



vuforiaTM studio

**Scaling Digital Twin
Experiences 401
Configurations and
Content Storage with 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 addition to being able to use the IRS and Thingworx for configuring your AR experience, you can use a feature of the Experience Service called the Content Delivery Service (CDS). The Content Delivery Service is a facility designed for holding 'static' content, data that doesn't change too often. The 3d representations of a product – the pvz files – and any 3d model target, these will typically not change once a product is released, so these are good candidates for content that could be managed within the CDS. In this example, we will adapt our previous example and we will place the 'static' content such as models into the CDS, and we will continue to use Thingworx for the dynamic content e.g. the battery charge information from the IOT system.

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

401.1 Creating the content package

In this example we will use a CDS representation to host all of the content for our examples. We will store the two model variants of our quadcopter that we have created in previous example. Note that a valid alternative approach might be to have one representation per model type – this might be a better approach if the different product variants include different content. The reader may wish to experiment with alternatives once they've completed this example.

For now, we'll work on a single representation containing all the variants.

The following steps describe how to go about preparing the content for upload to the CDS. To help you out, the folder structure has already been created for you as part of the example; the steps are included here so that you can learn how to adapt and extend for your own use.

1. Make a new directory – in this example, we've called it "**sdte401**" (abbreviated form of "scalable digital twin experience 401"). It technically doesn't matter what you call this folder, but you might want to choose a name that is in keeping with the content you are creating.
2. Inside here create a new folder called "**models**"
3. Copy the 2 quadcopter* pvz files into the "models" folder. Your folder should look like this :-

```
+ models
- QuadcopterDT1.pvz
- QuadcopterDT2.pvz
```

4. Alongside “models”, make another new folder called “WEB-INF”
5. In here, create the file “metadata.json”
- 6.

```
- Models
+ WEB-INF
  - metadata.json
```

7. In this file, create the following content

```
{
  "version": "1.0.0",
  "title": {
    "en": "Quadcopters"
  },
  "description": {
    "en": "Models of various quadcopters for exercise 400"
  }
}
```

This metadata descriptor helps in managing the files in the CDS repository

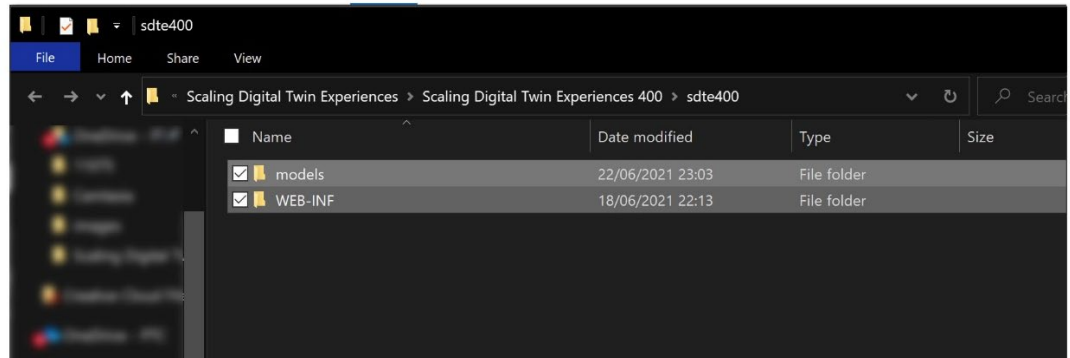
8. Your final folder structure should look like this :-

```
+ models
  - QuadcopterDT1.pvz
  - QuadcopterDT2.pvz
+ WEB-INF
  - metadata.json
```

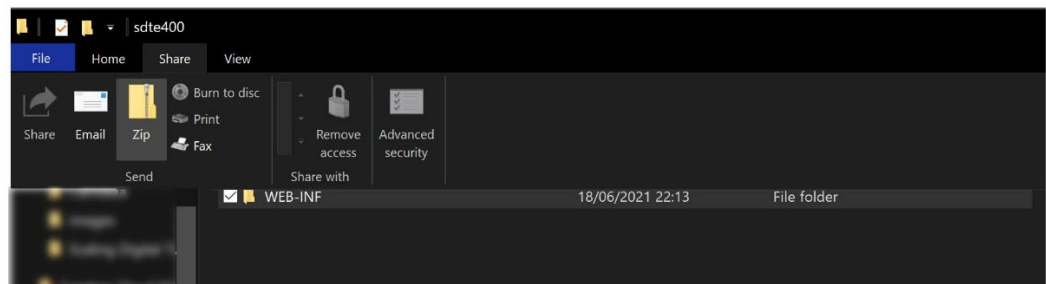
9. Create a zip file that includes these two folders.

If you are using Microsoft Windows, you can use the built-in “send to compressed (zipped) folder function that is provided in Microsoft Explorer.

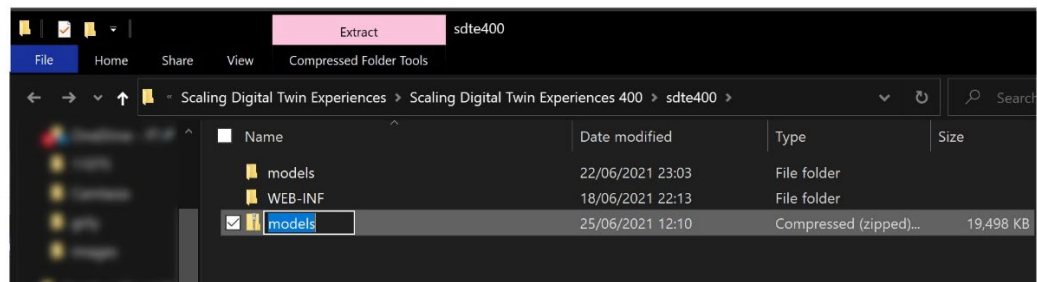
- a. Navigate to the folder containing the models/web-inf
- b. ctrl+click the two folders to select them



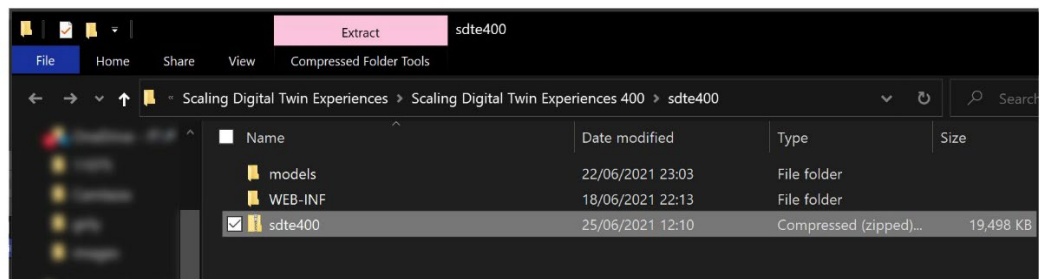
c. Click on the Share menu, and select Zip



d. This will create a new zipfile – you will be prompted to name it



Name the file **sdte400**



< todo : show how to zip up using mac finder? >

401.2 Uploading content to the CDS

Following on from the previous examples in the 300 series, this example assumes you have set the `uname`, `passwd` and `server` environment variables. You can also edit the `.bat` files that are provided, and set the environment variables there.

1. With the zip file you created in step .9 above, run the following command line

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

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

```
> create_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/repos
```

A helper batch script “**list_cds_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/content/rep  
s/sdte400","metadata":{"version":"1.0.0","title":{"en":"Quadcopters"},  
"description":{"en":"Models of various quadcopters for exercise 400"}}}]}
```

Here you can see the metadata associated with the representation.

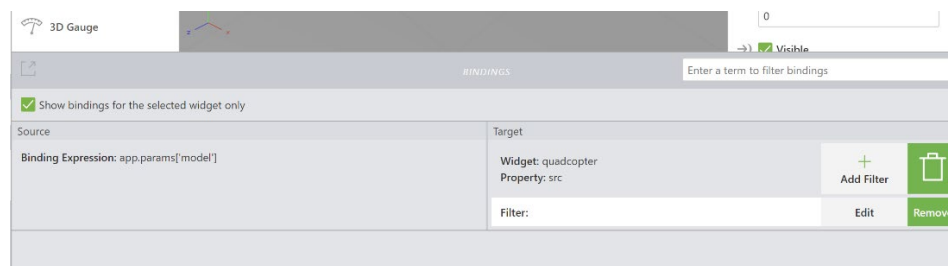
3. Make a note of the highlighted url property – this is the location that our new representation is now stored. We will use this in our experience.

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

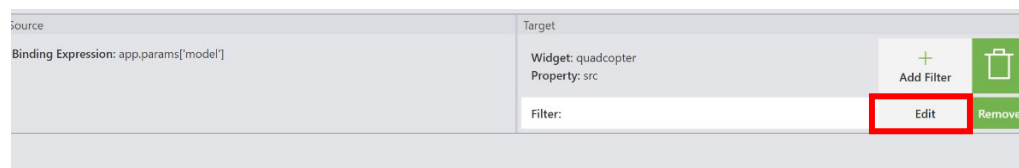
401.3 Update Your Vuforia Studio Experience

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

1. Open `ScalingDigitalTwinExperiences401` in Vuforia Studio.
2. Remove all PVZ and JSON files under **Resources**. These will be loaded from the Experience Service CDS now, using the data you prepared in the section above..
3. The binding between the **model** application parameter and the **Resource** property of the `quadcopter` model widget needs to be altered to reflect the location of the content.
 - a. Select the quadcopter model widget in the model tree on the left, and then open the Bindings pane at the bottom of the screen.



4. To identify the model loaded, we use a filter to take the model parameter (a number) and we append this to a path that describes the location of the model resource.. Click **Edit** to open a dialogue box that will allow you to add a filter.

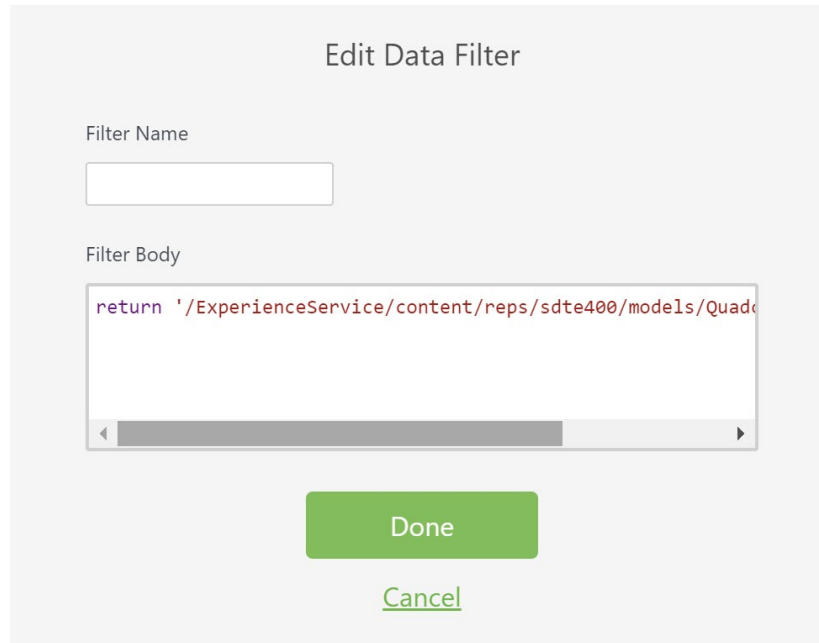


- a. In our previous example, we used the local experience content folder to host our model data, and we used the filter to create the file reference by appending the model ID (the Sapplication parameter) to this path.

```
return 'app/resources/Uploaded/QuadcopterDT'+value+'.pvz'
```

- b. With our content now hosted in the Experience Server CDS, we need to make a small change to the path to reflect the new location – this location was reported in step **302.2.2** above. Note we don't need the <http://server> here, as the Experience Service is hosting both the experience and the content, so relative paths can be used.

```
return '/ExperienceService/content/rep/sdte400/models/QuadcopterDT'  
+ value+'.pvz'
```

Filter Name

Filter Body

```
return '/ExperienceService/content/refs/sdte400/models/Quad'
```

Done

Cancel

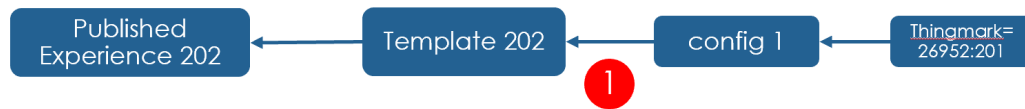
5. Your Vuforia Studio experience has now been updated to its proper state. Click **Publish** to publish your updated experience.
6. You can change the **model** and **color** parameters, re-publish, and you will see the results. The 3d models are retrieved from the CDS, and the color is assigned.
7. 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.

401.4 Mapping with the IRS

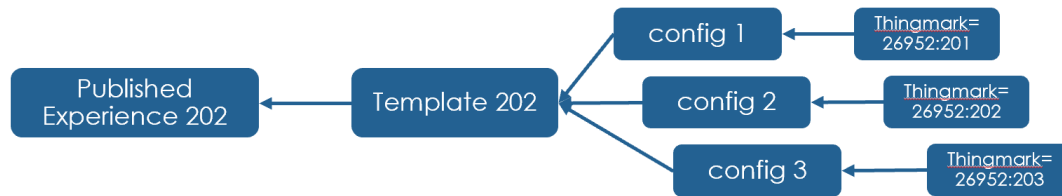
If you have been following the full tutorial series and have already completed example 202 (Configurations with the Identity Resolution Service), you now only need to make a few small changes to your IRS configuration to add the different configurations to this new experience.

If you have not yet completed example 202, we recommend following the detailed instructions in that tutorial before returning here. The remaining instructions here assume 202 is complete.

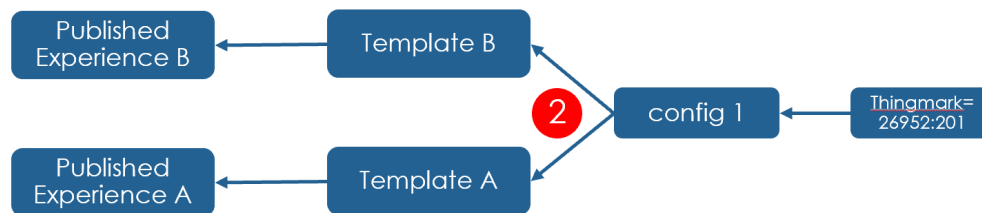
1. Before we start, a quick reminder of what the IRS mapping is providing here. In example 202, we created a number of 'configurations' (collections of parameter values) that we named (config:1, config:2 etc.) and linked these to our experience via a template :-



Following the same pattern, we were able to create many configurations and map them to a shared experience



In this example, we will apply the same principals as before, taking the various configurations we defined and mapping them to a new template and thus to a new experience. The result, see below, is that for each template A, B etc. we can now resolve the various configuration values and deliver these to the experience.



Let's start by adapting the mappings that were created in example 202.

2. In example 202, you started by creating a new 'template' which was then mapped to the corresponding experience. The IRS command looked like this :-

```

curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{ \"key\": \"urn:curriculum:template:202\",
  \"value\": \"projects/scalingdigitaltwinexperiences202/index.html?expId=1^&color=^%7B^%7Bcurriculum:color^%7D^%7D^&model=^%7B^%7Bcurriculum:model^%7D^%7D\",
  \"resourcetype\": \"Experience\", \"title\" : {
    \"en\": \"ScalingDigitalTwinExperiences202\" }, \"requires\" : [ \"AR-tracking\", \"w320dp\" ], \"description\": { \"en\": \"Curriculum demo\" } }"
%server%/ExperienceService/id-resolution/mappings
  
```

this mapped our key `\\"key\\":\\"urn:curriculum:template:202\\"`,
to our experience `\\"value\\":\\"projects/scalingdigitaltwinexperiences202`

Our first task here is to create a new key to map to experience we created above. Using the same syntax as in 202, let's create a new key (401) and map this to experience 401. Here's the new command for the IRS, with the changes highlighted.

```
curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\\"key\\":\\"urn:curriculum:template:401\\", \\"value\\":\\"projects/scalingdigitaltwinexperiences401/index.html?expId=1^&color=%7B^%7Bcurriculum:color^%7D^%7D^&model=%7B^%7Bcurriculum:model^%7D^%7D\\", \\"resourcetype\\":\\"Experience\\",\\"title\\" : { \\"en\\":\\"ScalingDigitalTwinExperiences401\\" }, \\"requires\\" : [ \\"AR-tracking\\",\\"w320dp\\" ], \\"description\\":{ \\"en\\":\\"Curriculum demo\\" } }" %server%/ExperienceService/id-resolution/mappings
```

3. Next we can take different configurations we created in example 202 and map these to the new template. For example, we mapped config:1 to template:202

```
curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\\"key\\":\\"urn:curriculum:config:1\\", \\"value\\":\\"urn:curriculum:template:202\\"}" %server%/ExperienceService/id-resolution/mappings
```

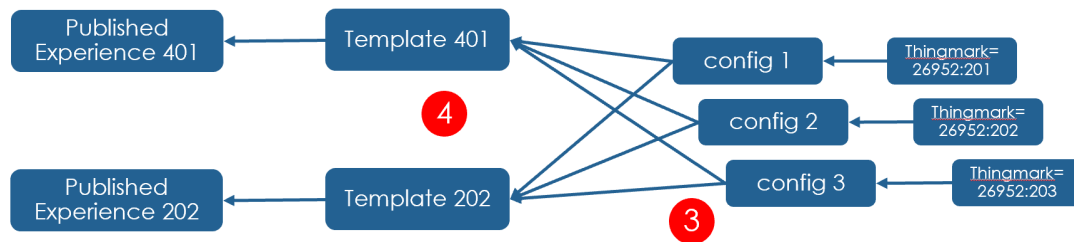
Let's map these to our new template – note that we only need to change the identity of the template:-

```
curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\\"key\\":\\"urn:curriculum:config:1\\", \\"value\\":\\"urn:curriculum:template:401\\"}" %server%/ExperienceService/id-resolution/mappings
```

Doing the same for other configurations e.g.

```
curl -u %uname%:%passwd% -H "Content-Type: application/json" -H "X-Requested-With: XMLHttpRequest" -k -d "{\\"key\\":\\"urn:curriculum:config:2\\", \\"value\\":\\"urn:curriculum:template:401\\"}" %server%/ExperienceService/id-resolution/mappings
```

we end up with configurations that point to both template202 and template401.



- Check your work by running `resolve urn:vuforia:vumark:YOURTHINGMARK` in the CLI. This is taking all the resolutions assigned to that ThingMark and putting them into text form. In example 202 you should have seen a single resolution with the parameters filled in correctly. In this example, you should now see two resolutions – I’ve highlighted the new one that we created above.

```

Select Node.js command prompt

D:\installations>resolve urn:vuforia:vumark:11075:101
{"resolutions":[{"value":"https://xuqztwnu.pp.vuforia.io/ExperienceService/content/projects/scalingdigitaltwinexperience
s202/index.html?expId=1&color=red&model=2","resourcetype":"Experience","title":"ScalingDigitalTwinExperiences202","requi
res":["AR-tracking","w320dp"]}, {"value":"https://xuqztwnu.pp.vuforia.io/ExperienceService/content/projects/scalingdita
ltwinexperiences401/index.html?expId=1&color=red&model=2","resourcetype":"Experience","title":"ScalingDigitalTwinExperi
nces401","requires":["AR-tracking","w320dp"]}]}
D:\installations>

```

- If you now run View and scan your thingmark, you should now be offered multiple choices for which experience to run. Choose Experience 401 and you will see the model now being loaded from the CDS.

401.5 Further investigation

Some important facts to note

- The folder structure outlined in 401.2 - this was just one example of the type of content you can choose to store. The main pre-requisite is the WEB_INF folder containing the metadata.json file. You can create other folders to help manage your content, for example you might have images and perhaps some pdf documents that are linked to the model – create “images” and “documents” folders and place the content in these. Once complete, zip the complete structure up as shown, and upload to the CDS. We will demonstrate this is a future example.
- In this first example, we constructed the path to our final model using a simple parameter that is defined within the experience. In the next tutorial we will look at

how we can use the IRS mapping service to take one launch parameter – the configuration ID – and return multiple assets directly from the CDS storage.