Teamcenter Integration for SolidWorks®

Installation Guide for Dispatcher Services
Version 9.2.0

TranscenData
A Business of International TechneGroup Incorporated
DuPont Circle
Milford, Ohio 45150

Contents

Introduction

This guide describes how to configure **Dispatcher** translation services for SolidWorks data. The **Teamcenter Integration for SolidWorks** supports two translators:

- **SWToJT**: For translating parts and assemblies, including configurations, to the DirectModel (JT) format.
- **SWToDXF**: For translating SolidWorks drawings to the DXF format.

Path delimiter characters ('\' and '/') should be entered following the existing entries in configuration files and following the examples in this document.

Important: Dispatcher translation is supported by a complex set of interrelated, yet independently developed and supported products. ***<u>Each of these products must work without errors in order for translation to succeed</u>***. These independent products are:

- SolidWorks Corporation's **SolidWorks** CAD software
- Teamcenter's **SWToJT** translator
- Teamcenter or Teamcenter Express
- The Teamcenter **Dispatcher Server** and **Dispatcher Client**
- The Teamcenter Integration for SolidWorks

Assumptions

- The software is being installed on a Windows platform. If you are installing the software on a supported UNIX platform, you must replace the Windows-specific syntax in the examples with the equivalent UNIX syntax.
- Only one instance of the SolidWorks software is installed on the module server. Sites with multiple SolidWorks installations must use fully qualified paths and settings so that the **SWToJT** and **SWToDXF** translators will find only one set of SolidWorks executables and configuration files.
- These instructions are only written and validated for **Teamcenter**, not **Teamcenter Express**. It is assumed that, as a derivative of **Teamcenter**, **Teamcenter Express** does not have issues which would prevent a customer from using the **Dispatcher** translation services for SolidWorks.

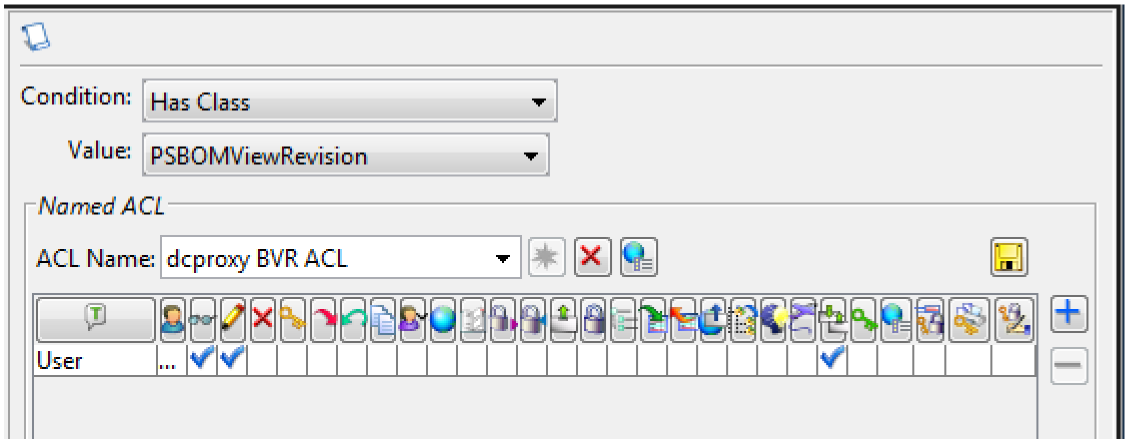
Configuration steps

Install the Teamcenter translation services

The Teamcenter translation services must be installed and operational. These are called **Dispatcher Server** and **Dispatcher Client**. The installation procedure for these services is documented in the Teamcenter install guide.

Grant BVR Modify access to the Proxy User

When translating SolidWorks models, the integration attempts to export models that may not be owned by the proxy user and tries to upload the output files into the Item Revision or Dataset. Therefore the proxy user must have permission to check out, modify, and check in objects which are related to the CAD datasets which are submitted to the translators. This figure shows an ACL that grants the necessary access, given the correct positioning within the Access Manager's rule tree:



Condition: Has Class

Value: PSBOMViewRevision

Named ACL

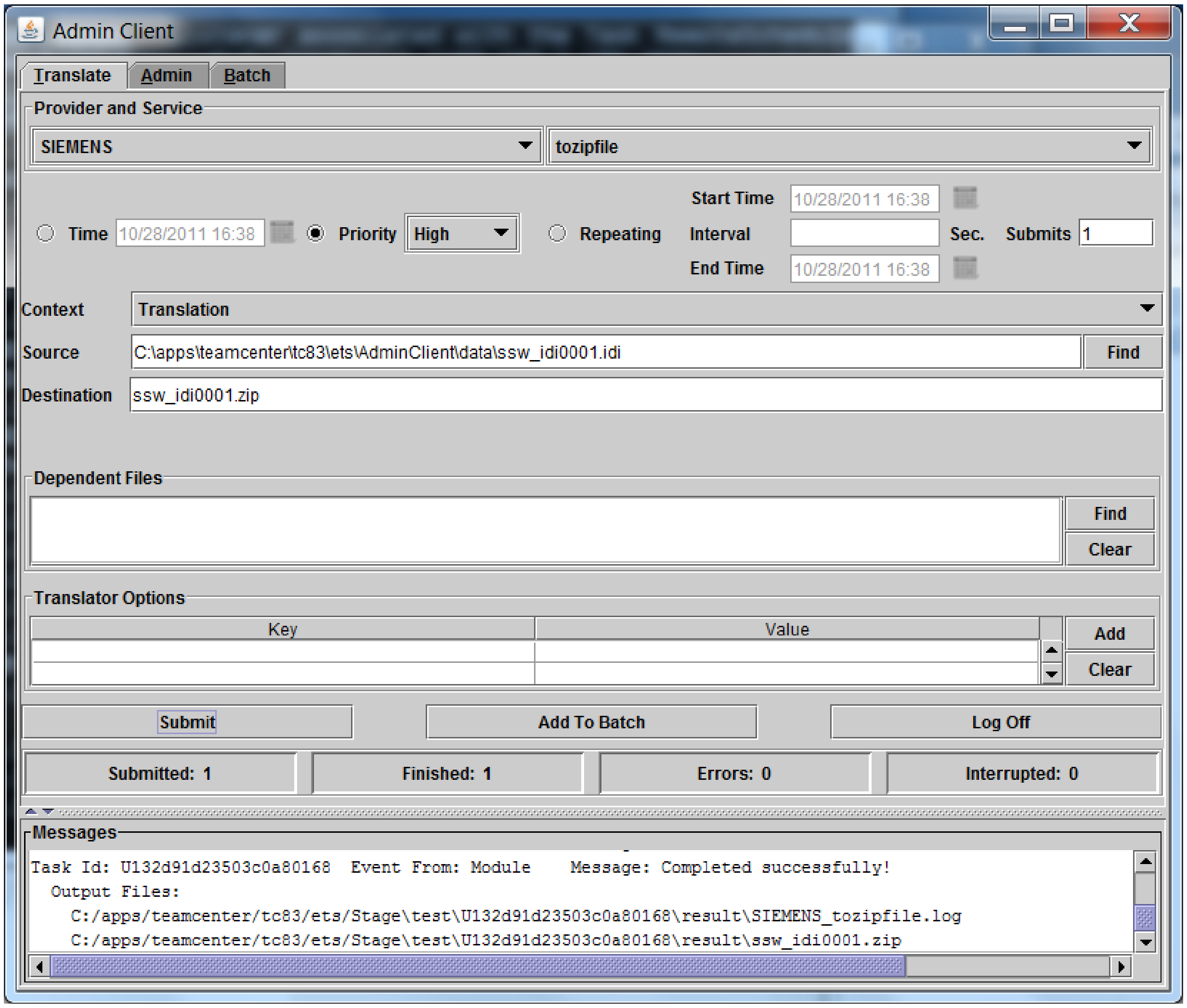
ACL Name: dcproxy BVR ACL

User ... ✓ ✓ ✓

Figure 1: ACL example for SolidWorks translation

Verify basic Dispatcher functionality

Before attempting to configure the SolidWorks translators, you must confirm that the basic services are available. This is best done by submitting a request to the **tozipfile** test translator, from the **Admin Client**:



Provider and Service: SIEMENS / tozipfile

Time 10/28/2011 16:38 [✓ Priority] High Repeating Start Time 10/28/2011 16:38 Interval Sec. Submits 1 End Time 10/28/2011 16:38

Context: Translation

Source: C:\apps\teamcenter\tc83\ets\AdminClient\data\ssw_idi0001.idi

Destination: ssw_idi0001.zip

Dependent Files

Translator Options

Submitted: 1 Finished: 1 Errors: 0 Interrupted: 0

Messages

```
Task Id: U132d91d23503c0a80168  Event From: Module    Message: Completed successfully!
  Output Files:
    C:/apps/teamcenter/tc83/ets/Stage\test\U132d91d23503c0a80168\result\SIEMENS_tozipfile.log
    C:/apps/teamcenter/tc83/ets/Stage\test\U132d91d23503c0a80168\result\ssw_idi0001.zip
```

Figure 2: Verifying basic translator functionality

Additional configuration may be required to set **FMS_HOME**, create **Access Manager** rules to grant the necessary level of access to the Dispatcher Services proxy user, and other tasks. These additional configuration steps are beyond the scope of this document; please refer to the appropriate Teamcenter manual(s) for details.

Install the Teamcenter Integration for SolidWorks client

The **Dispatcher Client** and the **Module Server** must have local access to a client installation of the **Teamcenter Integration for SolidWorks**. This is because during translation, a non-interactive session of the client is run to extract the data from Teamcenter, invoke the translator, and save the output files back into Teamcenter.

This client is used only for **Dispatcher** translation; it must not be used for interactive CAD data management or bulk import/export operations. It must be maintained in a stable configuration because several of its executable and configuration files are used to control the behavior of the SolidWorks translation processes.

Install and verify the SWToJT translator

The translator is a standalone product with its own licensing and its own installation procedure. It must be installed in a location that is locally accessible to the **Module Server**. Before proceeding, confirm that you can start SolidWorks and translate a part from within the session, using the **JT** menu function.

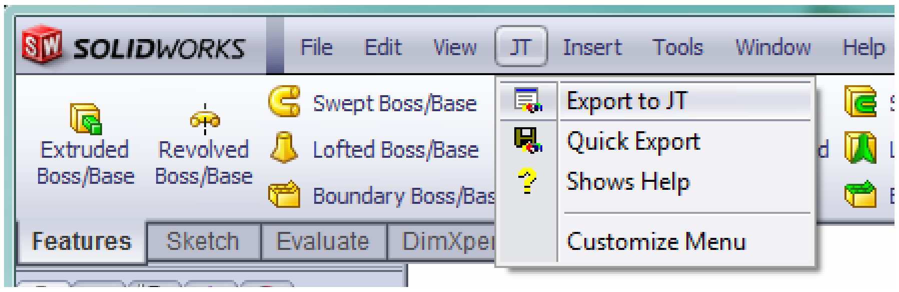


Figure 3: Verifying the SWToJT translator

Configure the Dispatcher Server

The Teamcenter **Dispatcher services** are comprised of two separate but interrelated sets of processes: The **Dispatcher Server** and the **Dispatcher Clien**t.

The **Dispatcher Server** is where all translation is performed. The **Dispatcher Server** runs the **Scheduler** service and the **Module** service[1]. It is highly recommended to configure the **Dispatcher Server** first, and then verify successful translation of SolidWorks data, before continuing to the **Dispatcher Client** setup. By doing so, you can more easily isolate runtime problems to one set of services or the other. Verification is done using the **Admin client**, which simulates the **Dispatcher Clien**t without requiring a connection to Teamcenter.

The **Dispatcher Client** requires the **Dispatcher Server** to be correctly configured and running, because it submits tasks to the **Dispatcher Server**. There will usually be only one instance of the **Dispatcher Client** per Teamcenter site.

When the **Dispatcher Server** is installed through the **Teamcenter Environment Manager** (**TEM**), most of the required configuration is done automatically. This includes the paths to the staging directory as well as to the translator executables. The **Dispatcher Server** and **Dispatcher Client** will be installed in the path specified in **TEM**, which is henceforth called %**DC_HOME%.**

In the instructions which follow, substitute the appropriate full path for the symbolic names. If a path contains any spaces, be sure to use the correct MS-DOS 8.3 format. For example, the path **C:\Program Files (x86)** should be given as **C:\PROGRA~2**:

DC_HOME	The **Dispatcher Root** directory of the **Dispatcher Server** and **Client** software
SWIM_DIR	The installed location of the Teamcenter Integration for SolidWorks client
TRANSLATOR_HOME	The path where the **Swtojt** translator is installed.

[1] For the sake of simplicity, it is assumed that a single Scheduler and a single Module server are installed. In reality, several instances of each service can be installed on the same, or different, server hosts, for load balancing purposes. It is assumed that the Teamcenter Administrator understands how to extend these instructions as needed for more distributed environments.

Configure the SWToJT translator

Modify the module server's classpath

In the **DC_HOME\Module\bin** directory, edit the **setmoduleenv.bat** file to add **SWIM_DIR\swjt\swtojt.jar** to the classpath as shown below.

```
set classpath=%classpath%;%HOME%\lib\activation.jar;%HOME%\lib\fileserver.jar
set classpath=%classpath%;%HOME%\lib\log4j-1.2.13.jar;%HOME%\lib\logwriter.jar
set classpath=%classpath%;SWIM_DIR\swjt\swtojt.jar
```

Modify translator.xml

In the **DC_HOME\Module\conf** directory, edit the **translator.xml** file as directed below.

1. Add **ENTITY SWTOJT** to the **DOCTYPE** Translators as shown in bold below

   ```
   <!DOCTYPE Translators[
   <!ENTITY MODULEBASE " d:\Module\">
   <!ENTITY JAVABIN "G:\TEAMCE~1\install\install\jre\bin">
   <!ENTITY IDEAS "C:\UGS\IDEAS12">
   <!ENTITY ORBIX "C:\ORBIX\Iona\asp\6.1\lib\">
   <!ENTITY PROJTRANS "d:\ProjTrans">
   <!ENTITY PT_JFV "_8000.0.0">
   <!ENTITY UGSPKG "com.teamcenter.translator.ugs.">
   <!ENTITY WRAPPER "com.teamcenter.tstk.server.translator.ugs.">
   <!ENTITY EAIWRAPPER "&WRAPPER;EaiTranslator">
   <!ENTITY SWTOJT "com.transcendata.swimsoa.ets.transwrapper.SWToJt">
   ]>
   ```

2. Find the **<swtojt>** service tag and set or change the following values as shown in bold:

   ```
   <SolidWorksToJt provider="SIEMENS" service="swtojt" maxlimit=”1” isactive="true"
   wrapperclass="&SWTOJT;">
   <TransExecutable dir="SWIM_DIR\swjt" name="swtojt.bat"/>
   ```

3. Set the path to the translator's configuration file, and add the **inputdir** option, if it does not already exist, as shown below in bold.

   ```
   <Options>
           <Option name="configfile" string="-z" value="TRANSLATOR_HOME\etc\tessSW.config"
           description="Path to the config file"/>
           <Option name="outputdir" string="-o" description="Full path to the output directory."/>
           <Option name="inputpath" string="" description="Full path to the input file."/>
           <Option name="inputdir" string="" description="Full path to the input directory."/>
   </Options>
   ```

   Note: Be sure to replace **TRANSLATOR_HOME** with the full path to your

SolidWorksToJT translator installation

4. Comment out the **Postprocess** tag as below[2],

```
<!-- <Postprocess provider="SIEMENS" service="previewservice"/> -->
```

Modify Integration configuration swtojt.bat file

Edit the **swtojt.bat** file in **SWIM_DIR\swjt** and set the path to the swtojt translator installation directory

```
set TRANSDIR=TRANSLATOR_HOME
```

Note: Ensure that the translator's executable file, **Swtojt.exe**, is present in the **TRANSLATOR_HOME** location. **TRANSLATOR_HOME** is the full path to the SolidWorks to JT translator installation.

Configure the SWToDXF translator

Modify translator.xml

1. Add the following content in **DC_HOME\module\conf\translator.xml**, if it does not already exist:

```
<!-- Configuration of the SolidWorks to DXF translator -->
<SolidWorksToDxf provider="SIEMENS" service="swtodxf" maxlimit="1" isactive="true">
        <TransExecutable name="swtodxf.bat" dir="SWIM_DIR"/>
        <Options>
                <Option name="inputpath" string="-i" description="Full path to the input file"/>
                <Option name="outputpath" string="-o" description="Full path to the output file."/>
        </Options>
        <FileExtensions>
                <InputExtensions nitem="1">
                        <InputExtension extension=".SLDDRW"/>
                </InputExtensions>
                <OutputExtensions nitem="1">
                        <OutputExtension extension=".dxf"/>
                </OutputExtensions>
        </FileExtensions>
</SolidWorksToDxf>
```

Be sure to replace **SWIM_DIR** with the full path to the SolidWorks integration client.

[2] In this example it is assumed that the previewservice will not be used to create JPG files from the JT files. Full instructions for configuring the translator to use the previewservice are provided below, in the optional configuration section

Modify the integration's swim.xml file

Edit the **SWIM_DIR\swim.xml** file and add the following entries to the **auxiliary_file_map**:

```
<auxiliary_file cad_type="slddrw" direction="cadtopdm">
        <pdm_location named_ref="DXF" relation_type="IMAN_Rendering" pdm_type="DXF"/>
        <file_name pattern="{cad_name}.DXF"/>
        <generate_command cmd="built_in {cad_name}.DXF" cad_type="DXF" ignore_status="true"
        phase="in_directory"/>
</auxiliary_file>
```

Optionally, the Integration can be configured to generate and import any 2D output file types that can be exported from a SolidWorks drawing. Some examples of these types are **PDF**, **TIFF** and **JPG**. To achieve this, **auxiliary_file** tags that define the generate command for the desired viewables should be added. For example, to generate **PDF** files as part of **DXF** translation, and to attach the **PDF** files with the **IMAN_Rendering** relation, the following **auxiliary_file** tag should be added to the **auxiliary_file_map**:

```
<auxiliary_file cad_type="slddrw" direction="cadtopdm">
        <pdm_location named_ref="PDF_Reference" relation_type="IMAN_Rendering"
        pdm_type="PDF"/>
        <file_name pattern="{cad_name}.PDF"/>
        <generate_command cmd="built_in {cad_name}.PDF" cad_type="PDF" ignore_status="true"
        phase="in_directory"/>
</auxiliary_file>
```

To create and save **TIF** files as **IMAN_specifications** of Item Revisions, add the following auxiliary file definition:

```
<auxiliary_file cad_type="slddrw" direction="cadtopdm">
       <pdm_location named_ref="TIF_Reference" pdm_type="TIF"
       relation_type="IMAN_specification"/>
        <file_name pattern="{cad_name}.TIF"/>
        <generate_command cad_type="TIF" cmd="built_in {cad_name}.TIF" ignore_status="true"
        phase="in_directory"/>
</auxiliary_file>
```

The system can be configured to generate and save additional file types, following these examples.

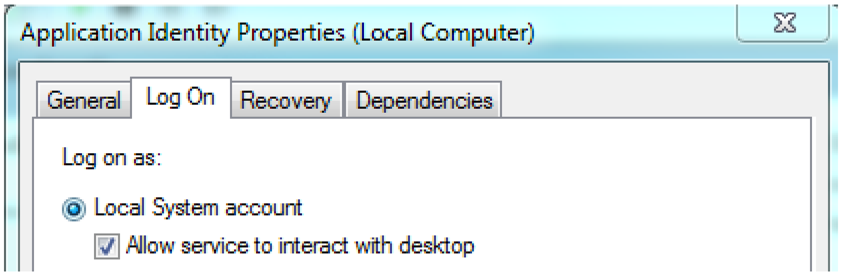
Verify the Dispatcher Server services

Start the **Scheduler** and **Module** services. It is recommended to start the services using the launch scripts provided:

1. Start the **Scheduler** by executing the **DC_HOME\Scheduler\bin\runscheduler.bat** file
2. Start the **Module** server by executing the **DC_HOME\Module\bin\runmodule.bat** file

When started this way, the services will run in a console window on the server's desktop, allowing you to see informative messages as translations are processed. Once correct and reliable translation is verified, these processes can be restarted as Windows services (if configured as services during or after **TEM** installation).

When the **Module Server** is started as a Windows service, ensure that the **Allow service to interact with desktop** option is turned on, otherwise the translator cannot start SolidWorks.



Log on as:

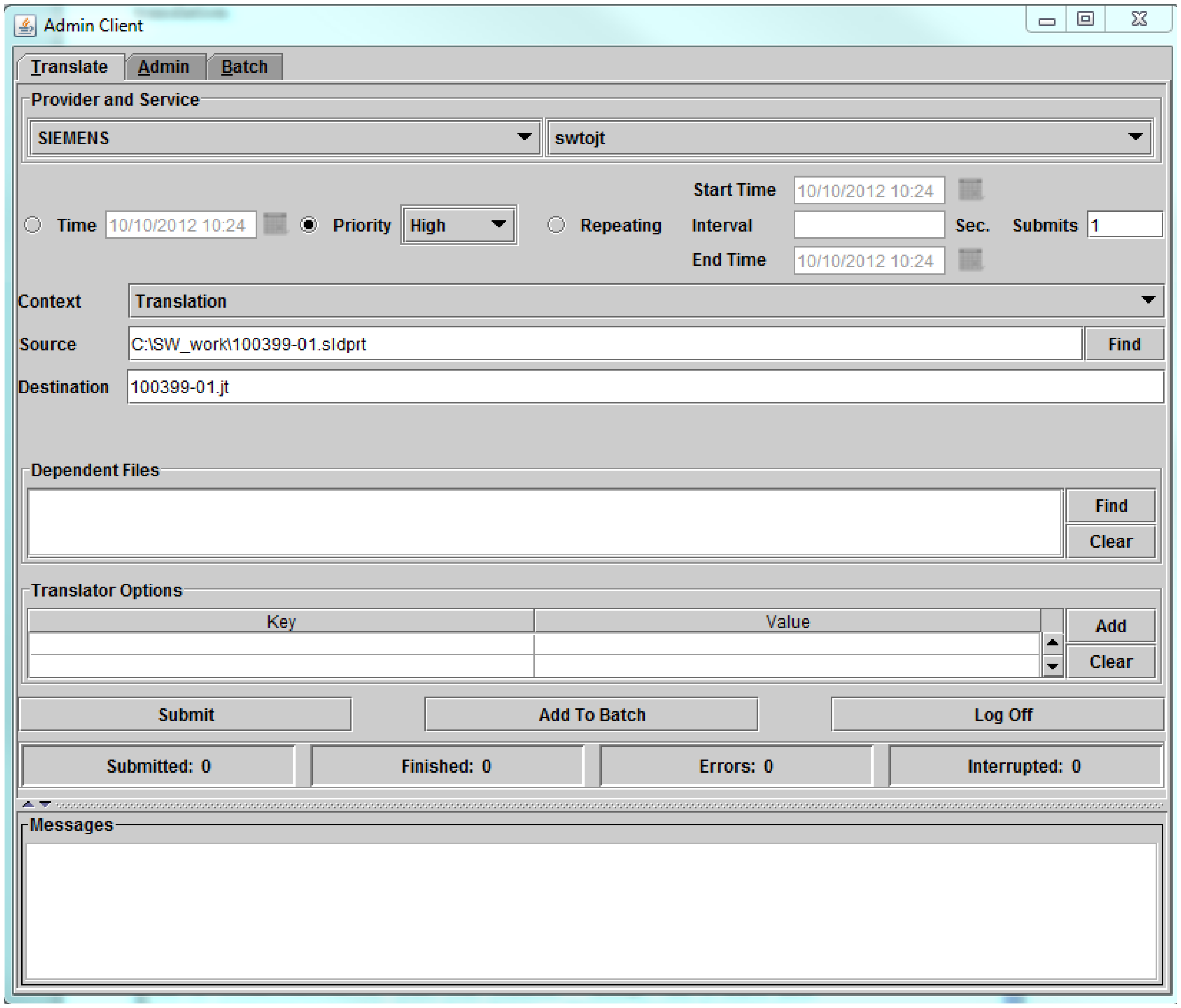
● Local System account

☑ Allow service to interact with desktop

Figure 4 - Module service properties

*When any modification to **DC_HOME\module\conf\translator.xml** file is done, the **Dispatcher Module** service must be restarted.*

Once the services are running, start the Admin **Client** by double-clicking the **runUI.bat** file in the **DC_HOME\AdminClient\bin directory**. Log into the **Admin Client** (no password required), then submit a translation request for a SolidWorks part. To submit the request, select the **swtojt** translator from the drop-down list, then navigate to select a CAD file of the appropriate type. The following shows an example of a SolidWorks part translation:



Provider and Service: SIEMENS — swtojt

Time 10/10/2012 10:24 | ● Priority High | Repeating | Start Time 10/10/2012 10:24 | Interval Sec. | Submits 1 | End Time 10/10/2012 10:24

Context: Translation

Source: C:\SW_work\100399-01.sldprt

Destination: 100399-01.jt

Dependent Files

Translator Options

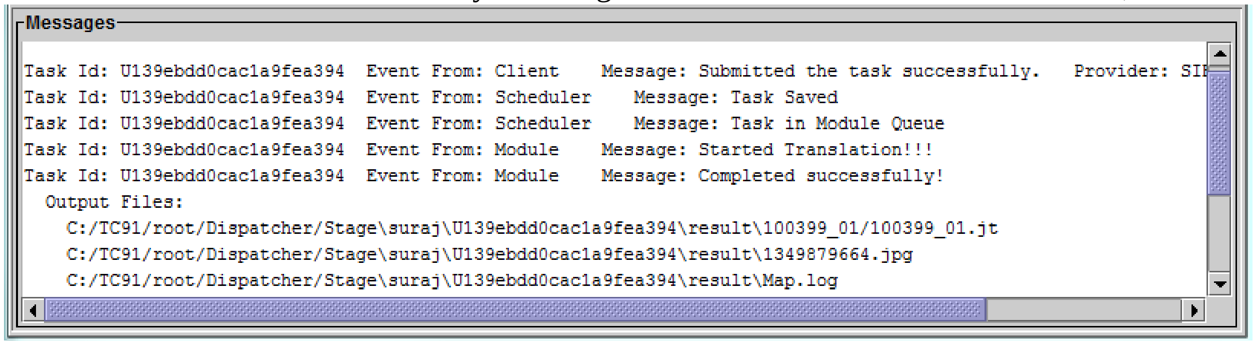
Key	Value

Submitted: 0 | Finished: 0 | Errors: 0 | Interrupted: 0

Messages

Figure 5: Submitting a SWToJT task from the Admin Client

Successful translation is confirmed by a message in the **Admin Client's status** window, as shown below:



```
Messages
Task Id: U139ebdd0cac1a9fea394  Event From: Client     Message: Submitted the task successfully.   Provider: SII
Task Id: U139ebdd0cac1a9fea394  Event From: Scheduler    Message: Task Saved
Task Id: U139ebdd0cac1a9fea394  Event From: Scheduler    Message: Task in Module Queue
Task Id: U139ebdd0cac1a9fea394  Event From: Module     Message: Started Translation!!!
Task Id: U139ebdd0cac1a9fea394  Event From: Module     Message: Completed successfully!
  Output Files:
    C:/TC91/root/Dispatcher/Stage\suraj\U139ebdd0cac1a9fea394\result\100399_01/100399_01.jt
    C:/TC91/root/Dispatcher/Stage\suraj\U139ebdd0cac1a9fea394\result\1349879664.jpg
    C:/TC91/root/Dispatcher/Stage\suraj\U139ebdd0cac1a9fea394\result\Map.log
```

Figure 6: Confirmation message from the Admin Client

*When using the **Admin Client**, be sure to submit only standalone parts. These are self-contained CAD models which have no dependencies on other files. Assemblies and drawings with dependencies to other models will not work, because the **Admin Client** is not aware of the integration data model and will not copy additional dependencies to the staging directory. This additional work is done by the **Teamcenter Integration for SolidWorks** when an on-demand request is submitted to the **Dispatcher Client**.*

Configure the Dispatcher Client

The **Dispatcher Client** is similar to the **Admin Client** in that it submits translation requests to the translators running under control of the **Dispatcher Server**. However, the **Dispatcher Client** is connected to the Teamcenter data base and volumes and so it can submit translation requests for data that is managed by Teamcenter.

1. Modify **DC_HOME\DispatcherClient\conf\Service.properties** to add **TSSWService** to the import statement as below:

   import TSBasicService,....,TSQSearchServices,**TSSWService**

2. Modify **DC_HOME\DispatcherClient\bin\setDispatcherClientEnv.bat**, extending the classpath with the entries highlighted below.

   .
   set classpath=
   set classpath=%classpath%;SWIM_DIR\ETSSW.jar
   set classpath=%classpath%;SWIM_DIR\swim.jar
   for %%i in (%HOME%\lib\*.jar) do call %HOME%\bin\tscpappend.bat %%i
   .
   .

3. Execute the **runDispatcherClient.bat** file to start **Dispatcher Client** service.

A translation request can be generated on demand, by selecting a dataset in Teamcenter and selecting **Translation → Translate** from the menu bar. Or, the integration clients can be configured to automatically submit parts, assemblies and drawings for translation each time these models are saved to Teamcenter as new versions or new revisions. This is documented in the section below titled ***Configure the SolidWorks Integration to submit Dispatcher tasks***.

Install CORBA services

The translation support provided by the Teamcenter Integration for SolidWorks, when running on Windows platforms, requires a persistent TAO process to be running at all times. This is configured as follows:

1. Run the %**TC_ROOT%\iiopservers\install_imrserv.bat** file to register the TAO processes as Windows services. This will create Windows services with the names shown in this example:

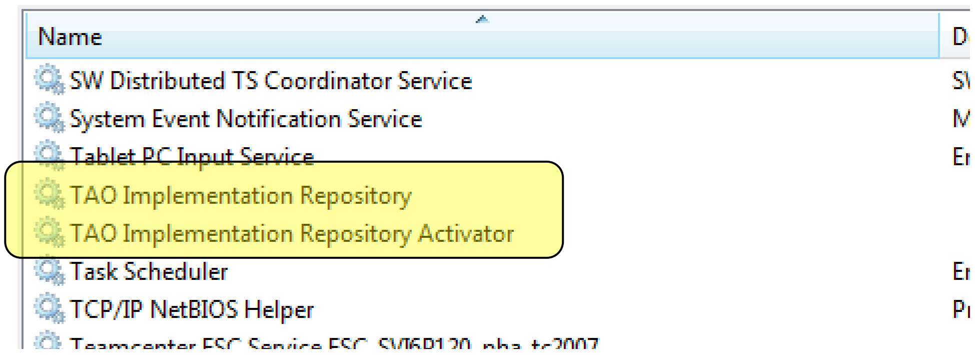

Name | D
SW Distributed TS Coordinator Service | S
System Event Notification Service | M
Tablet PC Input Service | E
TAO Implementation Repository
TAO Implementation Repository Activator
Task Scheduler | E
TCP/IP NetBIOS Helper | P
Teamcenter FSC Service FSC_SVI6P120_nhs_tc2007

Figure 7: Windows services installed by the install_imrserv script

2. Change the startup for both services from "manual" to "automatic"
3. Depending upon your Teamcenter and network configuration, it may also be necessary to change the "Logon" from "Local System Account" to a different user. If so, ask your Teamcenter administrator for the correct user name and password

Optional: Configure the SolidWorks Integration to submit Dispatcher tasks.

This step is optional, but highly recommended to improve performance for the SolidWorks design engineers[3]. It also ensures that whenever a part is checked into Teamcenter as a new version or revision, a translation task is automatically submitted, which in turn prevents the lightweight representation of the CAD model from becoming different from the native CAD geometry.

To configure automatic **SWToJT** task submission, edit the **SWIM_DIR\swim.xml** file and insert the following lines into the auxiliary_file map:

```
<auxiliary_file cad_type="sldprt:sldasm">
      <cadtopdm_control label="Save JT Files (Dispatcher)"/>
      <ets_request translator="swtojt" request_per_model="false" provider="SIEMENS"
      priority="2"/>
</auxiliary_file>
```

This entry in the auxiliary file map also displays a checkbox labeled **Save JT Files (Dispatcher)** in the **Save dialog**, allowing users to choose whether the translation request will be submitted. If you wish to prevent users

[3] Do not make these changes in the copy of swim.xml that is referenced by the Dispatcher Server. Doing so may cause the integration to re-submit tasks unnecessarily. These changes should only be made in the copy of swim.xml that is used by the SolidWorks CAD designers, for their CAD data management work.

from bypassing the **Dispatcher** translation, simply omit the **cadtopdm_control** tag from the XML entry. This will cause the integration to always submit a translation task to the **Dispatcher** for each model that is saved to Teamcenter, without giving the users a choice.

The configuration for translating drawings to DXF files is similar. Adding these lines to the auxiliary file map causes a translation request to be submitted each time a drawing is saved to Teamcenter, and displays a **Save DXF Files (Dispatcher)** checkbox in the Save dialog:

```
<auxiliary_file cad_type="slddrw">
      <cadtopdm_control label="Save DXF Files (Dispatcher)"/>
      <ets_request translator="swtodxf" request_per_model="false"
provider="SIEMENS" priority="2"/>
</auxiliary_file>
```

The **DispatcherClient** process can now be started, either from the command line or as a Windows service, if configured. Once started, the **DispatcherClient** will begin processing **SWToJT** and **SWToDXF** tasks.

Optional: Configure the previewservice post-processor to generate JPG files

This step is optional. It is not required for JT translation. Configuration of the previewservice is documented elsewhere in more detail, as part of the Teamcenter **Dispatcher Services** product. This section is intended as a "quick start" for customers who only need the previewservice for SolidWorks data.

SolidWorks data that was last saved to Teamcenter using an integration version earlier than 8.3.0, or migrated from a legacy system, will not contain preview images. These images can be created and stored as part of **JT** translation by configuring the previewservice post-processor to create **JPG** files[4]. The **JPG** files are derived from the **JT** files from the SWToJT translator, and written to the **Module Server's** staging directory, where the integration will find them and upload them to Teamcenter. The following steps are required:

Enable the previewservice post-processor

Edit the **DC_HOME\module\conf\translator.xml** file and uncomment the **previewservice** tag in the **SolidWorksToJT** translator section. For example, change this:

```
<FileExtensions>
    <OutputExtensions nitem="1">
    <OutputExtension extension=".jt"/>
    </OutputExtensions>
    </FileExtensions>
    <!--Postprocess provider="SIEMENS" service="previewservice"/-->
</SolidWorksToJt>
```

To this

```
<FileExtensions>
    <OutputExtensions nitem="1">
    <OutputExtension extension=".jt"/>
    </OutputExtensions>
    </FileExtensions>
    <Postprocess provider="SIEMENS" service="previewservice"/>
</SolidWorksToJt>
```

[4] Although the JPG images will be displayed in the Teamcenter Rich Client application, they will not be visible in the SolidWorks integration's dialog windows. Those dialogs only display PNG images.

Configure the previewservice to create JPEG files

Locate the **previewservice** translator in **DC_HOME\module\conf\translator.xml**, and change the following:

- Set the **isactive** flag to "**true**"
- Change the **OUTPUT_TYPE** to "**JPG**"
- Change **SIZE** to "200x200px"

For example, change this:

```
<!-- Configuration of the PreviewService translator -->
<PreviewService provider="SIEMENS" service="previewservice" isactive="false">
  <TransExecutable dir="&MODULEBASE;/Translators/previewservice"
     name="previewservice.bat"/>
  <Options>
    <Option name="inputdir" string="" description="Full path to the input
          directory."/>
    <Option name="outputdir" string="" description="Full path to the output
          directory."/>
    <Option name="clientoption" optionkey="OUTPUT_TYPE" string=""
          value="JPG_AND_PDF"
          description="The output file type. Values can be CGM, TIF, HPG, PDF,
          JPG, CGM_AND_JPG, CGM_AND_PDF or JPG_AND_PDF."/>
    <Option name="clientoption" optionkey="SIZE" string="" value="500x500px"
          description="Translator client options. The option define the size of
          the image." />
    <Option name="clientoption" optionkey="COMBINE_OUTPUT" string="" value=""
          description="Translator client options. The option to combine the
          output in single file. Values can be blank or -combine." />
  </Options>
  <FileExtensions>
     <InputExtensions nitem="21">
       <InputExtension extension=".jt"/>
```

To this:

```
<!-- Configuration of the PreviewService translator -->
<PreviewService provider="SIEMENS" service="previewservice" isactive="true">
  <TransExecutable dir="&MODULEBASE;/Translators/previewservice"
           name="previewservice.bat"/>
  <Options>
    <Option name="inputdir" string="" description="Full path to the input
           directory."/>
    <Option name="outputdir" string="" description="Full path to the output
           directory."/>
    <Option name="clientoption" optionkey="OUTPUT_TYPE" string="" value="JPG"
           description="The output file type. Values can be CGM, TIF, HPG, PDF,
           JPG, CGM_AND_JPG, CGM_AND_PDF or JPG_AND_PDF."/>
    <Option name="clientoption" optionkey="SIZE" string="" value="200x200px"
           description="Translator client options. The option define the size of
           the image." />
    <Option name="clientoption" optionkey="COMBINE_OUTPUT" string="" value=""
           description="Translator client options. The option to combine the
          output in single file. Values can be blank or -combine." />
  </Options>
  <FileExtensions>
     <InputExtensions nitem="21">
       <InputExtension extension=".jt"/>
       <InputExtension extension=".cgm"/>
```

```
            <InputExtension extension=".dxf"/>
```

Configure the previewservice executable

The previewservice executable is called **previewservice.bat**, and it is located in the **DC_HOME\module\translators\previewservice** directory. Edit this file and make the following changes:

- Set the **LM_LICENSE_FILE** variable to a valid license server
- Set the **TC_VVCP** variable to the **VVCP** subdirectory location within a Teamcenter Visualization installation[5].

For example, change this:

```
set LM_LICENSE_FILE=CHANGE_ME
set TC_VVCP=CHANGE_ME
```

to this:

```
set LM_LICENSE_FILE=28000@svi6p120
set TC_VVCP=D:\apps\Tc8.3Visualization\VVCP
```

Troubleshooting

When Dispatcher translation does not work as expected, here are some points to consider.

1. Make sure supported versions of SolidWorks and Teamcenter are being used. See the section on **Prerequisites for the Teamcenter Integration for SolidWorks** for more information.
2. Make sure all the specified configuration steps are done.
3. Verify that the SolidWorks part can be translated successfully using the standalone translator framework. This is done by starting the **Scheduler** and the **Module** servers, then submitting a task using the **Admin Client**.
4. Confirm that the JT translator for SolidWorks is working properly, by translating a part or assembly from within a SolidWorks session, using the JT menu.
5. If you experience crashes or hangs of the SolidWorks session, make sure that the **Module Server** does not have an extra sldworks.exe process running on the system (use the task manager). Only one sldworks.exe should be running on the system and it should be owned by the user who starts the **Module Server**.
6. Running standalone SolidWorks sessions on the **Module Server** is not supported. Doing so can cause SolidWorks to crash or hang, resulting in incomplete translations or lost data.
7. It is easier to troubleshoot problems with the **Dispatcher Client** and **Server** when these processes are run from the command line, instead of as Windows services.

[5] The VVCP directory is owned by Teamcenter Visualization, which is installed separately and must include the optional modules that support the jt2jpeg converter.