encrypted value. This ensures that the username has not been modified by the client.