ptc

vuforia™ studio

Scaling Digital Twin Experiences 402 Extending Configurations within the Experience Server

Copyright © 2021 PTC Inc. and/or Its Subsidiary Companies. All Rights Reserved.

User and training guides and related documentation from PTC Inc. and its subsidiary companies (collectively "PTC") are subject to the copyright laws of the United States and other countries and are provided under a license agreement that restricts copying, disclosure, and use of such documentation. PTC hereby grants to the licensed software user the right to make copies in printed form of this documentation if provided on software media, but only for internal/personal use and in accordance with the license agreement under which the applicable software is licensed. Any copy made shall include the PTC copyright notice and any other proprietary notice provided by PTC. Training materials may not be copied without the express written consent of PTC. This documentation may not be disclosed, transferred, modified, or reduced to any form, including electronic media, or transmitted or made publicly available by any means without the prior written consent of PTC and no authorization is granted to make copies for such purposes. Information described herein is furnished for general information only, is subject to change without notice, and should not be construed as a warranty or commitment by PTC. PTC assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

The software described in this document is provided under written license agreement, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. It may not be copied or distributed in any form or medium, disclosed to third parties, or used in any manner not provided for in the software licenses agreement except with written prior approval from PTC.

UNAUTHORIZED USE OF SOFTWARE OR ITS DOCUMENTATION CAN RESULT IN CIVIL DAMAGES AND CRIMINAL PROSECUTION.

PTC regards software piracy as the crime it is, and we view offenders accordingly. We do not tolerate the piracy of PTC software products, and we pursue (both civilly and criminally) those who do so using all legal means available, including public and private surveillance resources. As part of these efforts, PTC uses data monitoring and scouring technologies to obtain and transmit data on users of illegal copies of our software. This data collection is not performed on users of legally licensed software from PTC and its authorized distributors. If you are using an illegal copy of our software and do not consent to the collection and transmission of such data (including to the United States), cease using the illegal version, and contact PTC to obtain a legally licensed copy.

Important Copyright, Trademark, Patent, and Licensing Information:
See the About Box, or copyright notice, of your PTC software.

UNITED STATES GOVERNMENT RIGHTS

PTC software products and software documentation are “commercial items” as that term is defined at 48 C.F.

R. 2.101. Pursuant to Federal Acquisition Regulation (FAR) 12.212 (a)-(b) (Computer Software)(MAY 2014) for civilian agencies or the Defense Federal Acquisition Regulation Supplement (DFARS) at 227.7202-1(a) (Policy) and 227.7202-3 (a) (Rights in commercial computer software or commercial computer software documentation) (FEB 2014) for the Department of Defense, PTC software products and software documentation are provided to the U.S. Government under the PTC commercial license agreement. Use, duplication or disclosure by the U.S. Government is subject solely to the terms and conditions set forth in the applicable PTC software license agreement.

PTC Inc., 121 Seaport Blvd, Boston, MA 02210 USA

Introduction

In our previous example (401), we took our two configurations of our quadcopter, and we placed the 3d model content in the Experience Server Content Deliver Service (CDS).

In this example, we will jump to a consumer experience and look into how you can leverage the CDS for static content, Thingworx for dynamic content, and the IRS for helping to manage configurations i.e. which combination of assets go together, and bring these into an experience which actually has very little idea of what it is viewing – it could be a quadcopter or a bicycle – it simply receives a description of the configured item and it creates the appropriate AR view.

Firstly, to help the user identify the different models, we are going to use an Image Target. Let's imagine our model quadcopter is delivered in a nice presentation box, and the box has a photo of the model on the front. A prospective customer, wishing to see what the model looks like, can scan the box and will see an Augmented 3d model appear.

Our cover photos will be used to create Image targets which are used to attach the 3D augmented model to the presentation box – the 3d model will appear to be sitting on top of the box.

In this example. We're going to use a feature of Vuforia Engine called "Image targets", which allow an image/photo to be defined as the target.

<picture here showing the basic layout – static content in CDS, experience in ES, thingmarks calling out specific configs>

402.1 Creating the content package

Following on from the previous example, we will start first be preparing our content.

1. Make a new directory called "**sdte402**"

2. Inside here create the following folders
    a. “models”
    b. “images”
    c. “targets”
    d. “WEB-INF”
3. The “models” and “WEB-INF” folders are identical to the previous example – you can copy the files into this new folder structure.

4. Edit the metadata.json file, and amend the version number and description e.g.

```
{
  "version": "1.0.1",
  "title": {
    "en": "Quadcopters"
  },
  "description": {
    "en": "Models and targets of various quadcopters for exercise 400"
  }
}
```

5. In the “images” folder, we will place the images that form our product box (see example above). To create these images, take a look back at example 201, where we used Creo illustrate to create our 3D model of the quadcopter.
    a. Load quadcopterDT1.c3di into Illustrate, and create a new Figure.
    b. New Figure opens up a dialog asking you to select a starting view – choose the **3D Default** option**.** Click **“Create and Close”**
    c. By default, this is given name “Figure 1”, so lets rename this “box-cover”. You can right-click on the figure name an choose **Rename** or you can click the Rename option in the Figure menu.
    d. Using the mouse in the 3D window, position / orient the quadcopter model on the screen – find a nice angle that makes it look nice”
    e. At the bottom of the 3D View, there will be a notice saying “No Figure View has been set”. Once you’ve found your perfect view, click the **Use the current View** to record this. The Figure will now remember this view.
    f. Finally, lets export this image to a file – we’re going to use this to define an image target. With our “box-cover” figure open in the viewer, click **File > Save Figure As > Save as Image File.** Choose the ‘images’ folder we’ve just created, and save this file as `imgDT1.png`**.**

Note : example images (and the targets subsequently generated from these, are include in the examples that you have downloaded. These instructions are included here to help the reader understand the steps taken to generate these – you can skip these if you want to get the example working first – come back later,

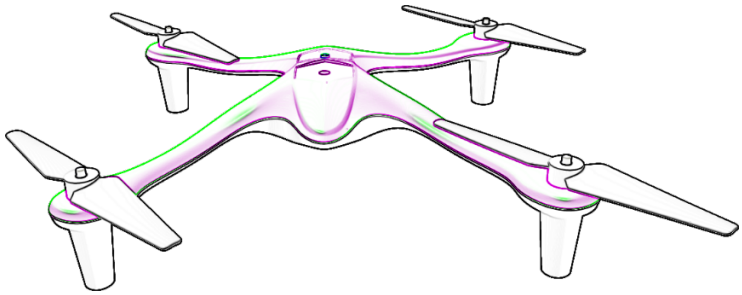
perhaps with you own model, and use these steps to customize the example for your own usecase.

6. Repeat the same steps for quadcopterDT2, saving the resulting file as `imgDT2`. Our images folder should look like this:-

```
+ images
        imgDT1.png
        imgDT2.png
```

7. In the “targets” folder, we will place the image target that we will create using the Vuforia image target generator. Along with the target itself, we have a guide view that will be used to help the user identify where to point their device. We will create the guide view from the same image that we use to create the target.

   guideDT1.png is a copy of imgDT1, saved from illustrate in the previous step. We use a different file here because we might want to treat the guide differently e.g. we might want an outline of the image, or perhaps a cartoon shaded image. It’s up to you what your guides look like – in the examples provided, we’ve used an outline approach. This was created using a standard image editing tool, for example Adobe Photoshop or equivalent.



```
+ targets
    + quadDT1
        guideDT1.png
        quadDT1.dat
        quadDT1.xml
    + quadDT2
        guideDT2.png
        quadDT2.dat
        quadDT2.xml
```

Note : this step is optional – working targets are provided in the folders provided. The information here is provided for cases where the reader may wish to learn how to create their own targets.

Image targets can be created using the Vuforia Target generator, available in the Vuforia Developer Portal.

<TBC – explain how to create an image target using the portal>

8. Your folder structure should look like this :-

```
+ images
+ models
+ targets
+ WEB-INF
```

9. As described in example 401, create a zip file that contains the 4 directories and contents. The zipfile should retain the same name as before e.g. **sdte400.zip**

402.2 Uploading content to the CDS

1. In example 401, we created a representation and named it sdte400. We now need to update the representation with the new assets. For this operation, we ill perform an update (PUT) operation against the CDS webservice. With the zip file you created in step .9 above, run the following commandline

```
curl -X PUT -u %uname%:%passwd% -k -H "X-Requested-With: XMLHttpRequest" -F
"File=@sdte400.zip" -H "Content-Type:multipart/form-data"
%server%/ExperienceService/content/reps/sdte400
```

A helper batch script “create_cds_rep.bat” has been provided; this takes the zipfile name as a parameter.

```
> update_cds_rep sdte400.zip
```

2. Your repository should have PVZ and JSON files for both quadcopter models when complete. To check, you can run the following command

```
curl -u %uname%:%passwd% -k %server%/ExperienceService/content/reps
```

A helper batch script “list_reps.bat” file is provided.

```
> list_cds_reps
```

You should see something like this as the response


```
{"totalCount":1,"items":[{"name":"sdte400","createdby": "YOU","createdon":"DATE",
"modifiedby":"YOU”,"modifiedon":"DATE","url":"https://YOURSERVER/ExperienceService/conten
t/reps/sdte400","metadata":{"version":"1.0.1","title":{"en":"Quadcopters"},
"description":{"en":" Models and targets of various quadcopters for exercise 400"}}}]}
```


Note that the version number and description will have been updated to reflect the contents of the new version of metadata.json

3. Make a note of the url property – this is the location that our new representation is stored. We will use this in our experience.


```
"url":"https://YOURSERVER/ExperienceService/content/reps/sdte400"
```


402.3Update Your Vuforia Studio Experience

Now that your configuration data has been stored inside ThingWorx, your Vuforia Studio experience needs to be edited to accept these changes.

1. Open `ScalingDigitalTwinExperiences402` in Vuforia Studio.
2. Add two new application parameters “guide” and “target”.
3. Add a 3dimage widget, and name it “photo”.
4. Bind app parameter “guide” to the photo ‘resource’
5. Click to add a filter, in here enter the following

```
return '/ExperienceService/content/reps/sdte400/targets/' + value;
```

6. Delete the spatial target
7. Next, create a tml widget. Name it dynamicImageTarget
   The tml widget allows the programmer to create content at the DOM level.
   In this example, we are going to create an image tracking target
8. Click on the ‘edit’ button top open the TML widget edit panel. In this panel, enter the following:-

```
<twx-dt-target
 id = "dynamicimagetarget"
 guide-src = "{{app.params.guidesrc}}"
 src = "{{app.params.imagesrc}}"
 x = "0" y = "0" z = "0"
```

```
    rx = "-90" ry = "0" rz = "0"
    istracked = "false" >
</twx-dt-target>
```

What is this doing? Well, we are declaring an AR tracking target; the database describing this target (the source, or src) is located at {{app.params.imagesrc}}, and the guide-view is {{app,params.guidesrc}}. The {{variable}} syntax says "replace the value of {{variable}} here". We will see where this variable comes from shortly.

This target is what we refer to as a dynamic target – the target itself is a parameter to the experience, and is defined and stored elsewhere. You can follow the same pattern for other target types, e.g. a model target or an area target. The syntax it exactly the same – the target src (a URL) declares HOW and WHERE the target is defined.

The position and rotational values define the location of this tracking target relative to the original of our world. For dynamic examples like this, its best to keep the positional offsets (x,y,z) at zero. In this example, we've set a rotational offset of -90 degrees, which places the target horizontal, like a box on a table top.

9. Open the javascript window. Add the following :-

```
$scope.app.params.guidesrc =
  '/ExperienceService/content/reps/sdte400/targets/' + $scope.app.params.guide;
$scope.app.params.imagesrc =
  'vuforia-image:////ExperienceService/content/reps/sdte400/targets/' + $scope.app.params.target;
```

The first line creates a new application parameter (guidesrc) which is built up from a path (the location of the image in the CDS, the bundle that we uploaded earlier) appended with the {{value}} of the 'guide' parameter, which is passed to the experience when the thingmark is scanned and the code is resolved in the IRS.

The second line is similar, but in this example it creates a URI that references an image target. The URI protocol (vuforia-image://) defines what the content is – an image target in this example - and the path declares where it can be found. Again, the value of the {{target}} application parameter is used here to form the final URI.

When the experience is launched, the act of scanning the thingmark results in the IRS resolving these different parameters, which are then handed to the experience as it starts. The code above the completes the task of defining the values that get inserted into the target definition (see step 8)

```
guide-src = "{{app.params.guidesrc}}"
 src = "{{app.params.imagesrc}}"
```

10. Next, we are going to display the new guideview. Studio has already provide an HTML template in the published application code. The template is, by default, left blank, but we can leverage this to display our guide image. Within the template HTML there is a that is declared with a class name “targetGuide”. Using HTML API, we can request the location of this div

    ```
    let targetGuideDiv = document.querySelector("div.targetGuide");
    ```

    if defined, we can then set the HTML ‘style’ for this div to show the image within the div.

    ```
    if (targetGuideDiv) {
     targetGuideDiv.style.backgroundImage = "url('" + $scope.app.params.guidesrc + "')";
    }
    ```

11. Your Vuforia Studio experience has now been updated to its proper state. Click **Publish** to publish your updated experience.
12. A Vuforia Studio project with the new changes added in this section named `ScalingDigitalTwinExperiences402` can be found in GitHub. **Note**: As with the last tutorial, this project is meant to be used as a reference material for the project file unless you have changed your template mapping.

402.4Update the IRS

Next, we point our thingmarks at the new image targets.

In the previous example (401) we mapped various configs (config:1, config:2 etc.) to our template (template:401). These configs were already mapped to thingmarks that were introduced in earlier examples (202).

In this example, we will create two new configs, 5,6, and we will map these to two new codes (q:do we have to use new codes?)

```
@set vumark5=<your thingmark>
```

Lets start by defining a certain configuration – this config (config:5) will reference our QuadDT1 model, with a yellow painted shell.

Map the color yellow, image guideDT1, target quadDT1 and model QuadcopterDT1 to config:5

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"**urn:curriculum:config:5**\", \"value\": \"urn:curriculum:**color:yellow**\"}" %server%/ExperienceService/id-resolution/mappings

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"urn:curriculum:config:5\", \"value\": \"urn:curriculum:**guide:quadDT1/guideDT1.png**\"}" %server%/ExperienceService/id-resolution/mappings

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"urn:curriculum:config:5\", \"value\": \"urn:curriculum:**target:quadDT1/quadDT1?id=**\"}" %server%/ExperienceService/id-resolution/mappings

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"urn:curriculum:config:5\", \"value\": \"urn:curriculum:**model:QuadcopterDT1.pvz**\"}" %server%/ExperienceService/id-resolution/mappings

Map config:5 to our template

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"urn:curriculum:**config:5**\", \"value\": \"urn:curriculum:**template:402**\"}" %server%/ExperienceService/id-resolution/mappings

And map the template to the experience, passing the parameters in the query section of the url

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{ \"key\":\"urn:curriculum:**template:402**\", \"value\":\"projects/**scalingdigitaltwinexperiences402**/index.html?expId=1^&target=^%7B^%7B**curriculum:target**^%7D^%7D^&model=^%7B^%7B**curriculum:model**^%7D^%7D^&vumark=^%7B^%7Bvuforia:vumark^%7D^%7D^&guide=^%7B^%7B**curriculum:guide**^%7D^%7D^&color=^%7B^%7B**curriculum:color**^%7D^%7D\", \"resourcetype\":\"Experience\",\"title\" : {

\"en\":\"ScalingDigitalTwinExperiences402\" }, \"requires\" : [\"AR-tracking\",\"w320dp\"], \"description\":{ \"en\":\"Curriculum demo 402\" } }" %server%/ExperienceService/id-resolution/mappings

And finally, map a new thingmark to our config

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"**urn:vuforia:vumark:%vumark5%**\", \"value\": \"**urn:curriculum:config:5**\"}" %server%/ExperienceService/id-resolution/mappings

To test the IRS settings, run the resolve script (from previous examples)

- **resolve.bat urn:vuforia:vumark:%vumark5%**

402.n Further investigation

Using a barcode instead of a thingmark…. TBC

@curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\"key\": \"urn:epc:id:sgtin:0000000.004025\", \"value\": \"urn:curriculum:config:5\"}" %server%/ExperienceService/id-resolution/mappings