How to Consume StackBuilder WCF Service

Contents

[Consuming the web service 1](#_Toc68064354)

[Access the web service url 1](#_Toc68064355)

[How to use the web service for homogeneous stacking 1](#_Toc68064356)

[How to use the web service for heterogeneous stacking 3](#_Toc68064357)

[Instantiate a list of content items. 3](#_Toc68064358)

[Add some content items to this list 3](#_Toc68064359)

[Build a pallet. 3](#_Toc68064360)

[Build a constraint set 3](#_Toc68064361)

[Call method SB\_GetHSolutionBestCasePallet() to get a global view of loading 3](#_Toc68064362)

[Call method SB\_GetHSolutionPart() to get each pallet image and data. 4](#_Toc68064363)

[Using the test application 5](#_Toc68064364)

[Download the sample application 5](#_Toc68064365)

[Run the sample application 5](#_Toc68064366)

# Consuming the web service

## Access the web service url

The web service is either available:

- on our Azure web server at: <http://stackbuildercloudappservice.cloudapp.net/StackBuilder.svc>

- on your local server at: <http://localhost/StackBuilder.svc>

## How to use the web service for homogeneous stacking

Calling SB\_GetCasePalletBestSolution() with:

* a case DCSBCase,
* a pallet DCSBPallet,
* a constraint set DCSBConstraintSet, 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 (StackBuilderClient client = new StackBuilderClient())

{

DCSBSolution sol = client.SB\_GetCasePalletBestSolution( new DCSBCase()

{

Name = "Case",

Description = "Default case",

DimensionsOuter = new DCSBDim3D() { M0 = 400.0, M1 = 300.0, M2 = 200.0 }, HasInnerDims = false,

DimensionsInner = null,

Weight = 4.5,

MaxWeight = 100.0,

NetWeight = 0.9 \* 4.5,

ShowTape = true,

TapeWidth = 50.0,

TapeColor = Color.Beige.ToArgb(),

Colors = Enumerable.Repeat<int>(Color.Chocolate.ToArgb(), 6).ToArray() }

, new DCSBPallet()

{

Name = "EUR2",

Description = "EUR2",

PalletType = "EUR2",

Color = Color.Yellow.ToArgb(),

Dimensions = PalletDimensions,

Weight = 22.0

}

, null

, new DCSBConstraintSet()

{

Overhang = new DCSBDim2D() {M0 = 0.0, M1 = 0.0},

Orientation = new DCSBBool3()

{

1. = false, // vertical length not allowed
2. = false, // vertical width not allowed
3. = true // vertical height allowed

},

MaxHeight = new DCSBConstraintDouble()

{

Active = true,

Value\_d = 1800.0 // max pallet height

},

MaxWeight = new DCSBConstraintDouble()

{

Active = false, Value\_d = 0.0

},

MaxNumber = new DCSBConstraintInt()

{

Active = false, Value\_i = 100

},

AllowMultipleLayerOrientations = true

}

, new DCCompFormat()

{

Size = new DCCompSize()

{

CX = 600,

CY = 600

},

Format = OutFormat.IMAGE

}

, true

); if (null != sol)

{

foreach (string err in sol.Errors) Console.Write(err); if (sol.Errors.Length > 0) return;

} // image

pbStackbuilder.Image = null; if (null != sol.OutFile)

using (var ms = new System.IO.MemoryStream(sol.OutFile.Bytes))

{

I mage img = Image.FromStream(ms); pbStackbuilder.Image = img;

}

// case count

Console.WriteLine($“CaseCount = {sol.CaseCount}”);

Console.WriteLine($“Total pallet weight = {sol.WeightTotal}”); Console.WriteLine($“Efficiency = {sol.Efficiency}”);

Console.WriteLine($”Bounding box load = {sol.BBoxLoad}”); }

# How to use the web service for heterogeneous stacking

## Instantiate a list of content items.

List<DCSBContentItem> items = new List<DCSBContentItem>();

## Add some content items to this list

items.Add(

new DCSBContentItem()

{

// number of cases

Number = 4,

// force vertical position

Orientation = new DCSBBool3() { X = false, Y = false, Z = true },

// case

Case = new DCSBCase()

{

Name = “Case1”,

Description = “Description case1”,

DimensionsOuter = new DCSBDim3D()

{ M0 = 400.0, M1 = 300.0, M2 = 200.0 },

HasInnerDims = false,

DimensionsInner = null,

Weight = 1.0,

MaxWeight = 1000.0,

NetWeight = 0.9,

ShowTape = true,

TapeWidth = 50.0,

TapeColor = Color.Beige.ToArgb(),

Colors = Enumerable.Repeat(Color.Chocolate.ToArgb(), 6).ToArray()

},

// priority index

PriotityIndex = -1

}

);

By default, items are ordered by decreasing volume before being laid.

You can force some items to be stacked first by setting their priority index to a positive number.

Items with a 0-priority index will be stacked first, before items with priority index 1 and so on…

Be warned that forcing priority of items comes with a cost in term of volume efficiency.

## Build a pallet.

var pallet = new DCSBPallet()

{

Name = "EUR2",

Description = "EUR2",

PalletType = "EUR2",

Color = Color.Yellow.ToArgb(),

Dimensions = new DCSBDim3D() { M0 = 1200.0, M1=1000.0, M2=144.0 },

Weight = 22.0

};

## Build a constraint set

var constraintSet = new DCSBHConstraintSet()

{

MaxHeight = new DCSBConstraintDouble() { Active = true, Value\_d = 1700.0 },

MaxWeight = new DCSBConstraintDouble() { Active = false, Value\_d = 1000.0 },

Overhang = new DCSB\_Dim2D() { M0=0.0, M1=0.0 }

};

## Call method SB\_GetHSolutionBestCasePallet() to get a global view of loading

Using this method, the client can receive an image of the full loading and the number of pallets needed.

using (StackBuilderClient client = new StackBuilderClient())

{

var hSolution = client.SB\_GetHSolutionBestCasePallet(

items.ToArray(),

pallet,

constraintSet,

new DCCompFormat()

{

Size = new DCCompSize()

{

CX = 600,

CY = 600

},

Format = OutFormat.IMAGE

},

false

);

// process image

if (null != hSolution.OutFile)

{

using (var ms = new System.IO.MemoryStream(hSolution.OutFile.Bytes))

{

// image generated can be used here

Image img = Image.FromStream(ms);

}

}

// get the number of pallets

int numberOfPallets = hSolution.PalletCount;

}

## Call method SB\_GetHSolutionPart() to get each pallet image and data.

Knowing the number of pallets, the client application can now call this method SB\_GetHSolutionPart() for each bin in the solution to get image of a specific pallet of the solution together with loading data.

for (int binIndex=0, binIndex < palletCount; ++binIndex)

{

using (var client = new StackBuilderClient())

{

var hSolItem = client.SB\_GetHSolutionPart(

items.ToArray(),

pallet,

constraintSet,

0, // solution index (0 for the best solution)

binIndex, // bin index

new DCCompFormat()

{

Size = new DCCompSize()

{

CX = 600,

CY = 600

},

Format = OutFormat.IMAGE

},

true

);

// user can access load and total bounding boxes

var bboxLoad = hSolItem.BBoxLoad;

var bboxTotal = hSolItem.BBoxTotal;

// user can access load and total (i.e. with pallet) weights

double weightLoad = hSolItem.WeightLoad;

double weightTotal = hSolItem.WeightTotal;

// a list of all items in a specific pallet can also be obtained

DCSBContentItem[] content = hSolItem.Content;

if (null != hSolItem.OutFile)

{

using (var ms = new System.IO.MemoryStream(hSolItem.OutFile.Bytes))

{

Image img = Image.FromStream(ms);

}

}

}

}

# Using the test application

## Download the sample application

The sample application can be downloaded from: <http://www.plmpack.com/stackbuilder/WCFService/treeDiM.StackBuilder.WCFService.Test.zip>The zip archive contains both a VS2017 project with sources and an executable. Once the zip extracted, the executable can be accessed under: treeDiM.StackBuilder.WCFService.Test\bin\release\treeDiM.StackBuilder.WCFService.Test.exe

## Run the sample application

## 

Before running the sample application, make sure the address of the service is correctly set in the file ***treeDiM.StackBuilder.WCFService.Test.exe.config*** .

It should read something as:

<client>

<endpoint address="http://localhost/StackBuilder.svc" binding="basicHttpBinding"

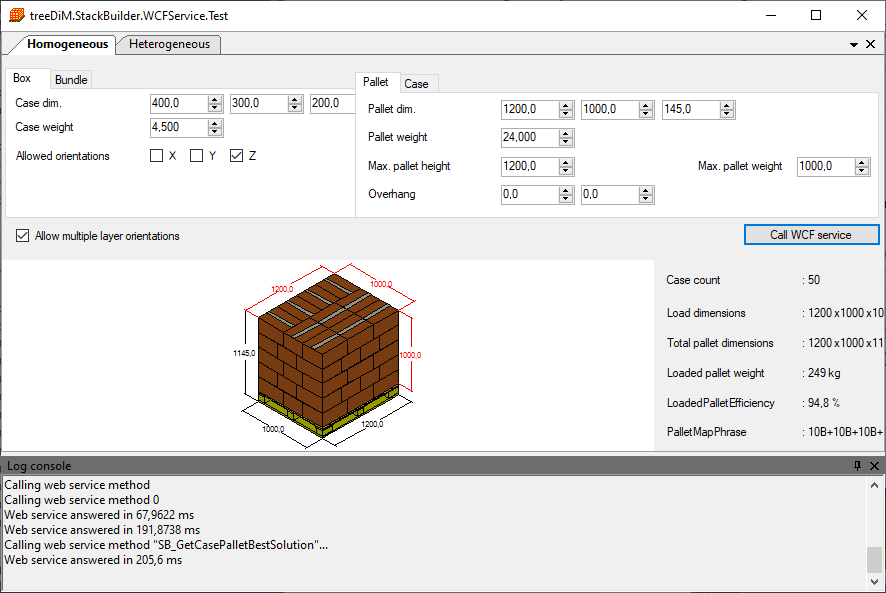
bindingConfiguration="BasicHttpBinding\_IStackBuilder" contract="SB\_SR.IStackBuilder"

name="BasicHttpBinding\_IStackBuilder" />

</client>

Once you have entered case and pallet dimensions, call the service by pressing the “Call WCF Service” button.

In homogeneous mode, you would see:



In heterogeneous mode, you would see:

