TEAMCENTER

Integrated Program Planning and Execution — Aurora Integration

Teamcenter 2412

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow’s idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: **support.sw.siemens.com**

Send Feedback on Documentation: **support.sw.siemens.com/doc_feedback_form**

Contents

Using the Aurora integration

Managing, testing, and troubleshooting the Aurora integration

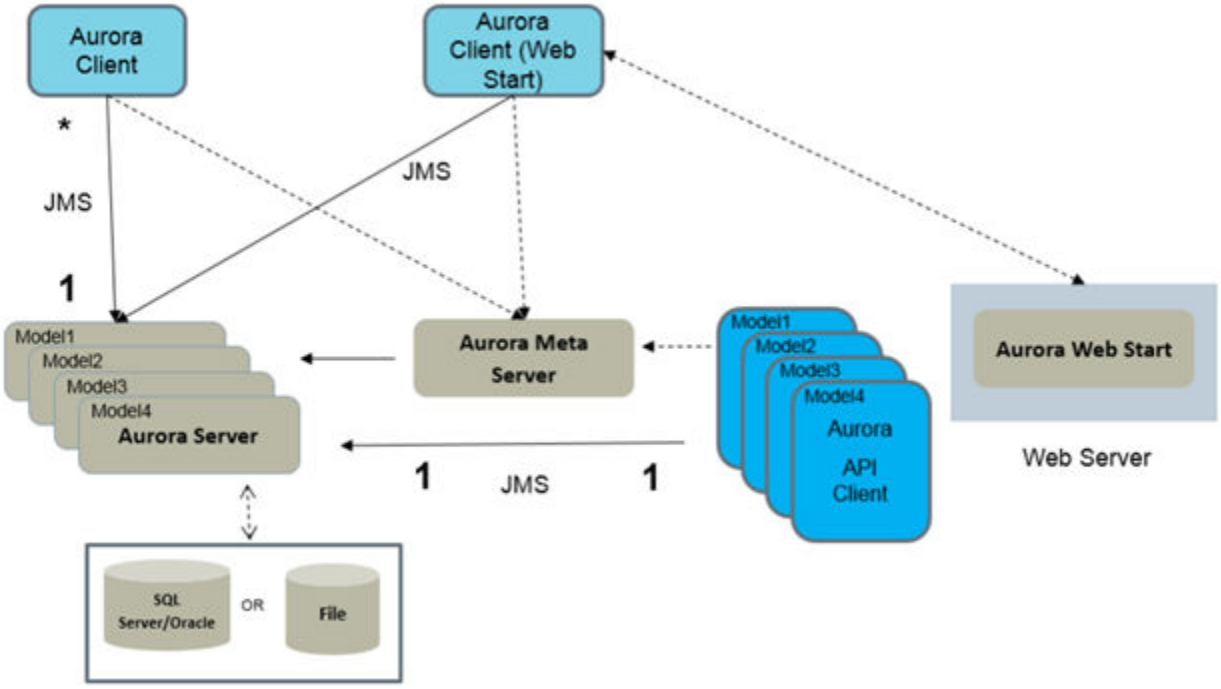
1. Teamcenter-Aurora integration overview

About Aurora

Aurora is an intelligent planning and scheduling software solution. It is mostly used to manage large projects with complex constraints and resource requirements.

IPP&E uses Aurora for its scheduling requirements.

Aurora architecture



Components	Description and Purpose
Aurora Client (Web Start)	This is the version of the Aurora client that is started using Java Web Start. A session-specific URL is provided to the user to start the Aurora client and to automatically connect to the specified session. The user need not install the Aurora client separately.
Aurora Client	This is the Aurora client version that can be separately installed from the Aurora installable rather than using Aurora Web Start. Using this, the user can initiate a new Aurora session or connect to a running Aurora session. This is an optional component.
Aurora API Client	This is the non-interactive client that is provided to enable the integration with Teamcenter. The API client is used by Teamcenter-Aurora Connector to exchange the data between Teamcenter and Aurora.

Components	Description and Purpose
Aurora Web Start	Aurora facilitates launching its client application using JNLP (Java Network Launch Protocol). This is hosted on the web server, which allows the user to start the Aurora client software for the Java Platform directly from the internet using a web browser.
Aurora Meta Server	When a user starts a new Aurora client, it initially connects to the Aurora metaserver. The metaserver checks the user permission and connects to a valid Aurora server session. If an Aurora server session is not running, it starts a session.
Aurora Server	The Aurora server executes all the requests from the client and broadcasts the response back to all the clients connected to that server session.
Database	This is an optional component. It is used by the Aurora server session to load or save the data from the database.
File	If the database component is not configured, the user can save the model as a file in a file system.

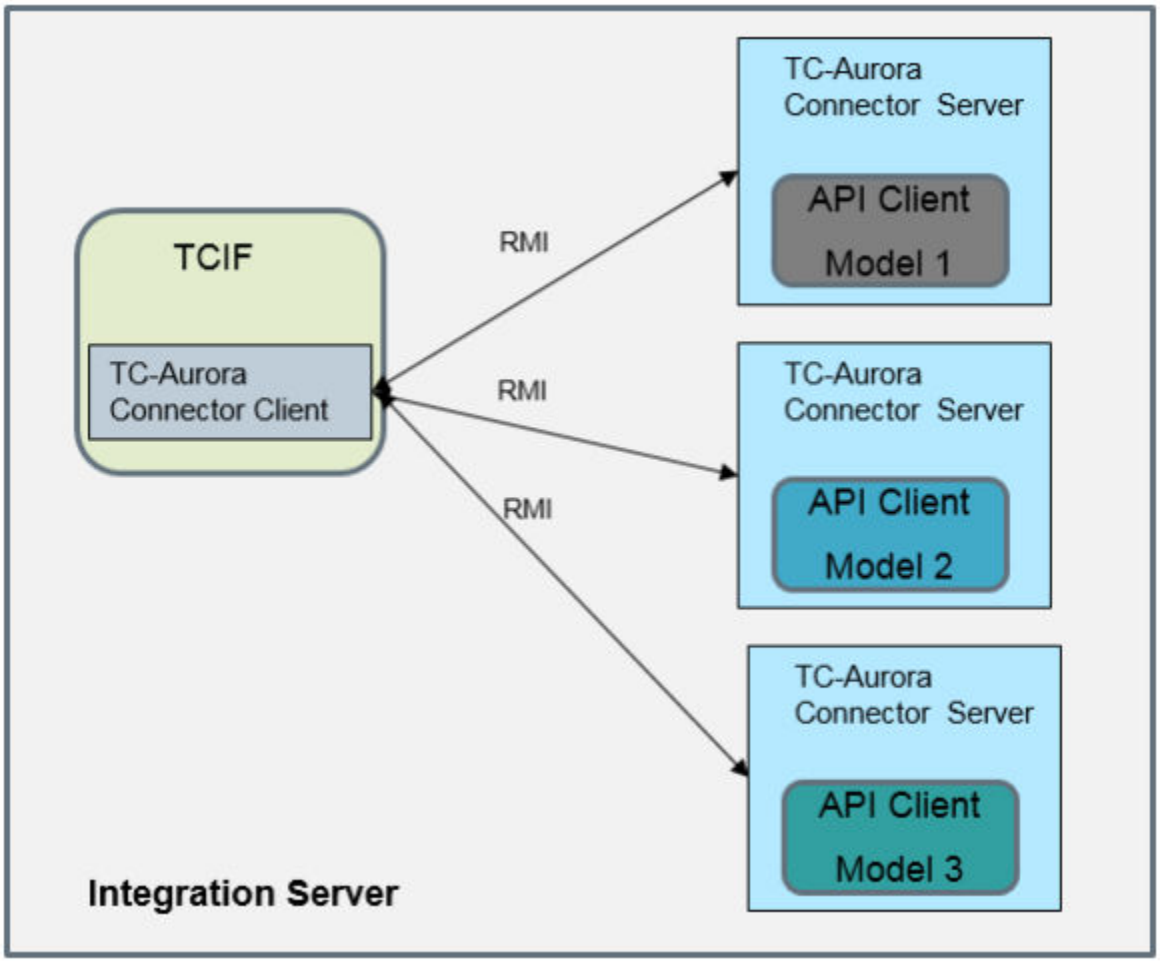
The Teamcenter-Aurora connector

Aurora provides an API client (non-UI client) which helps the other applications to integrate with the Aurora server. The Aurora API client is a Java package that has a set of APIs. The Teamcenter-Aurora connector developed by IPP&E is a Java program that uses the Aurora API client to exchange data with the Aurora server.

The Aurora architecture allows one–to-one mapping between the Aurora server and the Aurora model. This means that for every model, there should be one Aurora server running. A model represents a schedule here. If there are multiple active models (schedules), there will be multiple Aurora servers running.

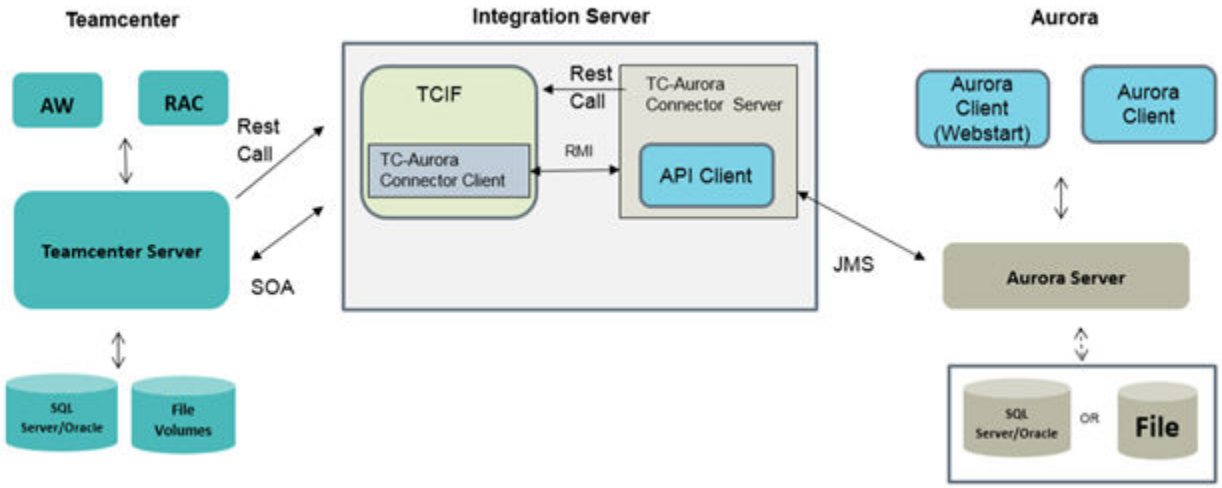
Similarly, there is a one–to-one mapping between the Aurora API client and the Aurora model for integrated models (as opposed to “what if” models, which do not have a corresponding API). This means that for every Aurora API client, there is a corresponding Aurora server session and Aurora model.

Only one API client can run in a given java virtual machine (JVM), so the Teamcenter-Aurora Connector is divided into two components, Teamcenter-Aurora Connector Client and Teamcenter-Aurora Connector Server.



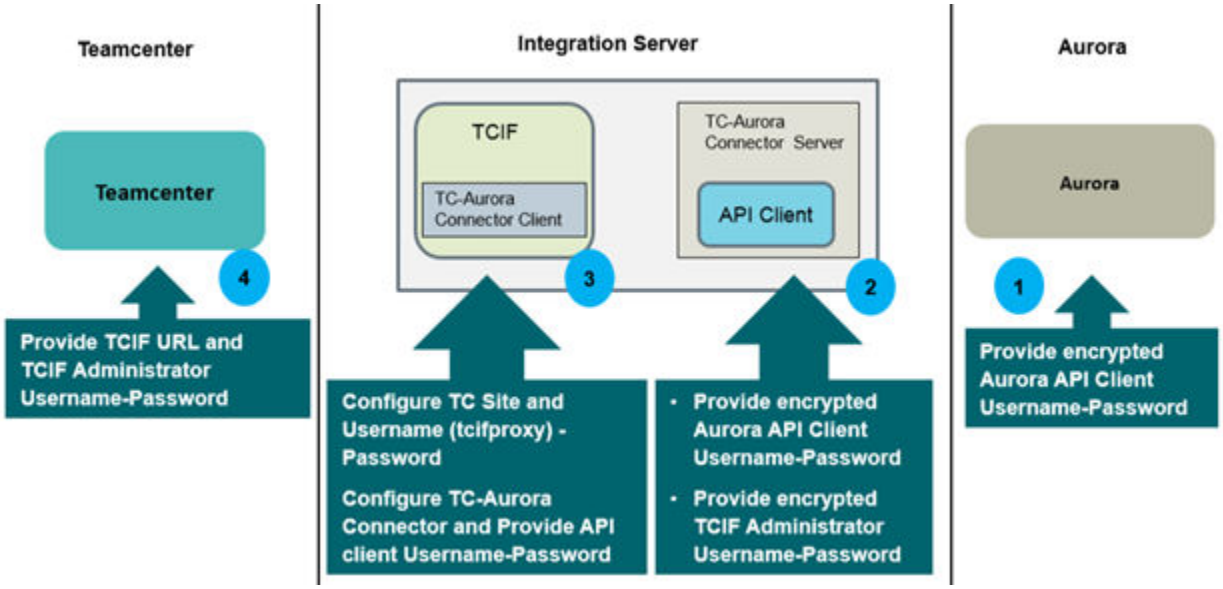
The Teamcenter-Aurora connector client is bundled with TcIF packages, and the Teamcenter-Aurora connector server is a standalone Java program. If there are multiple active models, you, as the administrator, must run the Teamcenter-Aurora connector server for each model. The client and the server communicate with each other using Remote Method Invocation (RMI).

Teamcenter-Aurora integration user setup



Components	Description and how they work
TcServer-TcIF	The Teamcenter server makes a REST call to TcIF with the basic information of WBS Root. TcIF then makes a SOA call to Teamcenter to pull in all the WBS hierarchy information and passes it on to Aurora to create a schedule model.
TC-Aurora Connector Server	This is a standalone Java program that uses the Aurora API client to communicate with the Aurora server to process various types of requests.
TC-Aurora Connector Client	The Teamcenter-Aurora connector client is packaged as a TcIF bundle. It picks up the request from Teamcenter and passes it on to the Teamcenter-Aurora connector server using RMI.
Integration Server	The server where TcIF and the Teamcenter-Aurora connector server is deployed.
TC Aurora Connector Server-TcIF	Users can export the schedule changes from Aurora to Teamcenter through TcIF using REST.

Teamcenter-Aurora integration user setup



Components	User Setup
Aurora	Aurora API client username and password: These user credentials are set up at the Aurora server end. It is used to authenticate the starting of the API client. When the Teamcenter-Aurora connector server process starts for any new model, it internally starts the Aurora API client for that model. To start a new Aurora API client, it requires the credentials of the API client user. If the user credentials provided do not match those in the Aurora server, the Aurora API client fails to start (that is, the TC-Aurora connector server process fails).
TC-Aurora Connector Server	Aurora API Client username and password: These user credentials are set up at the TC-Aurora connector server location. • These credentials are used when a new TC-Aurora connector server process starts. They are authenticated against those at the Aurora server end. • These credentials are used to authenticate the remote calls (using RMI) made from the TC-Aurora connector client to the TC-Aurora connector server. If the credentials provided with the remote call are the same as those set up in the TC-Aurora connector server, only the remote calls get processed. This is required to allow only authenticated clients to make remote calls from the RMI client to the RMI server. TcIF Admin username and Password: These user credentials are set up at the TC-Aurora Connector server location. A REST call is made from the TC-Aurora connector server to TcIF when data is exported from Aurora to Teamcenter. The TcIF Admin credentials are passed with the REST call and authenticated in TcIF. The export of data starts only if the credentials are correct.
TcIF	Teamcenter username (tcifproxy) and password: TcIF makes SOA calls to Teamcenter to pull or push the data, and these user credentials are used to make SOA calls.

Components	User Setup
	Aurora API Client username and password: TcIF makes remote calls from the TC-Aurora connector client to the TC-Aurora connector server using RMI. The Aurora API client credentials are used to authenticate the remote calls at the TC-Aurora connector server end.
Teamcenter	TcIF Admin username and password: Update the OOTB administrator preference for the TcIF administrator username and password. These credentials are passed with REST calls from Teamcenter to TcIF. The processing only starts if the credentials are correct.

Data security in Aurora

- Follow the instructions in the Aurora Help to set up user and data security.
- Once the data moves out of Teamcenter to Aurora, it follows the authentication and authorization security rules of Aurora.

2. Installing the Aurora integration

Download Aurora

Stottler Henke has created a version of Aurora customized for the IPP&E solution. Download this version of the software from Support Center. Download following software from “Additional Downloads” or corresponding compatible version

1. AuroraServer_windows-x64_Aurora_Siemens_20_0_0_6.zip
2. IPPTCAuroraConnectorBundle_2412.zip
3. Aurora_WebStart_20_0_0_6.zip

Install Aurora

IPP&E uses Aurora as a scheduling application to manage medium to very large programs.

1. Extract *AuroraServer_windows-x64_Aurora_Siemens_20_0_0_6.zip*.
2. Navigate and run *AuroraServer_windows-x64_Aurora_Siemens_20_0_0_6.exe* to install at desired location.
3. Follow all the default picks to install Aurora server.

Aurora Post Installation Task

Go to the <Aurora installation>\config folder and replace the word localhost with the actual hostname in the following files:

- apiClient.ini
- client.ini
- loadingHelper.ini
- metaServer.ini
- server.ini

Configure the Aurora API client user

These user credentials are used to authenticate the starting of the Aurora API client. Starting a new Aurora API client requires the credentials of the API client user. If the user credentials provided do not match those of the Aurora server, the Aurora API client fails to start.

Encode the Aurora API client username and password using the following steps:

1. Go to the Aurora install directory and run %AuroraServer%/Password Encoder.exe.
2. Enter the Username and Password and click Encode. For example, **APIClientAdmin**.
3. After the encoding is successful, click OK and then Close.

> Note:
>
> This username and password will be used later to start the Aurora API client from an external application, namely, TcIF and the TC-Aurora connector server.

Set up Aurora Web Start

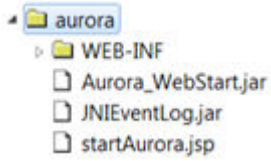
Extract the Aurora Web Start package from IPP&E Aurora Connector 20.0.0.6.

This package consists of the following components:

Files	Purpose
startAurora.jsp	This file is downloaded as a JNLP file when a user starts the Aurora client from a web browser (the AW client). Once downloaded, the user double-clicks this file to download the Aurora client JAR file.
JNIEventLog.jar	This file contains records of the event logs.
AuroraWebStart.jar	This Aurora client JAR file is downloaded when the user starts the JNLP file and automatically launches the Aurora client for a particular model.

> Note:
>
> Aurora Web Start requires Oracle Java 1.8 on the client machine.

1. Extract *Aurora_WebStart_20_0_0_6.zip* at the same machine where webserver is installed. The structure of the Aurora web start deployable package should be as follows:



2. Navigate to startAurora.jsp file and update the port number on line# 50 as per your webserver. The default is 8080.

3. Stop any active running JBOSS services.

4. Copy **aurora** deployable package to the %JBOSS_HOME%\standalone\deployments location.

5. Start JBOSS

Deploy Aurora

You can deploy Aurora using Tomcat. The process of deployment using Tomcat is defined below.

- **Deploy on Tomcat**

  1. Stop any active running Tomcat services.

  2. Copy your Aurora deployable package to the %TOMCAT_ROOT%\webapps\ROOT location.

  3. Start Tomcat webserver services.

Aurora license and user setup

- **Aurora license**

  - Aurora support will supply you with a license definition file, aurora_license. This will define how many named users of each type are available.

  - Save the aurora_license file in your %AuroraServer% directory.

- **Aurora user setup**

  - You must now define the type of license for each user. Certain role configurations will depend on the Aurora options that you select.

    > Note:
    >
    > The Aurora evaluation version does not require licenses, but it still requires user setup.

- Use the Aurora Role Manager to define users. As the desired roles change, use this interface to add, remove, edit, enable, and disable users.

1. Make sure that the Aurora MetaServer is running. If it is not, then start it from the Aurora Server installation directory.

2. Run the Aurora Role Manager from one of the following locations:

   - From the Start menu option, from Aurora RoleManager.exe in the %AuroraServer% directory.

   - From Aurora Role Manager WebStart, if you requested external role manager access.

3. From the Aurora Role Manager, click Add.

4. Enter the user's first and last names (these are for informational purposes only). Then enter the user's Windows ID in the User ID box.

5. (Optional) If desired, enter organization and program category information. This is not relevant to the Teamcenter integration, but may control supplemental customizations within Aurora for multiple clients.

6. Select the appropriate User Tier which will be the user role type. The options are available based on the license file.

User Role	Role Description
Program Scheduler	Allows users to use all Aurora functionality including generation and optimizing schedules and advance analytical reports.
Basic Scheduler	Allows users to make changes and schedule and view the schedule, but not optimize or use any of the analytic reports.
Resource User	Allows users to schedule and analyze progress, and enter actuals, but not make any other changes (primarily for use in an execution schedule).
Viewer	Allows users to run reports and view the model, but not make any changes or schedule.

7. If you wish to have external access to the role manager and/or MetaServer user interface, select User Role Administrator and Aurora Administrator appropriately for the administrative user.

   Note

   As these external access capabilities are not part of the standard package, you need to request the appropriate supplemental WebStarts from Stottler Henke to obtain this functionality.

8. When finished, click Apply and close the role manager window.

9. Restart the Aurora Metaserver.

3. Install and configure TcIF

Install TcIF

Refer **TcIF installation** documentation.

Install the IPP&E Aurora connector package

The following are steps for installing the IPP&E Aurora connector package in TEM:

1. In the **Maintenance panel**, select **Perform maintenance on an existing configuration** and click **Next**.

2. In the **Configuration** panel, select the configuration from which the corporate server was installed and click **Next**.

3. In the **Feature Maintenance** panel, under the **Teamcenter** section, select **Add/Remove** Features and click **Next**.

4. In the **Features** panel, choose **Extensions** - **Integrated Program Planning and Execution (IPP&E)** - **IPP&E Aurora Integration** and click Next.

5. In the **Service Stop** window, click **OK**.

6. If the **Teamcenter Administrative User** panel appears, enter your username and password to log into the server, and then click **Next**.

   The **Database Template Summary** panel lists the installed templates.

7. Click **Next** to display the **Confirmation** panel and click **Start**.

8. When the installation is complete, close TEM.

Start TcIF

A message stating that TcIF has started successfully is displayed.

On TCIF prompt, Press Enter and run the list -t=0 command to check that all the solutions have been deployed properly and verify that there are no errors. Alternatively, you can check the log file located in the %TcIF%/container/log folder.

Configure the Teamcenter site in TcIF

1. To configure the Teamcenter site, go to *<hostname>*:*<rest-services-port>*/**tcif/rest/login** and log on using the following credentials:

   - **Username**: **IFAdmin**
   - **Password**: **admin**

2. Choose **Configure** - **Sites**.

3. From the **Site** homepage, click **New**.

4. Go to the *%TC_ROOT%/fsc/fmsmaster_FSC_hostname_<installing>_<user>.xml* file and copy the value of the **enterpriseid site identifier**. Enter this value in the **Teamcenter Site ID** box. For the **Site Type**, choose Teamcenter from the list and click **Next**.

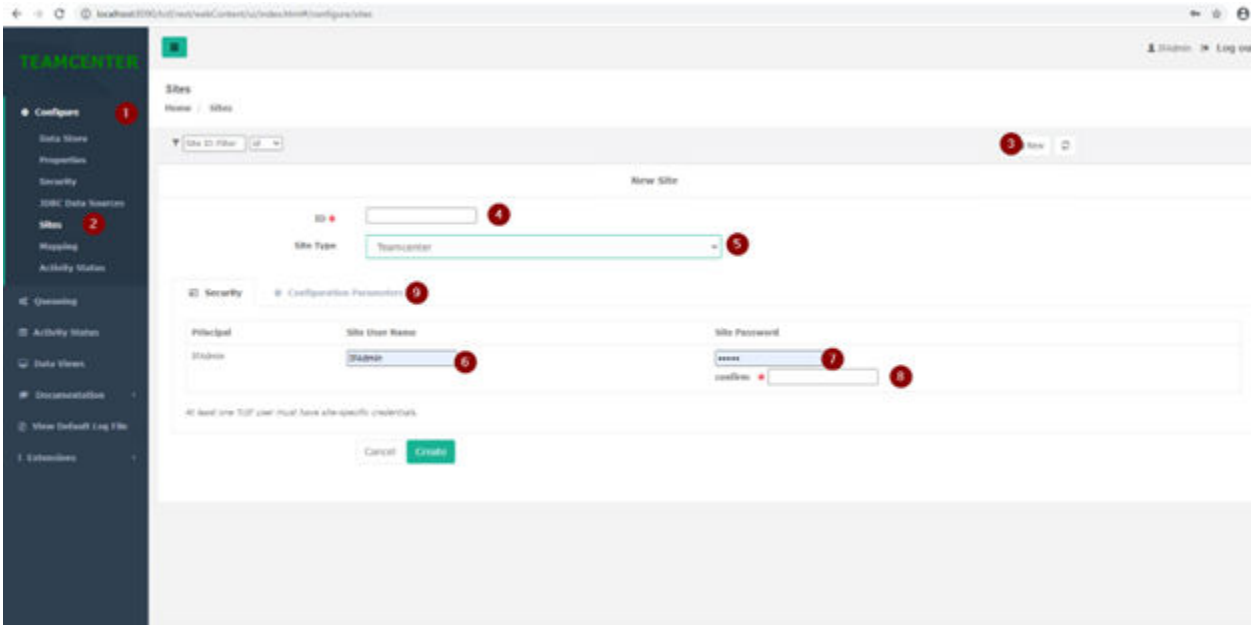
5. From the Security tab, enter the site credentials for the Teamcenter login.

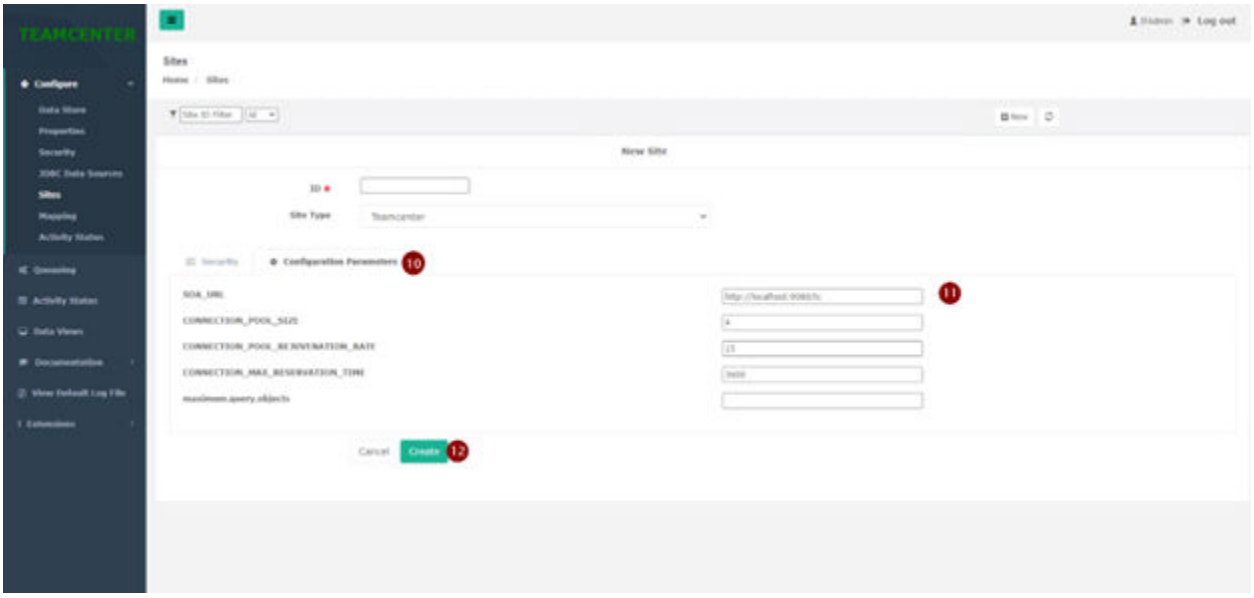
   - **Site User Name**: **tcifproxy**.

   Site Password: **<password>**

6. From the **Configuration Parameters** tab, enter the **SOA_URL** and click **Create** to create the site.

> Note:
>
> Replace **localhost** with your hostname and port number.



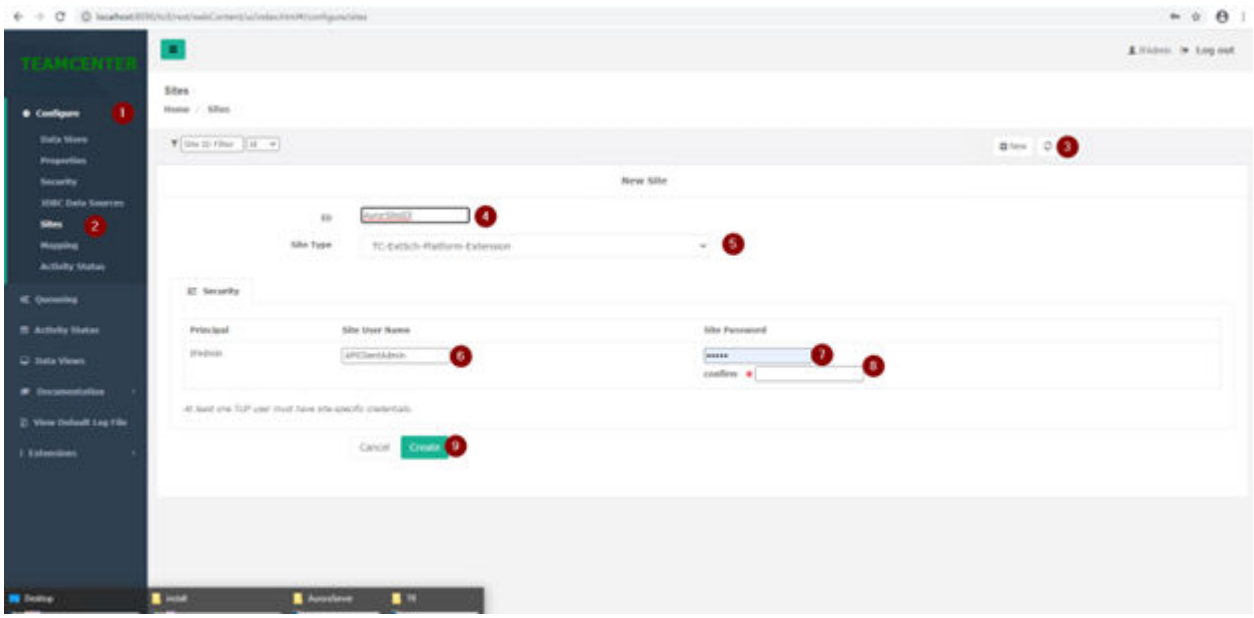


Configure Aurora for TcIF

1. Click **New** from the **Site** homepage to create another site for Aurora.

2. Enter **AuroraSiteID** as the **Site ID** and select **TC-ExtSch-Platform-Extension** as the **Site Type**.

3. From the **Security** tab, enter the site credentials for starting the Aurora API client. These credentials are the same as those specified when encoding the Aurora API client password. Click **Create**.



Configure email in TcIF

1. Choose **Configure** - **Properties**.

2. Expand **camel** from the list of **Property Names** and in the **Properties** section under it, update the **request.timeout** and **emailservice.smtp-uri** values.

   For example,

   request.timeout: 180000

   emailservice.smtp-uri: smtp://your_company_mail_server_com?
   from=admin@company_name.com&to=admin.@company_name.com

   Enter a valid admin email ID for both **From** and **To** arguments.

3. Click **Save changes** and log out.

Complete TcIF post-installation tasks

Create Teamcenter user for TcIF nas follows:

1. Log on to the Teamcenter Rich Client as an administrator and open the **Organization** application.

2. Create a user with the following details:

   - **User Name: tcifproxy**
   - **Group: dba**
   - **Role: DBA**

Update access control

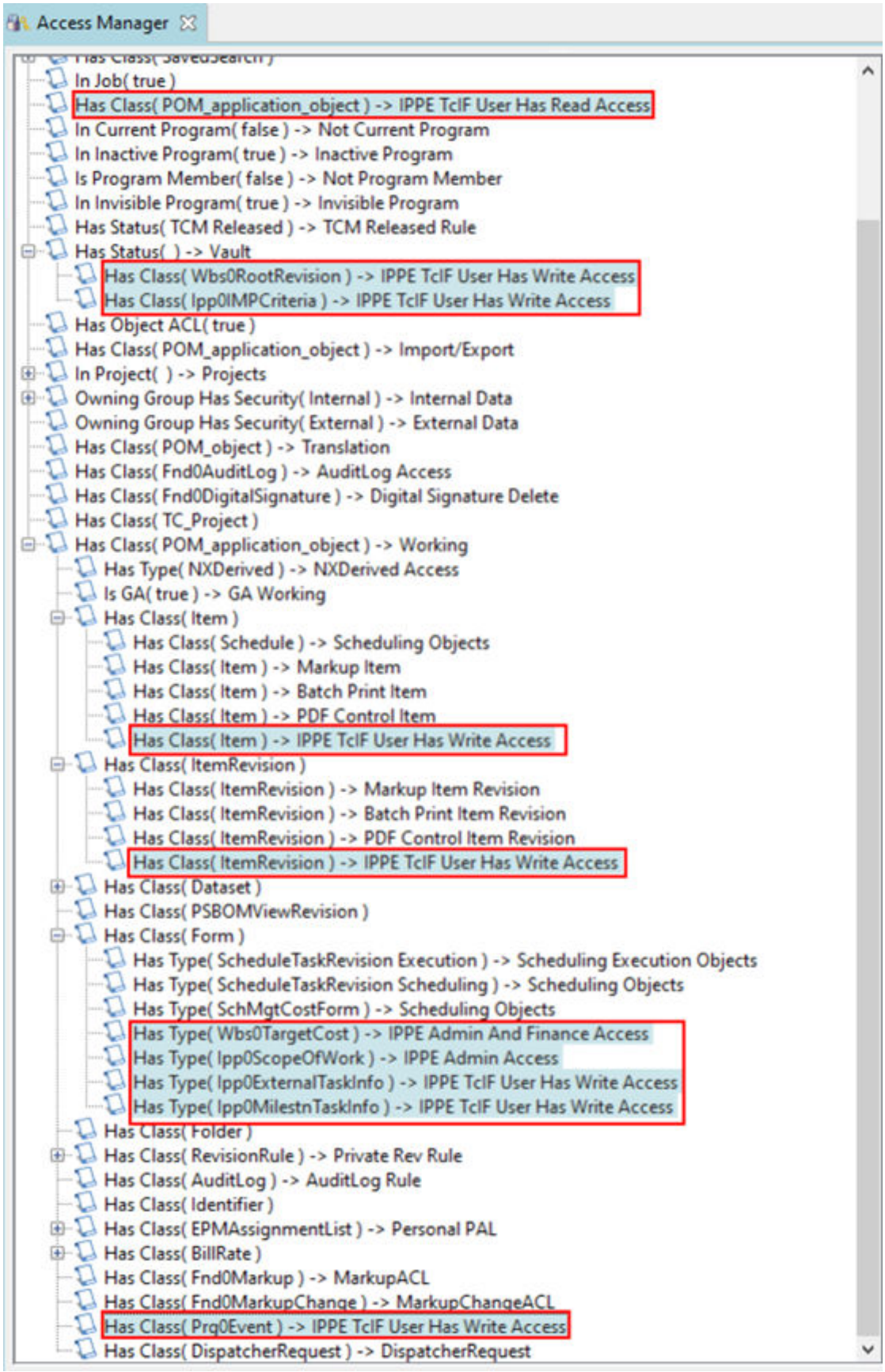
1. Log on to the Teamcenter Rich Client as an administrator and open the **Access Manager** application.

2. Create the following ACLs:

ACL Name	Accessor Type	Accessor ID	Access
IPPE TcIF User Has Read Access	User	tcifproxy	Read
IPPE TcIF User Has Write Access	User	tcifproxy	Write

ACL Name	Accessor Type	Accessor ID	Access
IPPE Admin And Finance Access	Role in Projects of Objects	Corporate Program Manager	Read Write Change ownership
	Role in Projects of Objects	Program Manager	Read Write Change ownership
	Role	Corporate Finance	Read Write
IPPE Admin Access	Role in Projects of Objects	Corporate Program Manager	Read Write Change ownership
	Role in Projects of Objects	Program Manager	Read Write Change ownership

3. Update the rule tree as follows:



Note:

If any accessors or roles are not available, then you need to create those manually.

Set up background processing for workflows

You must enable and configure the Teamcenter Dispatcher to set up background processing for workflows.

1. Set the **EPM_task_execution_mode** preference to **CONFIGURABLE**.

   This enables the tasks for which the **Process in Background** box is selected in the workflow process template to run in the background.

2. Create an access control list (ACL) named **DispatcherRequest** as part of the post-dispatcher translator configuration.

3. Add the **DispatcherRequest** ACL under the **Has Class (POM_application_object)** → **Working** branch in **Access Manager**.

4. In the **Organization** application, update the **Site Node/URL** and **SOA URL** to the web-tier Teamcenter URL (for example, http://servername:port_number/context_name).

   a. Select sites.

   b. Enter the **SOA URL** pointing to the web-tier URL.

   The **Site ID** displays in the **User Settings** of the **My Teamcenter** application.

Set email preferences

1. For the **Mail_server_name** preference value, enter the name of the email gateway server.

   The server should be running a **sendmail** daemon. The name can be a text string or a numeric IP address.

2. For the **SiteTimeZone** preference value, enter the server's time zone identifier in the following format: **region/city**

   Example: America/New_York

   For the complete list of zone identifiers, see the **zoneinfo database**.

3. Set the **Mail_internal_mail_activated** preference value to **false**.

4. Set the **Mail_server_port** preference value to **25**.

5. Set the **Mail_server_charset** preference value to **windows-1252**.

6. Set the **Mail_send_file_attachments_activated** preference value to **true**.

7. Set the **Mail_OSMail_activated** preference value to **true**.

Install the TC-Aurora connector server package

The TC-Aurora connector package consists of the following installable:

Installable	Purpose
IPPTCAuroraConnectorBundle_2412.zip	This is used to start the RMI registry on the configurable port and the RMI server for the Aurora API client.

You must extract the contents of the IPPTCAuroraConnectorBundle_2412.zip and place it parallel to the TC_ROOT directory, or parallel to the TcIF installation directory if TcIF is installed standalone. The contents of the package are as follows:

Folder	Description
scripts_wntx	This directory contains the following: • Scripts and jar files to run the TC-Aurora connector server (which internally runs the Aurora API client). • Jar file to encrypt the TcIF administrator login and Aurora API client password mentioned in the Credentials.txt file.
config.cfg	This is the file that contains various required configurations to run the TC-Aurora connector server for any model. These configurations are common for all the models.
aurora.bat	Batch file to start or stop the TC-Aurora connector server for all the models (WBS Root ID from Teamcenter) specified in the Startaurora.ini/Stopaurora.ini file.
startaurora.ini	When the Aurora.bat script is executed with the start option, this file is referenced by the script to start the TC-Aurora connector server for all the models mentioned in this file.
stopaurora.ini	When the Aurora.bat script is executed with the Stop option, this file is referenced by the script to stop the TC-Aurora connector server for all the models mentioned in this file.

> Note:
>
> The Aurora.jar file is available in the <AuroraServerInstallDir> location, and this file should be the same in IPPTCAuroraConnectorBundle_2412\scripts_wntx\lib. If it is not the same, copy it from <AuroraServerInstallDir> to IPPTCAuroraConnectorBundle_2412\scripts_wntx\lib.

Encryption of TcIF and Aurora user credentials

The IPPE TC-Aurora connector package contains the scripts_wntx/encryptpass/IPPEncrypter.jar file, which encrypts the username and password present in the credentials.txt file. Typically, the credentials.txt file contains the credentials for the TcIF Admin and the Aurora API client.

1. Open the IPPE TC-Aurora Connector/scripts_wntx/encryptpass folder.

2. Edit the credentials.txt file to update the username and password for the Aurora API client and TcIF Admin.

Attribute Name	User Action
aurora-username	Enter the same Aurora API client username that was set during Aurora server configuration.
aurora-password	Enter the same Aurora API client password that was set during Aurora Server configuration.
tcif-login-username	Enter the TcIF Admin username as **IFAdmin**.
tcif-login-password	Enter the TcIF Admin password as **<Admin_Password>**.

3. Go to the IPPE TC-Aurora Connector/scripts_wntx/encryptpass folder and open the command line to set **JAVA_HOME** and **JRE_HOME**. Use Java 8 or above for this utility and be sure **JAVA** is set in the **PATH**.

4. Encrypt the credentials.txt file with IPPEncrypter.jar using the following command:

```
java -jar IppeEncrypter.jar credentials.txt credentialsCypher.properties
```

   The encrypted file encrypted.enc is generated in the same directory.

> Note:
>
> The encrypted.enc file is used when the TC-Aurora connector server is executed for any model.

Update config.cfg to run Aurora.bat

IPPTCAuroraConnectorBundle_2412/config.cfg file contains the following:

> Note:
>
> Use **/** instead of **** while specifying file path in config.cfg

Attribute	User Action
teamcenter-site-id	Enter the Teamcenter site ID. The site ID should be same as that configured in TcIF. Go to the *%TC_ROOT%/fsc/fmsmaster_FSC_hostname_<installing>_<user>.xml* file and copy the value of the enterpriseid site identifier
rmi-port	Specify the RMI port on which the RMI registry will start. The TC-Aurora connector client and the TC-Aurora connector server communicate with each other using this RMI port, for example, 5000.
aurora-config-file-path	Specify the file path of *apiClient.ini*. This file can be located under *<AuroraServer>/config*. If the Aurora server is installed on the same server, the user can specify the same path. If installed on a different server, copy the file from the Aurora server location and paste it where the TC-Aurora connector is installed to specify the path.
credential-encrypt-file-path	Enter the file path of the *encrypted.enc* file that was generated during TcIF Admin and Aurora API client password encryption. Update the path to *<tc-aurora-connector>/scripts_wntx/encryptpass/encrypted.enc*.
cipher-properties-file-path	Enter the file path of the *credentialsCypher.properties*. Update the the path to *<aurora-connector>/scripts_wntx/encryptpass/credentialsCypher.properties*
aurora-saved-model-dir	Create a **Saved_Models** directory and specify the path. Make sure that it has read and write access. This directory is used to save the models as *cmp* files by the Aurora API client. Default path is: *<tc-aurora-connector>/scripts_wntx/config/Saved_Models*
tcif-rest-base-URL	Enter the Tcif REST URL. This value should be same as the value of the Teamcenter **TcIF_REST_BASE_URL** preference. Note: http://hostname>:8090/tcif/rest
use-default-queue-for-export	Enter **Yes** or **No**. This configuration identifies whether to use the default queue **IPPEQueueForDataExchange** or to create a new queue for each model in Teamcenter and name the queue with the Model ID.
ExtSch-TCProperty-mapper-file-path	This is the mapper file which gives the information about the properties mapped from Teamcenter to Aurora. Enter the file path where *PropertyMapper.cfg* is located.

Update aurora.bat

Open the *<TC-Aurora-Connector-server>\aurora.bat* file and update **JAVA_HOME**, **PERL_HOME**, and **CONFIG_LOCATION**

> Note:
>
> Check for perl under the **TC_ROOT** folder. The **CONFIG_LOCATION** is the location of the *<TC-Aurora-Connector-server>\config.cfg* file.

TC-Aurora Connector logging: To get detailed information about the errors and warnings in the Aurora log files, turn the logging level to **DEBUG** by changing the **AURORA_API_DEBUG** variable to **ON** in the *aurora.bat* file.

Prerequisites to start the TC-Aurora connector server

Make sure that Aurora is installed, TcIF is configured, and the config.cfg file contains all valid entries. Open the <TC-Aurora-Connector-server>\aurora.bat file and update **JAVA_HOME, PERL_HOME**, and **CONFIG_LOCATION**.

> Note:
>
> Check for perl under the **TC_ROOT** folder. The **CONFIG_LOCATION** is the location of the config.cfg file.

TC-Aurora Connector logging:

By default, TC-Aurora connector log files are generated in the <TC-Aurora-Connector-server>\logs directory. Default logging level is set to **INFO**.

To get detailed information about the errors and warnings in the Aurora log files, turn the logging level to **DEBUG** by changing the **AURORA_API_DEBUG** variable to **ON** in the aurora.bat file.

Configure TcIF to automatically start the TC-Aurora connector server

1. Open the Teamcenter Integration Framework web interface in a browser.

   <hostname>:*<rest-services-port>*/**tcif/rest/login**

2. In Teamcenter Integration Framework, choose **Configure** - **Properties**.

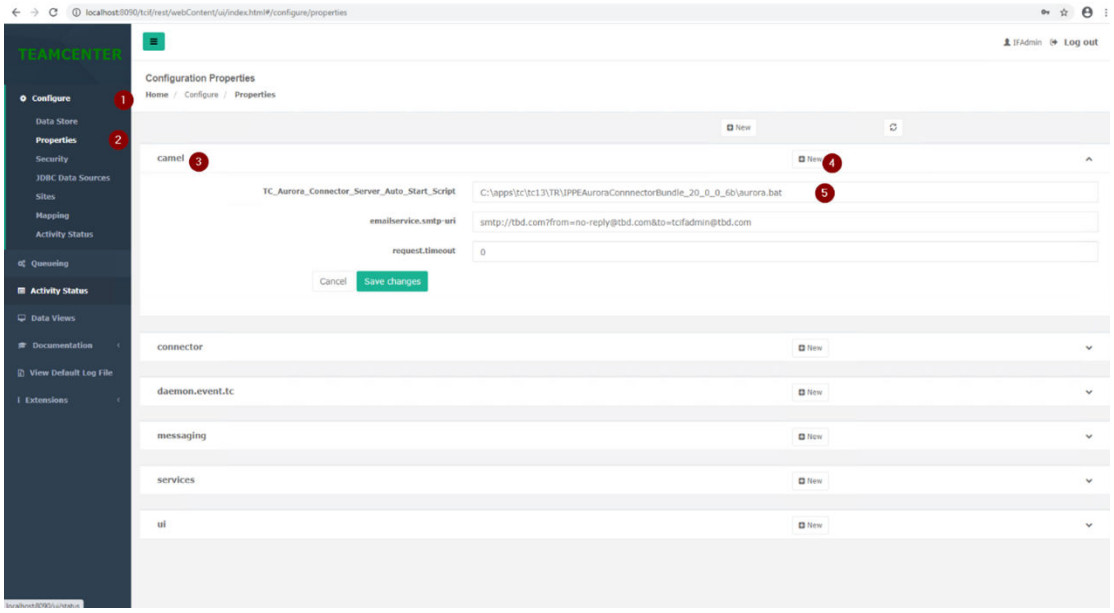
3. Select the **camel** property section and click to add a new property.

4. Enter **TC_Aurora_Connector_Server_Auto_Start_Script** as the property name. The value for this will be the path to the *aurora.bat* script located inside the TC-Aurora Connector server package.

5. Click **Save changes** to save the new property.

Note:

If this property is not set, the TC-Aurora connector server will not start automatically. When data is exported from Teamcenter, it expects that the TC-Aurora Connector server for the model is already started along with other required services. Refer to the troubleshooting tips for more details.



4. Configuring the Aurora integration

Verify the Teamcenter Aurora integration

1. Ensure that no errors come up when the WBS root element is submitted to the workflow.
2. Ensure that there are no errors in the TcIF console.
3. Check for the following message in the TcIF console window:

   Checking for RMI REGISTRY status.

4. Ensure that the Aurora RMI Registry is open and a Windows process with the WBS-ID is open.
5. Once the previous step is complete, ensure that the WBS-ID session is created on the Aurora Metaserver under joinable sessions.
6. Open the Aurora client and ensure that the session is listed under joinable sessions.
7. On the Active Workspace client, ensure that the **Exported to Baseline** property is set to **True**.

Configure Teamcenter for TcIF and Aurora

To configure TcIF and Aurora in Teamcenter, you must modify certain preferences in Teamcenter.

1. Log on to Teamcenter as **dba**.
2. Choose **Edit** → **Options**.
3. In the **Options** window, click **Search**.
4. From the **Preferences** list, modify the following preferences:

Preference name	Value
IPP_aurora_client_url_for_model	http://<HOSTNAME>:<PORT>**>** **/aurora/startAurora.jsp?session_type=** **join&session_to_join=**
TCIF_REST_BASE_URL	**http://***<HOSTNAME>***:***<PORT>***/tcif/rest**
TCIF_USER_NAME	**IFAdmin**
TCIF_USER_PASSWORD	**admin**

Preference name	Value
IPP_TCIF_EXTERNAL_SCHEDULE_CONNECTOR_RMI_PORT	5000 This value should be the same as the **rmi_port** value defined in the *config.cfg* file.
IPP_tcif_number_of_queues	2

Configure the property mapper for data exchange between Teamcenter and Aurora

When data is transferred between Teamcenter and Aurora, certain property values are always transferred by default, for example, WBS number and element name values are exported from Teamcenter to Aurora. Likewise, when importing data from Aurora to Teamcenter, start date, end date, and duration values are always transferred. If you want to transfer the value of other properties, you can configure the solution to transfer those properties.

Examples of configuring the property mapper for data exchange follows:

- Only work package property values can be transferred from Teamcenter to Aurora and the solution updates the task or activity property in Aurora. To configure this, update the **WBSElemRev.ExportPropertyToExternalSchedule** preference, which contains a map of properties in Teamcenter and the corresponding property in Aurora. Each entry specifies a property in Teamcenter that should update the value of a property in Aurora. For example:

  wbs0PlanObject: originating Model

  In this example, **wbs0PlanObject** is a property of the Work Element Revision whose value is to be mapped to the **originating Model** property in Aurora.

- Configure additional property values to transfer from Aurora to Teamcenter by updating the **PropertyMapper.cfg** configuration file in the TC-Aurora-Connector package. This file contains the properties that are mapped to Teamcenter and its path is specified in the **config.cfg** file in the package.

  Only activity and task property values in Aurora can be transferred to Teamcenter, and these properties can only be mapped to **ipp0ExternalTaskInfo** properties. This example of shows a property mapping in the **PropertyMapper.cfg** file.

  object_desc = description

  In this example, **description** is the property of the task in Aurora and its value is mapped to the **object_desc** property of the task **Info** object in Teamcenter.

5. Using the Aurora integration

Export WBS data to Aurora for first time

1. In Active Workspace, select a WBS root element to be exported to Aurora from the WBS structure.
2. Click **More commands** ••• > **Manage** > **Submit to Workflow**.
3. Select the **All** option.
4. In the Template box, enter **IPPE Export Root Info**.
5. Click **Submit**.

   The WBS root element is submitted to the workflow.

Export modified WBS data to Aurora

1. In Active Workspace, select a WBS root element from the WBS structure that is modified.
2. Click **More commands** ••• > **Manage** > **Submit to Workflow**.
3. Select the **All** option.
4. In the Template box, enter **IPPE Export Delta Info**.
5. Click **Submit**.

   The WBS root element is submitted to the workflow.

Open the Aurora schedule from Active Workspace

Once the WBS structure is exported to Aurora, you can open the Aurora schedule from Active Workspace.

1. In Active Workspace, open the WBS structure in a new tab or new window.
2. Click **More commands** ••• > **Open** > **Open Aurora Schedule**.

   The startAurora.jnlp file is downloaded.

3. Click **Keep** when the browser asks if you want to keep the file.

4. Click the downloaded startAurora.jnlp file.

   A java web start application is launched. The Aurora web client opens for the WBS structure that was exported. Using this client, you can make a schedule and process other related information for the WBS structure.

Export the Aurora schedule to Teamcenter

Teamcenter IPP&E provides the ability to read the scheduling data from Aurora. A user can export the scheduling data back to Teamcenter and notify Teamcenter of all the modifications. Active Workspace is then able to display the data on the appropriate object.

1. In Aurora, click **File** → **Save** to save the scheduling data which is processed on the model.

2. Select the WBS root, choose **File** → **Import/Export**, and then choose the option to notify Teamcenter of all updates, including model and schedule changes.

   Aurora sends the schedule information to Teamcenter.

Export WBS data for special scheduling cases

Teamcenter IPP&E provides the ability to schedule the WBS data in special scheduling cases such as best case, worst case, and most likely case.

Best case schedule

The WBS can be scheduled in the *best case* mode. This mode considers all the best-case scenarios (for example, opportunities) while scheduling, and ignores all other scenarios.

1. In Active Workspace, select a WBS root element from the WBS structure that is modified.

2. Click **More commands** ••• > **Manage** > **Submit to Workflow**.

3. Select the **All** option.

4. In the **Template** box, enter **IPPE Export Best Case Schedule Info**.

5. Click **Submit**.

   The WBS root element is submitted to the workflow.

Worst case schedule

The WBS can be scheduled in the *worst case* mode. This mode considers all the worst-case scenarios (for example, risks) while scheduling, and ignores all other scenarios.

1. In Active Workspace, select a WBS root element from the WBS structure that is modified.
2. Click **More commands** ••• > **Manage** > **Submit to Workflow**.
3. Select the **All** option.
4. In the **Template** box, enter **IPPE Export Worst Case Schedule Info**.
5. Click **Submit**.

   The WBS root element is submitted to the workflow.

Most likely case schedule

The WBS can be scheduled in the *most likely* case mode. This mode considers all the scenarios (risks and opportunities) based on their probabilities while scheduling.

1. In Active Workspace, select a WBS root element from the WBS structure that is modified.
2. Click **More commands** ••• > **Manage** > **Submit to Workflow**.
3. Select the **All** option.
4. In the **Template** box, enter **IPPE Export Most Likely Case Schedule Schedule Info**.
5. Click **Submit**.

   The WBS root element is submitted to the workflow.

6. Managing, testing, and troubleshooting the Aurora integration

Start and stop the TC-Aurora Connector server for a model

This is not an installation or a configuration step.

This step is required when a Teamcenter user is ready to export data from Teamcenter to Aurora for any new model. The administrator must start the TC-Aurora Connector server for that model, which will internally start the Aurora API client for the same model. Once the TC-Aurora connector server for a model is running, the data can be exported from Teamcenter to Aurora and vice-versa. To simplify the TC-Aurora connector server start and stop process, IPP&E has provided some scripts for the administrators.

The IPPE TC-Aurora Connector package contains the following .ini files:

- startAurora.ini
  - This file contains the model entries for which the TC-Aurora connector must be started. If the data for a new model is ready to be exported, the administrator can update this file with that model.
- stopAurora.ini
  - This file contains the model entries for which the TC-Aurora connector has to be stopped. To stop the TC-Aurora connector server for any model, the administrator must update this file and remove the entry of that model from the startAurora.ini file.

The format of these model entries is <Model_ID>|<Model_Name>, where Model ID is the Project WBS Root ID and the Model Name is the Project WBS Root name.

Start and stop the TC-Aurora connector server as follows:

1. Open the Aurora.bat file and update JAVA_HOME, PERL_HOME and CONFIG_LOCATION, where CONFIG_LOCATION is the location of the config.cfg file.

2. Update the config.cfg file to make sure it contains all valid entries.

3. In a command window, navigate to directory location that contains the Aurora.bat script.

4. Run the Aurora.bat script.

   This script starts the RMI registry if it is not started with the port set in the config.cfg file.

   The options for the Aurora.bat script are as follows:

Option	Use
start	For this argument, the script starts the TC-Aurora connector server process for all the models that are listed in the startaurora.ini file. It ignores the models for which the TC-Aurora connector is already running.
stop	For this argument, the script will stop the TC-Aurora connector server process for all the models that are listed in the stopaurora.ini file.
stopall	This option is used to stop all the TC-Aurora connector server processes that are running and to remove them from the registry.
startmodel <Model_ID| Model_Name>	This option is used to start the TC-Aurora connector server for a specified model.

Note:

It is recommended that the administrator updates the model details in the startaurora.ini file as well so that if the Start option is used to start the TC-Aurora connector server for all the models after all the services are shutdown (for example, because of maintenance), it will also pick up the models that were individually started.

Note:

The Aurora.bat script internally uses PowerShell. It is important to have the latest version of PowerShell. If the TC-Aurora Connector server is already running for any model, the script will not start another session. If the PowerShell version is old, this functionality will not work, in which case, the administrator must ensure that there are no duplicate TC-Aurora connector sessions.

Performance tuning

You may need to consider performance factor when it comes to starting hundreds of Aurora clients. By default, the Java heap allocation size for Aurora startup is set to -Xmx256m. You might want to increase or decrease the size based on your needs.

To make the changes, update the AuroraRMI.bat file with the new Java heap size value.

Create Teamcenter Integration Framework queues

This section covers the basic details of managing TcIF queues and jobs. For more details, refer to the TcIF documentation. IPP&E provides two mechanisms for queue usage. One option is to use only one queue for all data exchange. The second option is to create separate queues for each model, so that when data is exported for a particular model, it will use the queue with the name of its Model ID.

IPP&E has provided the capability to automatically create a queue for each model (WBS Structure) in TcIF. If the automatic creation of a queue fails, TcIF administrators can also create the Teamcenter Integration Framework queues using the manually as follows:

1. Run trun.bat from the %TcIF%/container/bin folder or double-click the Teamcenter Integration Framework icon from the desktop.

   Open the Teamcenter Integration Framework web interface by browsing to *<hostname>*:*<rest-services-port>*/**tcif/rest/login**.

2. In the Teamcenter Integration Framework web interface, click **Queueing** to display the current queues.

3. Click New. The default settings for the queue are displayed.

4. Enter a name for the queue. The queue name must be unique in the Teamcenter Integration Framework cluster.

5. On the **General Settings** tab, update the queue settings as necessary for the queue.

   Stuck Time Out

   Specifies the length of time, in minutes, before an unprocessed job is moved to the stuck state.

   Hung Time Out

   Specifies the length of time, in minutes, a job remains in the stuck state before being moved to the failed state.

   Automatic Retries

   Specifies the number of attempts to make to retry a failed job. If the job does not succeed after this number of retries, it is sent to the dead letter queue (if available).

   Initial Retry Delay

   Specifies the length of time, in milliseconds, to pause before retrying a failed job.

   Dead Letter Queue

   When set to **On**, a dead letter queue is available for this queue.

   Parallelism

   Specifies the number of messages that can be processed simultaneously. Siemens Digital Industries Software recommends setting **Parallelism** to a value of **3** or the total number of processors, whichever is greater.

   Setting **Parallelism** to **1** specifies that messages are processed sequentially.

   Processor

   Specifies the Groovy script that processes messages in the queue.

6. Click **Create**. The queue is created.

7. Locate the new queue in the list of queues. Click ⏸ next to the queue name and then click ▶ Activate to start the queue.

Manage Teamcenter Integration Framework jobs

Administrators can manage Teamcenter Integration Framework queues and messages using the Teamcenter Integration Framework web interface.

To perform the following queue management tasks, first open the Teamcenter Integration Framework web interface by browsing to *<hostname>*:*<rest-services-port>*/**tcif/rest/login**.

List the jobs in a queue

1. In the Teamcenter Integration Framework web interface, click **Queueing** in the left pane. The currently configured queues are displayed.

2. Click ☰ for the queue containing the jobs you wish to view. The currently queued jobs are listed along with their current statuses.

Change job states

1. In the Teamcenter Integration Framework web interface, locate the job you want to manage.

2. Change the job state as follows:

   Pause jobs

   Click ▶ next to the job name and choose **Pause**.

   Stop jobs

   Click ▶ next to the job name and choose **Stop**.

   Advance jobs to the next state

   Click ⊕ to move the job to the next state.

   Restore jobs in the dead letter queue

   Jobs in the dead letter queue are identified with a 💣 symbol. Click ♻ to restore a job in the dead letter queue.

   Delete (purge) jobs

   Click 🗑 to remove a job from the queue. Only jobs in paused or failed states can be purged from a queue.

Manage job properties

Administrators can modify certain properties of queued jobs, such as a job's priority relative to other jobs in the queue, a job's publisher, and a job's time out settings.

1. In the Teamcenter Integration Framework web interface, locate the job you want to manage.
2. Click ⌄ for the job you wish to view. Detailed information about the job is displayed.
3. Click the **Properties** tab. A listing of the available job properties is displayed.
4. Modify the properties as necessary.
5. Click **Save Changes**. The job is updated with the changes.

Be aware that synchronous messages can be affected by the value of the **camel** area's **request.timeout** property. **request.timeout** specifies the length of time (in milliseconds) the server will wait for a message reply. By default, **request.timeout** is set to `0`, meaning there is no timeout for synchronous messages. Side effects may occur when synchronous messaging requests are made and timeouts are set too low.

The most likely side effect is receiving unexpected responses because processes did not complete in time. Also, messages may remain unprocessed in the reply queue.

Paused synchronous jobs may also show side effects. If a synchronous job is paused, it is moved to another queue. Consequently, the response returned from the server may be the request message that was sent.

If you set **request.timeout** to a value other than `0`, specify a large enough value to avoid possible side effects.

Manage Teamcenter Integration Framework queues

Administrators can manage Teamcenter Integration Framework queues and messages using the Teamcenter Integration Framework web interface.

To perform the following queue management tasks, first open the Teamcenter Integration Framework web interface by browsing to *<hostname>*:*<rest-services-port>*/**tcif/rest/login**.

Pause and restart a queue

1. Locate the new queue in the list of queues.
2. Review the jobs currently in the queue. Pause any jobs with a pending status.

3. Click next to the queue name and then click Pause to pause the queue. All pending jobs in the queue are paused until the queue is restarted.

4. Click next to the queue name and then click Activate to restart the queue. Pending jobs will be processed in priority order.

Manage queue properties and settings

Administrators can modify certain settings and properties of queues, such as queue priority, the number of job retries, retry intervals, and so on.

1. In the Teamcenter Integration Framework web interface, locate the queue you want to manage and pause the queue.

2. Click for the queue you wish to manage. Detailed information about the queue is displayed.

3. Review and update the settings on the **General Settings** and **Properties** tabs as necessary.

4. Click **Save Changes**. The queue is updated with the changes.

5. Restart the queue.

Manage the dead letter queue

Some jobs may not be processed due to the target system being unavailable, network errors, or other environmental issues. After a specified number of failed retry attempts, a job is moved to a storage queue (a dead letter queue) if its target queue has the **Dead Letter Setting** setting enabled. Once issues causing the failure are addressed, the job can be restarted.

In the Teamcenter Integration Framework web interface, locate the job you want to restart. Jobs in the dead letter queue will be listed in their target queue, identified with a symbol. Click to restore a job in the dead letter queue.

Delete a queue

1. In the list of queues, locate the queue to be deleted.

2. Let all jobs in the queue complete, or cancel any jobs in the queue.

3. Once the queue is empty of jobs, pause the queue.

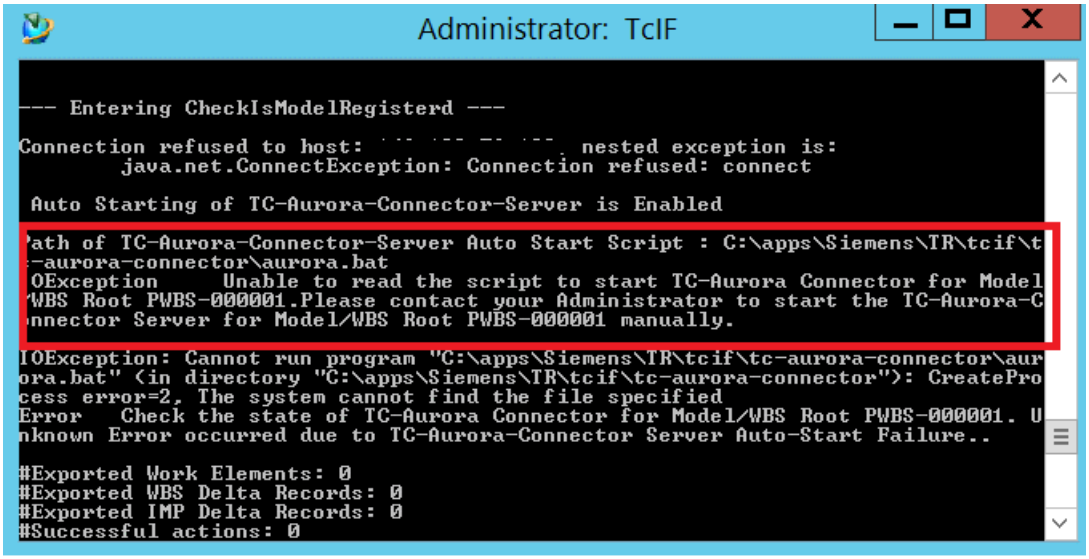
4. Click for the queue to delete the queue.

Fail to auto start TC-Aurora connector server

To export any WBS structure data (model) from Teamcenter to Aurora, the TC-Aurora connector server must be up and running for that model. IPP&E has provided the capability to auto start the TC-Aurora connector server for the model. However, if the server fails to start, use the following steps to help with troubleshooting:

> Note:
>
> You can disable the auto start by removing the TC_Aurora_Connector_Server_Auto_Start_Script entry from the TcIF camel properties. Without this entry, the administrator must manually start the TC-Aurora connector server for each model.

1. Look for an error message in the TcIF window. In this example, the path to the batch file which automatically starts the Tc-Aurora connector server is wrong. There could be other reasons for an error, too.



2. Verify that TcIF and the TC-Aurora connector is installed on the same server.

3. Verify that TcIF is up and running. If required, you can restart TcIF using the diag command to ensure there are no errors.

4. In the TcIF console, run the list to verify that the bundles are resolved and active or not. Correct the location of the placed bundles if necessary. Restart TcIF after making corrections and verify the status using the list command.

5. Verify that the TC-Aurora connector site is configured in TcIF with the correct user credentials.

6. Verify that the path of the aurora.bat file (included in the IPPTCAuroraConnectorBundle_2412 package) which starts the TC-Aurora connector server is set correctly in TcIF camel properties.

a. Log into the Teamcenter Integration Framework WebConfig tool.

b. Click Configure - Properties.

c. Expand the Camel section and check the path in the TC_Aurora_Connector_Server_Auto_Starts box.

The path should not contain any special characters.

7. Verify that the email server is set correctly in TcIF.

8. Verify that the API Client Admin password on the TC-Aurora connector server is set correctly. It must match the API client admin password set on the Aurora Server, as well as the credentials in the TcIF configuration.

9. Verify that the config.cfg file in the TC-Aurora connector server has all valid entries.

10. Verify that the PERL_HOME, JAVA_HOME, AURORA_API_DEBUG, and CONFIG_LOCATION values are set correctly in the aurora.bat file under the TC-Aurora connector server location.

11. Verify that the RMI port value is updated in the Teamcenter preference and that it is the same as entered for the rmi-port variable in the config.cfg.

    - Teamcenter preference:

    IPPE_TCIF_EXTERNAL_SCHEDULE_CONNECTOR_RMI_PORT

    - Entry of rmi-port in the config.cfg file.

12. TcIF waits for a maximum of five minutes to auto start the TC-Aurora connector server. If there is a delay in auto start, it will time out.

13. Verify that the Aurora Meta server is up and running.

14. Verify that TcIF is installed by the user with administrator access. An administrator user must start the TcIF services.

Fail to export schedule data from Aurora to Teamcenter

When data is exported from Teamcenter to Aurora, the data flows through different levels before finally getting exported to Aurora.

1. A REST call is made from Teamcenter to TcIF with WBS Root information.

2. Using the WBS Root information, TcIF connects to Teamcenter and pulls all the data using SOA.

3. After pulling the data, TcIF uses the TC-Aurora connector client to push data to the TC-Aurora connector server using RMI.

4. The TC-Aurora connecter server exports the data in Aurora using the Aurora API client. The communication between the Aurora API client and the Aurora server is through JMS.

If data fails to export from Teamcenter to Aurora, there could be an issue at any level. The following are steps for troubleshooting the different components:

1. Troubleshoot export failure in Teamcenter.

   - Verify that TcIF is up and running.
   - Verify that the TcIF REST URL value in the TCIF_REST_BASE_URL Teamcenter preference is correct.
   - Verify that the TCIF_USER_NAME and TCIF_USER_PASSWORD Teamcenter preferences have the correct values.
   - Verify that the port for the TcIF REST URL is not used in any other services.

2. Troubleshoot export failure in TcIF

   - Verify that the Teamcenter site is configured in TcIF with accurate tcifproxy user credentials.
   - Verify that a queue was created with the WBS root ID in TcIF. These queues are generally created by IPP&E automatically.
   - Check for errors in the queue for that model in TcIF.
   - Verify that there are enough Teamcenter warm servers to process the SOA request from TcIF.
   - Verify that the TC-Aurora connector server is configured in TcIF with the correct Aurora API client admin credentials.
   - Verify that the RMI register is running on the configured port. If the TC-Aurora connector server is configured with auto start, it auto starts the RMI registry as well. However, if the RMI registry fails to auto start, an admin must manually start it.
   - Verify that the encrypted.enc path is updated correctly in the config.cfg file.

3. Troubleshoot export failure in TC-Aurora connector.

   - Verify that the RMI port mentioned in the Teamcenter preference is the same as in the TC-Aurora connector config.cfg file.

- Verify that the TC-Aurora connector server with the WBS Root ID is up and running. If the TC-Aurora connector is configured for auto start, then it should auto start the connector server for that model. However, if it fails to auto start, an administrator must manually start it.
- Verify that the Aurora metaserver is running.
- Verify that the API client with the WBS Root ID is running by looking in the Aurora metaserver window.
- Verify that the Aurora API client user credentials are the same as the credentials for TcIF, TC-Aurora connector, and Aurora server.

Fail to export schedule data from Aurora to Teamcenter

When data is exported back from Aurora to Teamcenter, it also flows through different levels before getting to Teamcenter.

1. A user performs the export action in the Aurora GUI client which triggers an event in the TC-Aurora connector server.
2. The TC-Aurora connector server collects all data, makes a REST call to TcIF, and attaches the data in a file.
3. TcIF processes the input data file, connects to Teamcenter using a SOA call, and updates the data.

If the data fails to export from Aurora to Teamcenter, there could be an issue at any level. The following are steps for troubleshooting the different components:

1. Troubleshoot export failure in TC-Aurora connector.
   - Verify that the Aurora metaserver is running.
   - Verify that the Aurora API client for the model (WBS Root ID) is running.
   - Verify that the TC-Aurora connector server for the model (WBS Root ID) is running.
   - Check the following for the TC-Aurora connector server:
     - The Teamcenter Site ID is correct in the config.cfg file.
     - The TCIF REST URL is correct in the config.cfg file.
     - The TcIF Admin username and password is correct in the encrypted file passed.
2. Troubleshoot export failure in TcIF.

- Verify that TcIF is up and running.
- Verify that the IPP&E bundles are installed in TcIF.
- Verify that the Teamcenter site is configured in TcIF with accurate tcifproxy user credentials.
- Verify that a queue was created with the WBS root ID in TcIF.
- Verify that the Teamcenter server is running.

3. Troubleshoot export failure in Teamcenter.
   - Verify that there are enough Teamcenter warm servers to process the SOA request from TcIF.
   - Verify that the tcifproxy user is properly set and that the user has all valid ACLs.

Fail to open in Aurora from the Active Workspace client

IPP&E has provided a command in Active Workspace to open the Aurora GUI client and connect to specific model data (WBS Structure data). If you are unable to open the Aurora client using this command, use the following steps to help with troubleshooting:

1. Verify that the IPP_aurora_client_url_for_model preference in Teamcenter has the correct URL.
2. Verify that pop-up browsers are enabled for the site.
3. Verify that the Aurora metaserver is running.
4. Verify that the TC-Aurora connector server for the model is running.
5. Verify that the Aurora API client for the model is running.
6. Verify that the Aurora web startup is properly set up in the web server.
7. Verify that the name and port that Aurora_WebStart used to deploy the web server matches the name and port in the IPP_aurora_client_url_for_model Teamcenter preference and the startAurora.jsp file.

Fail to trigger auto emails

1. Log into the Teamcenter Integration Framework WebConfig tool.
2. Click **Configure** - **Properties**.

3. Expand the **Camel** section and ensure the **emailservice.smtp-uris** box contains the appropriate email ID.

API client cannot load saved model

When the TC-Aurora connector server is started, it internally starts the Aurora API client and loads the model from the file (if it was previously saved). When you start the GUI client using the Open Schedule option, you are connected to the same model that is loaded in the API client.

If the API client fails to load the model, the GUI client appears blank. Use the following steps to help with troubleshooting:

1. The Aurora API client loads the model from the folder that is designated using the aurora-saved-model-dir variable in the config.cfg file. Verify that the folder path is correct.

2. Verify that there is a .cmp file for the model in the folder location.

3. If the .cmp file is missing for the model, you have the following options:

   - Aurora automatically makes a backup of the .cmp file located in the AuroraServer\backups\Unknown folder. The administrator copies the latest good file, places it in the saved model folder defined in the config.cfg file, and restarts the API client.

   - Run a full export from Teamcenter again. This recreates the model in Aurora.

Restart Aurora Metaserver

Before restarting the Aurora Metaserver, you should stop all running TC-Aurora connector servers.

1. Stop the Metaserver using File - Exit in the Aurora Metaserver window.

2. If the Metaserver fails to stop, you must forcefully terminate the Aurora Metaserver. Make sure that all the java processes that are running because of the Aurora Metaserver are stopped. It is recommended that you terminate the entire process tree using process explorer software.