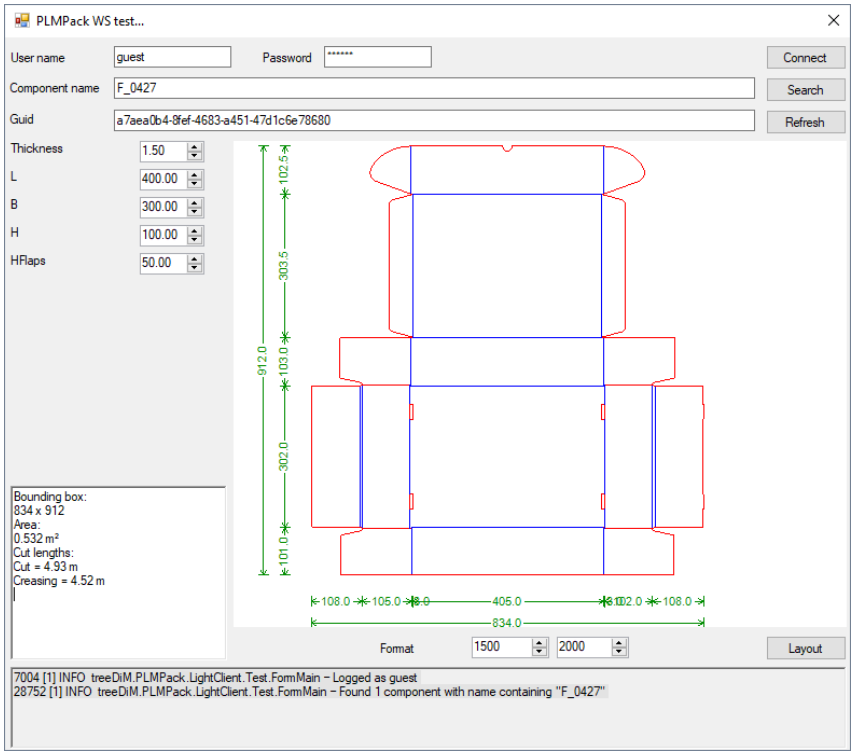
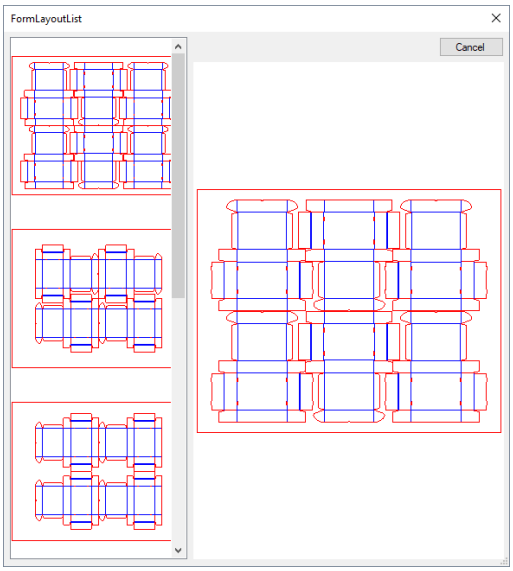
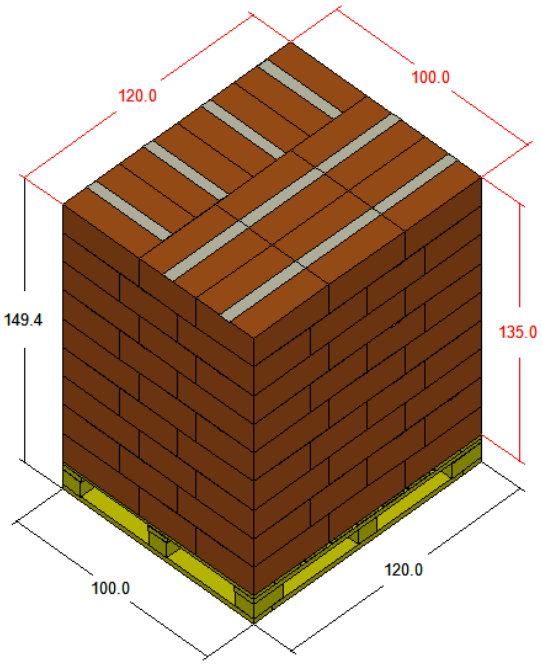
How to consume treeDiM PLMPack WCF services







Contents

Goal:

This is a summary of steps needed to consume the treeDiM PLMPack service hosted on Azure.

Prerequisites:

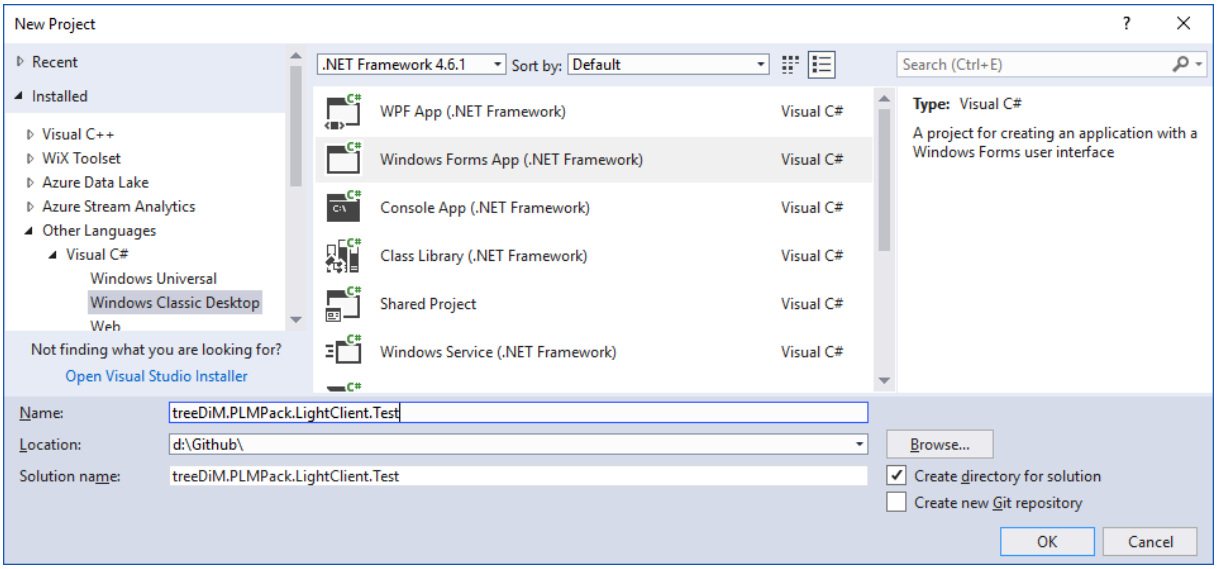
WCF web services are known to be consumable by a wide variety of environment/languages but for the sake of simplicity, we will be assuming the following development environment:

- Windows 10,
- Visual Studio 2017 (Community edition is freely downloadable here)

Referring the WCF service:

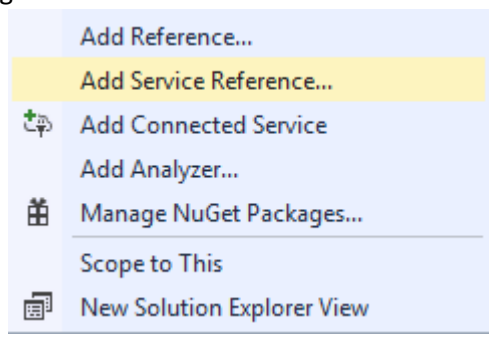
Create a new project in Visual Studio

Here we are creating a simple windows forms application.



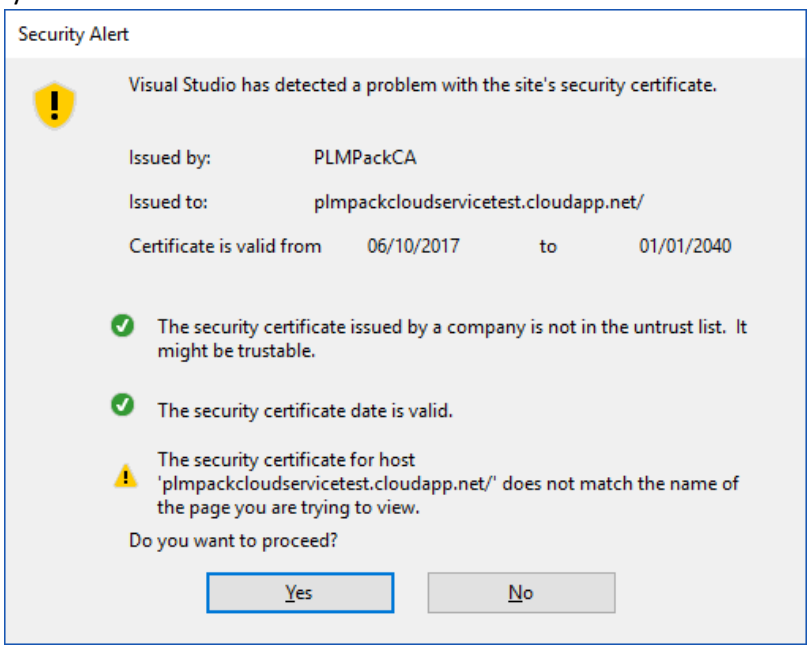
Add reference to WCF service.

In the “Solution explorer” tree, right-click the “References” node and select **Add Service Reference**.



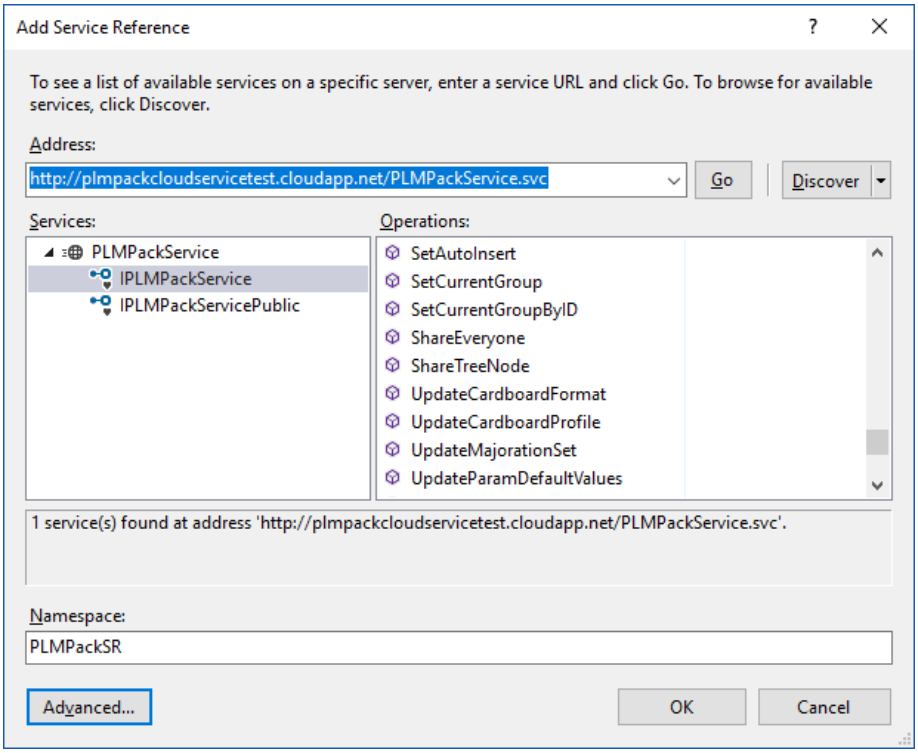
In the field “**Address**” enter the address of our service test instance:
http://plmpackcloudservicetest.cloudapp.net/PLMPackService.svc
Click the button “**Go**” in order to obtain a service description.

You then get a security alert because the service certificate is a test one that does not come from a certification authority…



You can safely proceed and click '**Yes**'.

You now need to provide a namespace (we user PLMPackSR) and click **'OK'** to generate a service reference class.



Connect to the service using the following credentials.

```
System.Net.ServicePointManager.ServerCertificateValidationCallback = ((sender, certificate, chain, sslPol
icyErrors) => true);
using(PLMPackServiceClient client = new PLMPackServiceClient()) {
    client.ClientCredentials.UserName.UserName = "guest";
    client.ClientCredentials.UserName.Password = "guest_";
    string message = string.Format("Logged as {0}", client.get_UserName());
}
```

Good practices regarding usage of client object

When calling the service, either resort to a using block or do not forget to call the client method Close().

```
client.Close();
```

Why is authentication even needed for PLMPack in the product?

When using a parametric component, there are both default values and material dependent values that are defined by user.

Default parameters are attached to a component only whereas majorations depends both on materials and on component.

Accessing a component

Obtain a component GUID using its name

To do this, one can use the method `PLMPackServiceClient.GetComponent(string groupName, string componentName).`

```
string componentName = "F421"; // FEFCO 421
Guid[] guids = null;
using(PLMPackServiceClient client = new PLMPackServiceClient()) {
    client.ClientCredentials.UserName.UserName = "guest";
    client.ClientCredentials.UserName.Password = "guest_";
    guids = client.GetComponent("treeDiM", componentName);
}
if (guids.Any()) {
  // some component was found
}
```

A list of all available component names can be obtained by calling the method `string[] GetAllComponentNames(string groupName).`

Get a list of parameters and image with default parameters

To get the list of parameters of the parametric component and get a first image of it, use the ProcessComponent() method.

```
Guid componentGuid = guids[0]; // the component Guid obtained in previous step
int imageWidth = 0, imageHeight = 0; // the dimensions of image wanted
DCCompParameter[] parameters = null;
DCCompProcessResult result = null;
using(PLMPackServiceClient client = new PLMPackServiceClient())
{
        client.ClientCredentials.UserName.UserName = "guest";
        client.ClientCredentials.UserName.Password = "guest_";
        result = client.ProcessComponent(componentGuid, ReadParameters(), new DCCompFormat()
{
```

```
            Format   =   OutFormat.IMAGE,   Size   =   new   DCCompSize()   {

                CX   =   imageWidth,
   CY   =   imageHeight
            }
        });
        if   (null   !=   result.Errors)   {
            foreach(string   err   in   result.Errors)
   {
       // do something with error messages
   }
        }
    }   // image
if   (null   !=   result.OutFile.Bytes)
   using(var   ms   =   new   System.IO.MemoryStream(result.OutFile.Bytes))
  {
        Image   img   =   Image.FromStream(ms);
        // do something with image
    }
   // use the parameters to display values of
   // * cut length
   // * area
   // * bounding box dimensions
```

The list of parameters and their default values is available in the result object and accessible through the OutParameters property.

```
DCCompParameter[] parameters = result.OutParameters;
```

Building a layout

Get a list of possible layouts with thumbnails for a given set of parameters

One must first define a cardboard format, and instantiate a DCCompLayoutSettingFormat class that will define the layout settings such as format, margins, spaces and alignments. Then, call method `ComponentLayoutList()` in order to receive a list of possible layout sorted by efficiency.

Each `DCCompLayout` contains a `DCCompOutputFile` that contains a thumbnail of the layout.

A list of all layouts should be presented to the user for him to select the best layout.

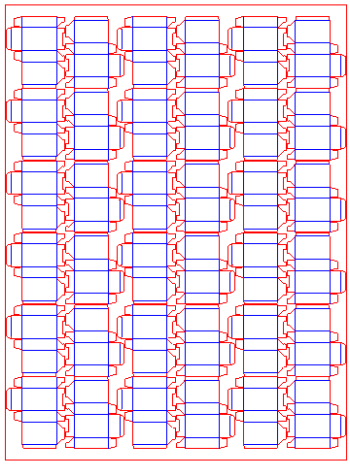
```
                DCCompLayoutSettingsFormat layoutSettings = new
DCCompLayoutSettingsFormat()
                {
                    CarboardFormat = new DCCardboadFormat()
                    {
                        ID = 0,
                        Name = string.Empty,
                        Description = string.Empty,
                        Length = 2000,
                        Width = 1600
                    },
                    AlignH = AlignmentH.AH_CENTER,
                    AlignV = AlignmentV.AV_CENTER,
                    AllowRotationCol = true,
                    AllowRotationRow = true,
                    Margin = new DCDimensions() { X = 5, Y = 5 },
```

```
                    MarginRemaining = new DCDimensions() { X = 5, Y = 5 },
                    SpaceBetween = new DCDimensions() { X = 3, Y = 3 }
                };
                DCCompLayoutList listLayouts = null;
                using (PLMPackServiceClient client = new PLMPackServiceClient())
                {
                    client.ClientCredentials.UserName.UserName = UserName;
                    client.ClientCredentials.UserName.Password = UserPassword;
                    listLayouts = client.ComponentLayoutList(ComponentGuid, Parameters,
layoutSettings, 200);
                 }
```

Select a layout and get all available data

Once the user as selected a layout among all the choice, function ProcessLayout() can be called to get a larger image of the layout with all corresponding data.

```
                DCCompLayout layout = lbImages.Items[iSel] as DCCompLayout;
                DCCompProcessResult result = client.ProcessLayout(ComponentGuid,
Parameters,
                    new DCCompFormat()
                    {
                        Format = OutFormat.IMAGE,
                        Size = new DCCompSize()
                        { CX = pbLayout.Size.Width, CY = pbLayout.Size.Height }
                    },
                    layout);
                if (null != result.Errors)
                {
                    foreach (string error in result.Errors)
                    { /* do something with error messages */}
                }
                using (var ms = new System.IO.MemoryStream(result.OutFile.Bytes))
                {
                    Image img = Image.FromStream(ms);
                    // show larger image
                }
```



Getting the best StackBuilder palletization

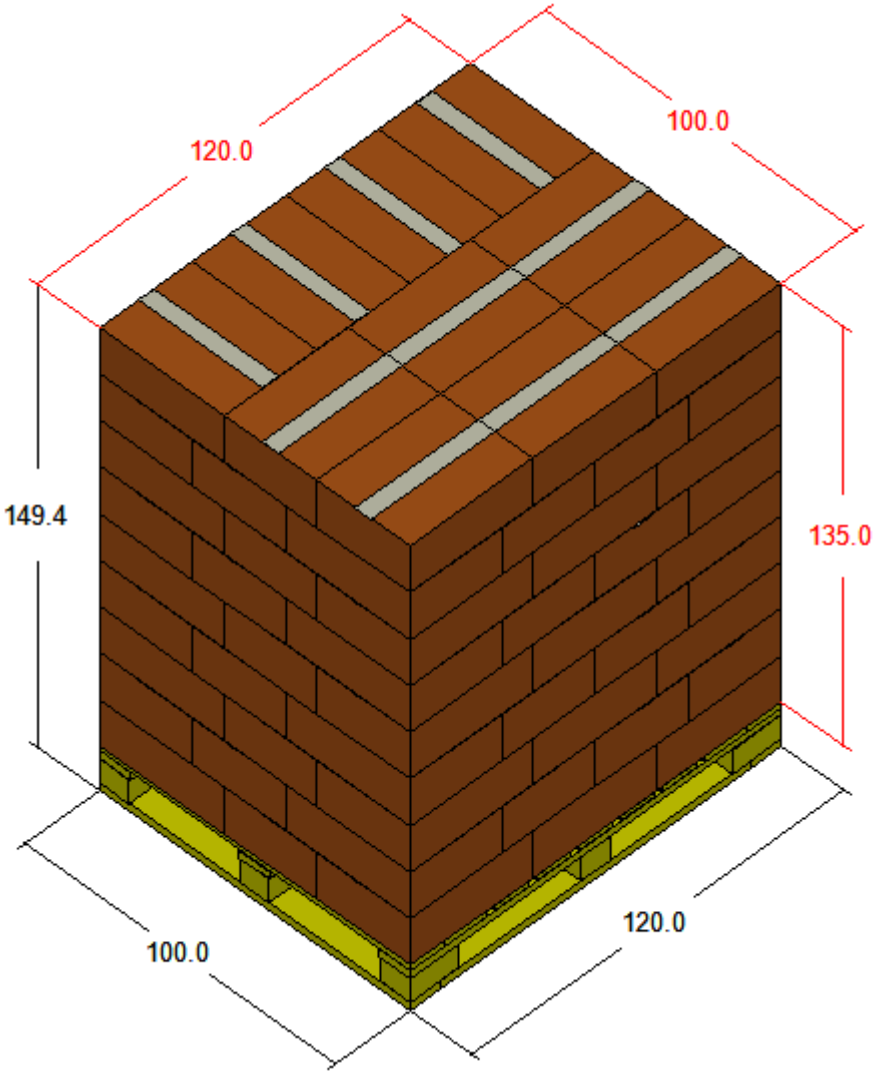
Calling SB_GetBestSolution() with a case, a pallet and a constraint set allows getting an image of an optimal pallet plus corresponding result data without choosing a layer pattern. This feature allows to quickly assess the number of cases that can be loaded on an given pallet.

```
using (PLMPackServiceClient client = new PLMPackServiceClient())
{
        client.ClientCredentials.UserName.UserName = UserName;
        client.ClientCredentials.UserName.Password = UserPassword;

       DCSBSolution sol = client.SB_GetBestSolution
               (
                       new DCSBCase()
                       {
                              Name = "Case"
                            , Description = "Default case"
                            , DimensionsOuter = OuterDimensions
                            , HasInnerDims = false
                            , DimensionsInner = InnerDimensions
                            , Weight = CaseWeight
                            , MaxWeight = 1000.0
                            , NetWeight = 0.9 * CaseWeight
                            , ShowTape = true
                            , TapeWidth = 50.0
                            , TapeColor = Color.Beige.ToArgb()
                            , Colors = new int[6]
                                {   Color.Chocolate.ToArgb()
                                    , Color.Chocolate.ToArgb()
                                    , Color.Chocolate.ToArgb()
                                    , Color.Chocolate.ToArgb()
                                    , Color.Chocolate.ToArgb()
                                    , Color.Chocolate.ToArgb()
                                }
                        }
                        , new DCSBPallet()
                        {
                            Name = "EUR2"
                            , Description = "EUR2"
                            , PalletType = "EUR2"
                            , Color = Color.Yellow.ToArgb()
                            , Dimensions = PalletDimensions
                            , Weight = PalletWeight
                            , AdmissibleLoad = 2000.0
                        }
                        , null
                        , new DCSBConstraintSet()
                        {
                            Overhang = PalletOverhang
                            , Orientation = new DCSBBool3() { X = true, Y = true, Z = true }
                            , MaxHeight = new DCSBConstraintDouble()
                                            { Active = true, Value_d = MaxPalletHeight }
                            , MaxWeight = new DCSBConstraintDouble()
                                            { Active = false, Value_d = 1000.0 }
                            , MaxNumber = new DCSBConstraintInt()
                                            { Active = false, Value_i = 100 }
                        }
                        , new DCCompFormat()
                        {
                            Size = new DCCompSize()
                            {
                                CX = pbStackbuilder.Size.Width
                                , CY = pbStackbuilder.Size.Height
                            }
                            , Format = OutFormat.IMAGE
                        }
                        , true
                    );
                    if (null != sol)
                    {
                        foreach (string err in sol.Errors)
                        {
                             // do something with error messages
                        }
                        if (sol.Errors.Length > 0)
                            return;
                    }
                    else
                        return;
```

```
            if (null != sol.OutFile)
                using (var ms = new System.IO.MemoryStream(sol.OutFile.Bytes))
                {
                    Image img = Image.FromStream(ms);
                      // do something with error messages

                }
        }
```



Sample :

1. Download and unzip the following Visual Studio project :

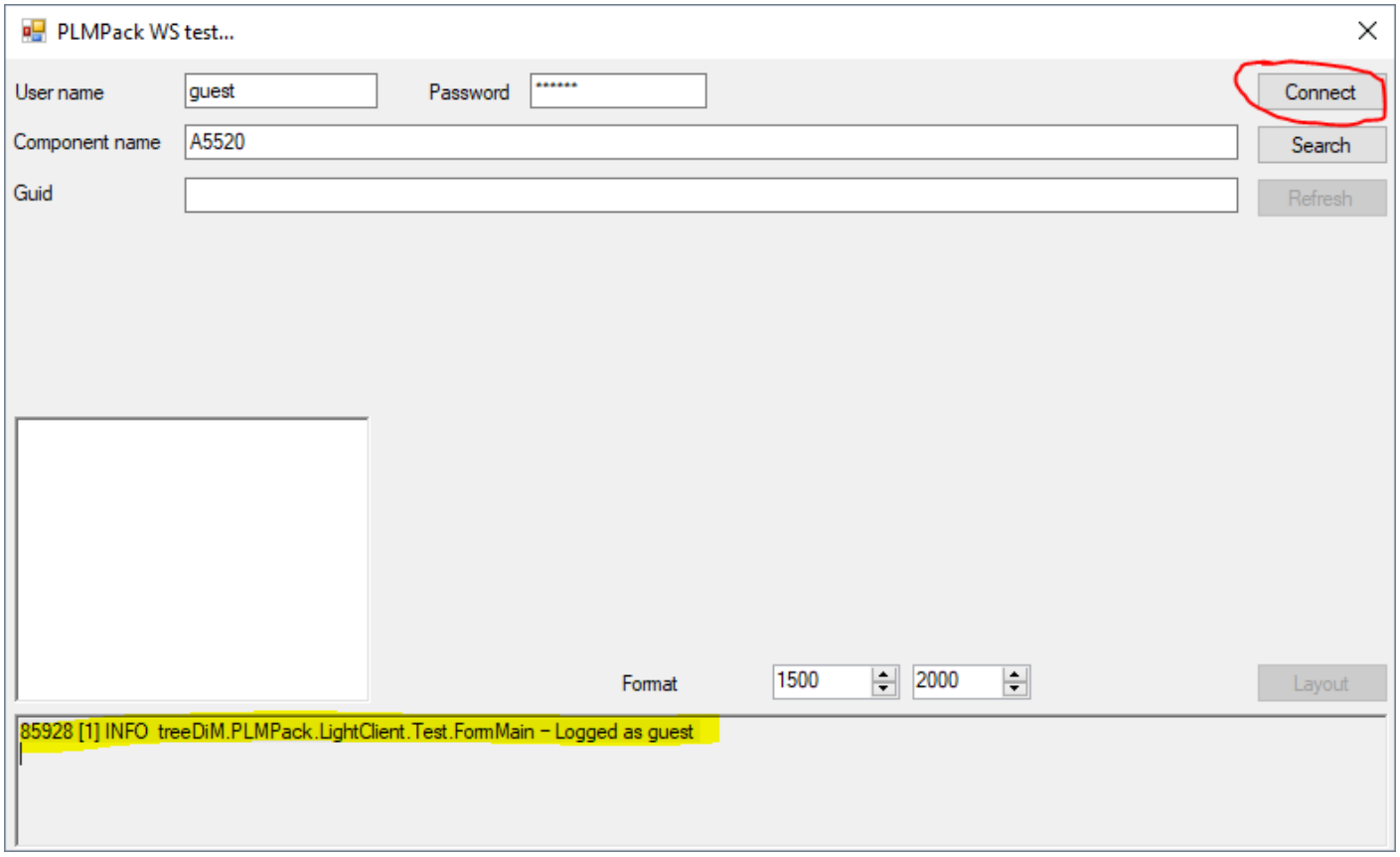
   http://www.treedim.com/DL/integration/treeDiM.PLMPack.LightClient.Test.zip

2. Open the folder :

   C:\.....\treeDiM.PLMPack.LightClient.Test\bin\Release

3. Launch the exe :

   treeDiM.PLMPack.LightClient.Test.exe

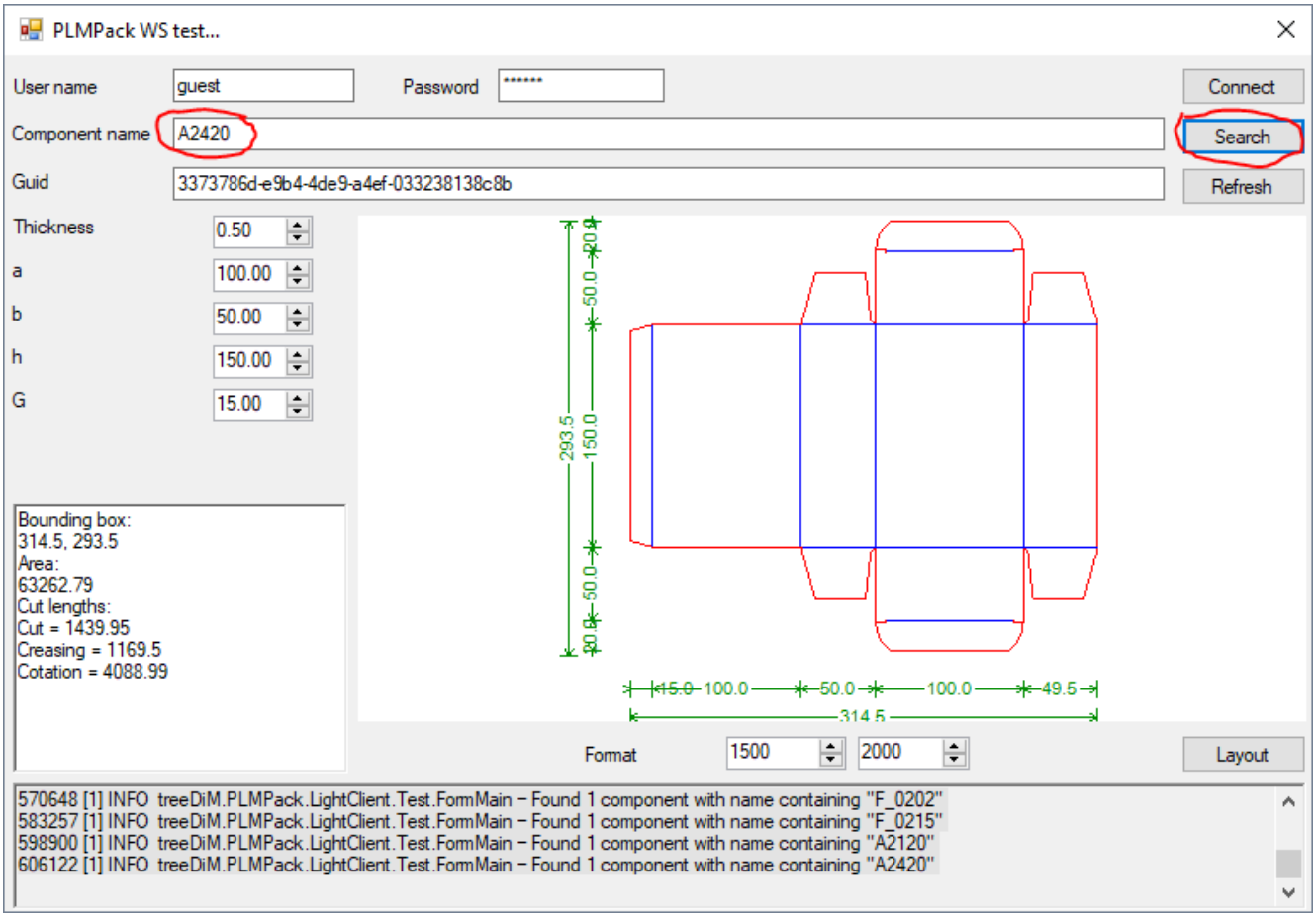


4. Connect to the Azure plateform as “guest” user name :

   Use the Connect button and wait until the message “Logged as guest” is displayed

5. Then select a Component name (an ECMA or FEFCO model) :

Select a component name (see the list in the treeview of PackLib) and use the Search button



Then the component is displayed and you can change the parametric values to resize the model. The results (size, areas, lengths) are displayed in the bottom left window.

6. Layout
    a. Select Format values and then use the “Layout”