Aras CAD Converter Installation

The following steps outline the process of installing Aras CAD Converter:

1. Download the **Aras 3D Visualization 35 CD Image** from Aras File Sharing site.
2. Unzip the Aras 3D Visualization 35 CD Image on the local machine, in its own folder.
3. Create a folder with subfolders in the local work repository:
   `GIT_REPO_PATH\CodeTree\ConversionServer\customExtensions\Hoops\`
4. Copy the HOOPS Communicator Linux package (i.e. `HOOPS_Communicator_2025.4.0_Linux.tar.gz`) into the Hoops folder created above.
5. Copy the .sh" script:
   `GIT_REPO_PATH\AutomatedProcedures\tools\Aras_PDS_SaaS2\tools\ThirdPartyDependencies\onbuild.sh`
   into the folder: `GIT_REPO_PATH\CodeTree\ConversionServer\customExtensions\`
6. Copy the .sh" script:
   `GIT_REPO_PATH\AutomatedProcedures\tools\Aras_PDS_SaaS2\tools\ThirdPartyDependencies\installHoops.sh`
   into the folder: `GIT_REPO_PATH\CodeTree\ConversionServer\customExtensions\Hoops\`
7. Make sure that the copied file `installHoops.sh` contains the correct name and version number of the Hoops Communicator package:
   - Open the file `installHoops.sh` with a text editor that supports Unix line endings (e.g. VIM, Notepad++, VS Code, or other).
   - Change the INSTALLATION_PACKAGE_NAME to the required Hoops Package:
     `INSTALLATION_PACKAGE_NAME="HOOPS_Communicator_2025.4.0_Linux.tar.gz"`
   - Change the INSTALLATION_PACKAGE_VERSION to the correct version:
     `INSTALLATION_PACKAGE_HOOPS_VERSION="HOOPS_Communicator_2025.4.0"`
   - Ensure all script line endings are Unix line endings (LF).
   - Save the file.
8. If HOOPS templates are required, create a folder:
   `GIT_REPO_PATH\CodeTree\ConversionServer\customExtensions\Hoops\HoopsConverterTemplates\`
   `local Work.git repository`.
9. Add the template files.
10. Navigate to the `\Packages\CADConverter\ConversionServer\` folder of the Aras 3D Visualization 35 CD Image.
11. Copy the “bin” folder into the `GIT_REPO_PATH\CodeTree\ConversionServer\` folder of the local Work.git repository.
12. Add a configuration transformation file for the `ConversionServerConfig.xml` in the `\TransformationsOfConfigFiles\` folder of the Work.git repository with the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns:xdt="<bsp-code-inline>http://schemas.microsoft.com/XML-Document-Transform</bsp-code-inline>">
<configSections xdt:Transform="Replace" xdt:Locator="XPath(/configuration/configSections)">
<section name="oauth" type="Aras.OAuth.Configuration.OAuthSection, Aras.OAuth.Configuration" />
<!-- Common converter service configuration -->
```

```
 <section name="ConversionServer"
type="Aras.ConversionFramework.ConversionServer.Configuration.ConversionServ
Conversion.Base" />
 <sectionGroup name="ConverterSettings">
 <section name="ArasCadConverter"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterC
ArasCadConverter" />
 <section name="ArasCadConverterPrc"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterC
ArasCadConverter" />
 <section name="DpnCadConverterStepJt"
type="Aras.ConversionFramework.Converter.Hoops.Configuration.HoopsConverterC
ArasCadConverter" />
 </sectionGroup>
 </configSections>
 <ConversionServer>
 <Converters>
 <Converter name="Aras CAD to PDF Converter"
type="Aras.ConversionFramework.Converter.Hoops.HoopsConverter,
ArasCadConverter" xdt:Transform="Insert" />
 <Converter name="Aras PRC to SCS Converter"
type="Aras.ConversionFramework.Converter.Hoops.HoopsConverterPrc,
ArasCadConverter" xdt:Transform="Insert" />
 <Converter name="JT Step CAD Converter"
type="Aras.CadConverter.StepJtCadConverter, ArasCadConverter"
xdt:Transform="Insert"/>
 </Converters>
 </ConversionServer>
 <ConverterSettings xdt:Transform="Replace">
 <ArasCadConverter>
 <Application converterPath="/usr/bin/xvfb-run"/>
 <Command arguments="'--auto-servernum' '-s' '-screen 0 640x480x24'
/opt/ts3d/bin/linux64/converter --sc_compute_bounding_boxes 'All' --
```

```
load all configurations 'True' --input pdf template file '/app/HOOPS
Converter/Templates/Blank Template L.pdf' --output pdf
'%filepath%/%filename%.pdf' --output png '%filepath%/%filename%.png' -
-output png resolution '150x150' --output scs
'%filepath%/%filename%.scs' --output xml assemblytree
'%filepath%/%filename%.xml' --output prc '%filepath%/%filename%.prc' -
-background color '1.0, 1.0, 1.0' --output_logfile
'%filepath%/%filename%.log'" />
 <AssemblyCommand dynamicEnabled="True" arguments="'--auto-servernum'
'-s' '-screen 0 640x480x24' /opt/ts3d/bin/linux64/converter --
load all configurations 'True' --input pdf template file '/app/HOOPS
Converter/Templates/Blank Template L.pdf' --output pdf
'%filepath%/%filename%.pdf' --output png '%filepath%/%filename%.png' -
-output png resolution '150x150' --output scs
'%filepath%/%filename%.scs' --output xml assemblytree
'%filepath%/%filename%.xml' --output prc '%filepath%/%filename%.prc' -
-background color '1.0, 1.0, 1.0' --output_logfile
'%filepath%/%filename%.log'" />
 <Output>
 <UploadToVault>
 <File extension="prc" argsMarkers="--output prc"/>
 <File extension="scs" argsMarkers="--output scs"/>
 <File extension="pdf" argsMarkers="--output pdf"/>
 <File extension="png" argsMarkers="--output png"/>
 <File extension="stl" argsMarkers="--output stl"/>
 <File extension="xml" argsMarkers="--output_xml_assemblytree"/>
 </UploadToVault>
 </Output>
 </ArasCadConverter>
 <ArasCadConverterPrc>
 <Application converterPath="/usr/bin/xvfb-run"/>
 <Command arguments="'--auto-servernum' '-s' '-screen 0 640x480x24'
```

```
/opt/ts3d/bin/linux64/converter --sc compute_bounding_boxes 'All' --
load all configurations 'True' --output scs
'%filepath%/%filename%.scs' --output xml assemblytree
'%filepath%/%filename%.xml' --output_logfile
'%filepath%/%filename%.log'" />
 <Output>
 <UploadToVault>
 <File extension="prc" argsMarkers="--output prc"/>
 <File extension="scs" argsMarkers="--output scs"/>
 <File extension="pdf" argsMarkers="--output pdf"/>
 <File extension="png" argsMarkers="--output png"/>
 <File extension="stl" argsMarkers="--output stl"/>
 <File extension="xml" argsMarkers="--output_xml_assemblytree"/>
 </UploadToVault>

 </Output>
 </ArasCadConverterPrc>
 <DpnCadConverterStepJt>
 <Application converterPath="/opt/exchange/bin/linux64"/>
 <Parameters>
 <Parameter key="JTVersion" value="8.1" /> <!-- Optional. Possible
values: 8.1|9.5|10.0 -->
 </Parameters>
 </DpnCadConverterStepJt>

 </ConverterSettings>
 </configuration>
```

Important

The **ConversionServerConfig.xml** is available from the Baselines Artifact feed of the SDE for reference when building configuration transformation files.

13. Copy the folders 3d, common, ArasCadToPdfConverter, com, and PLM from the Aras 3D Visualization 35 CD `Image\Packages\CADConverter\Imports\` folder into the AML-packages folder of the Work.git repository.

Important

Do not copy the **imports.mf** file, as it will overwrite the existing **imports.mf** file in the Work.git repository.

14. Open the **imports.mf** file.
15. Copy the following `<package>` elements:

```
<imports>
<package name="hoops converter" path="ArasCadToPdfConverter">
<dependson name="com.aras.innovator.conversion" />
<dependson name="3d_common" />
</package>
<package name="com.aras.innovator.cui_default" path=".\" />
<package name="com.aras.innovator.solution.PLM" path="PLM" />
</imports>
```

16. Paste the `<package>` elements into the **imports.mf** file of the Work.git repository.
17. Open the file Aras 3D Visualization 35 CD `Image\Packages\CADConverter\Imports\3d_common.mf`
18. Copy the following <package> element:

```
<imports>
<package name="3d_common" path="3d_common" />
</imports>
```

19. Paste the `<package>` element into the **imports.mf** file of the Work.git repository, at the top of the list of packages to import:

```
<imports>
<package name="3d common" path="3d common" />
<package name="hoops converter" path="ArasCadToPdfConverter">
<dependson name="com.aras.innovator.conversion" />
<dependson name="3d_common" />
</package>
<package name="com.aras.innovator.cui default" path=".\" />
<package name="com.aras.innovator.solution.PLM" path="PLM" />
</imports>
```

20. Stage and review the changes to ensure that no customizations are overwritten.
21. Commit the changes.
22. Run the “**continuous-integration**” pipeline.
23. Run the “**deploy-innovator**” pipeline to complete the installation.