TEAMCENTER

Linked Data Framework

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

1. What is Linked Data Framework?

Linked Data Framework is an integration framework that allows you to integrate different applications or enterprise information systems such as Product Lifecycle Management (PLM) systems and Application Lifecycle Management (ALM) systems using data linking techniques.

Linked Data Framework helps with the following business problems:

- How do you implement a process such as change management across different domains such as PLM and ALM?

  You can create links between parts of a process, such as between a change request in PLM and a defect in ALM. You can also extend Linked Data Framework to enhance its capabilities.

- How do you avoid creating new UIs for integrating different applications, and help users avoid learning new UIs?

  Using UI-based services (UI delegation), you can reuse existing UIs, validation, and business rules that are already enforced by the host application from the external application.

- How do you customize integrations without having to learn new techniques?

  Using standards such as REST, JSON, RDF, and web-based APIs.

- How do you enable users to access PLM data without having to learn PLM concepts or new tools?

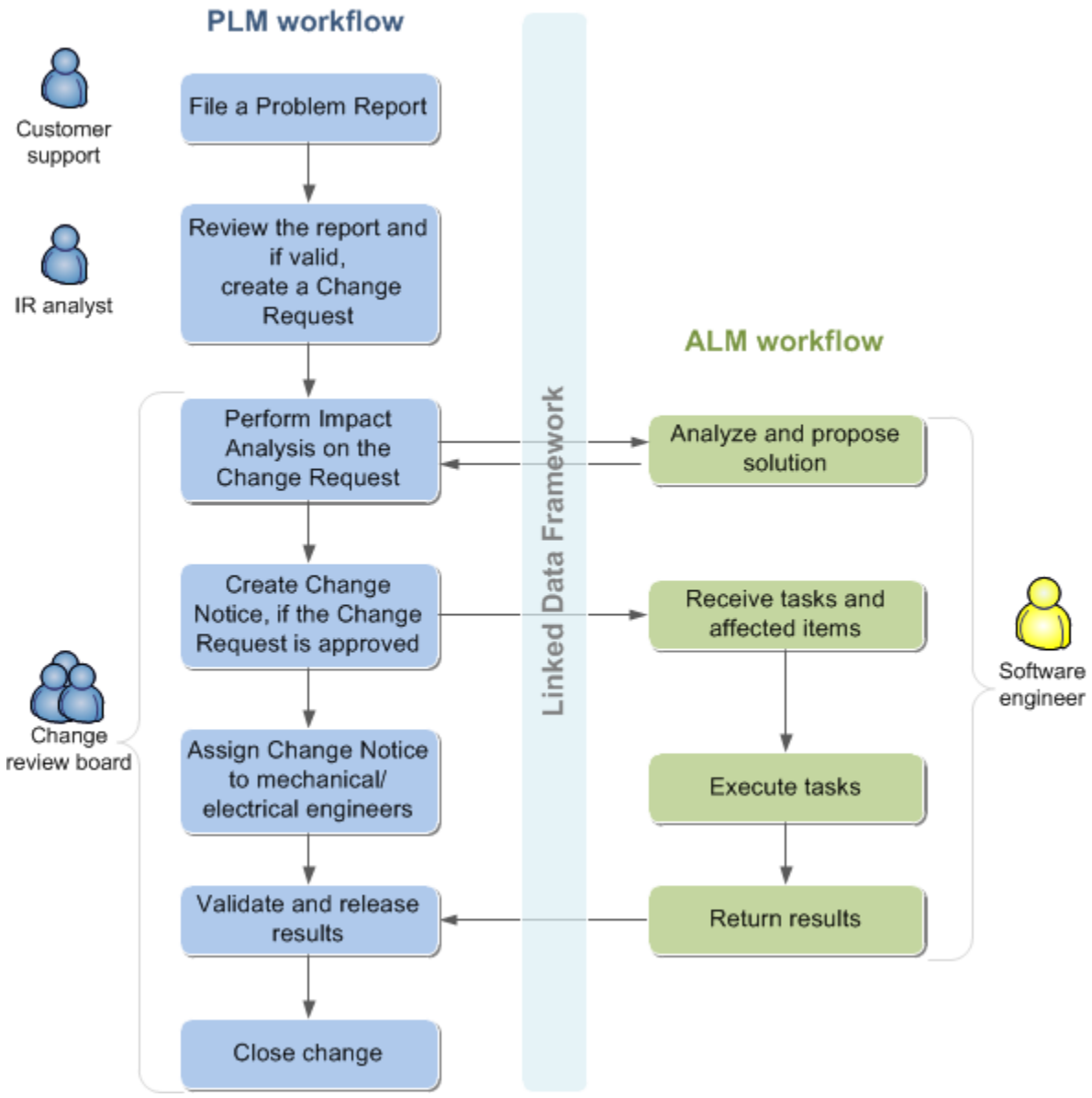
  You can use your tools you are familiar with and use UI delegation to view PLM data.

2. Sample change management process that uses Linked Data Framework

This process uses Linked Data Framework to integrate Teamcenter, which is a PLM tool, with Rational Team Concert (RTC), which is an ALM tool that is used to manage software products.

Without Linked Data Framework, you would have to coordinate the PLM and ALM workflows manually.

Assume an automobile company wants to upgrade the power window system of their car based on customer feedback. The following change process shows how the change can be handled in two different domains using Linked Data Framework.



3. Overview of integrating applications using Linked Data Framework

- **Install and deploy Linked Data Framework**
- Establish *trust* between Teamcenter and the external application

  To ensure that two applications can exchange information, they need to become trusted applications. This is achieved by exchanging OAuth consumer keys. In Teamcenter, you can establish trust while **creating a site representing the external application**.

  You must also create a representation of Teamcenter in the external application.
- Configure Teamcenter to act as a consumer

  A consumer application accesses and creates information from the external application.

  To set up Teamcenter as a consumer, you must **add service providers** exposed by remote systems.
- Configure Teamcenter to act as a provider

  A provider application provides access and services such as change management.

  To set up Teamcenter as a provider, add Teamcenter as a service provider in the external application.
- **Test your setup**

  By default, Change Management and core services are available. If you have set up Teamcenter as a consumer, you can create links and preview information in external applications. If you have set up Teamcenter as a provider, the external tools can create links in and view information from Teamcenter.
- Advanced customization

  In addition to Change Management and core services, you can **set up other domains** such as Requirements Management to use Linked Data Framework.

4. Installing Linked Data Framework

Linked Data Framework installation checklist

You can print the following checklists and use them to verify if you have completed all the tasks required for installing and deploying Linked Data Framework. Refer to the subsequent topics for more information.

Installation checklist

Check if you have selected the following features in **Teamcenter Environment Manager (TEM)** or **Deployment Center**.

Deployment checklist

Check if you have deployed the components required for Linked Data Framework to work:

	Deploy the **lis.war** file to a J2EE based web application server. The **lis.war** file is located at *TC_ROOT*/**lis/stage/lis/lis.war**. Note: **lis.war** is not supported on .NET based web application servers.

Verify if the Linked Data Framework installation is successful

	Check if the Linked Data Framework services are installed in the *TC_ROOT*/*lis* directory.
	Check if the core Linked Data Framework services are installed either as a Windows service or a Linux daemon.
	Check that **you can start the Linked Data Framework services**
	Check that you are able to access the Linked Data Framework root services.

Install Linked Data Framework using Teamcenter Environment Manager (TEM)

Teamcenter and Polarion integration works only with the following security modes:

- Both Teamcenter and Polarion use the http protocol.

- Both Teamcenter and Polarion use the https protocol.

Linked Data Framework requires a 4-tier Teamcenter installation.

1. Start Teamcenter Environment Manager (TEM).
2. Select the appropriate options until you reach the **Features** panel.
3. In the **Features** panel, under the **Base Install** navigation, select the following prerequisite features:
   a. **Teamcenter Foundation**
   b. (Optional) **Business Modeler IDE Standalone**, **Business Modeler IDE 2-tier**, or **Business Modeler IDE 4-tier**

      Choose this option only if you need to customize Linked Data Framework.
   c. **Teamcenter Rich Client 4-tier**
   d. In the **Features** panel, expand **Extensions→Enterprise Knowledge Foundation** and select **Change Management**.

      This installs the Change Management functionality. **LDF Foundation** requires this feature.
4. In the **Features** panel, expand **Extensions** and select **Embedded Software Management**.

   This installs the Embedded Software Management functionality. **LDF Embedded Software Management Integration** requires this feature.
5. In the **Features** panel, expand **Extensions→Platform Extensibility→Linked Data Framework Services** and select the following:
   a. **Java EE Based Linked Data Web Services**

      Installs the WAR file for Linked Data Framework and core services.
   b. **LDF Foundation**

      Installs Linked Data Framework.
   c. **LDF Change Management Integration**

      Installs Change Management Integration for Linked Data Framework. This option is automatically selected when you install **LDF Foundation**.
   d. **LDF Server Support**

Installs Linked Data Framework. This option is automatically selected when you install **LDF Foundation**.

e. **LDF Embedded Software Management Integration**

Installs Embedded Software Management Integration for Linked Data Framework.

f. **LDF Requirements Management Integration**

Installs Requirements Management Integration for Linked Data Framework.

6. In the **Features** panel, expand **Base Install→Active Workspace→Client** and select the following:

a. **Active Workspace Gateway**

Installs the Active Workspace Gateway client and is required by Linked Data Framework.

b. **Relationship Browser**

Installs the Relations functionality that shows the Linked Data Framework relations in a graphical format.

c. **Linked Data Framework**

Installs Linked Data Framework for Active Workspace.

7. In the **Features** panel, expand **Base Install→Active Workspace→Indexing Server** and select the following:

a. **Active Workspace Indexing Engine**

Installs the indexing engine that is used for searching Linked Data Framework artifacts.

b. **Active Workspace Indexer**

Configures the Indexer settings.

8. In the **Features** panel, expand **Base Install→Active Workspace→Server Extensions** and select the following:

a. **Active Workspace**

Installs Active Workspace binaries and templates. This feature is required by the **LDF Foundation** feature.

b. **Relationship Viewer**

Installs the **Relationship Viewer** template. This feature is required by the **LDF Foundation** feature.

c. Expand **Active Workspace Linked Data Framework Services** and select the following:

A. **LDF Foundation**

Installs Linked Data Framework for Active Workspace.

B. **Active Workspace LDF Change Management integration**

Installs Change Management Integration support for Active Workspace. This feature is automatically selected when you select the **LDF Foundation** feature.

C. **Active Workspace LDF Embedded Software Management Integration**

Installs Embedded Software Management Integration support for Active Workspace.

D. **Active Workspace LDF Requirements Management Integration**

Installs Requirements Management Integration support for Active Workspace.

E. **LDF Polarion Types Integration**

Installs Polarion business objects. This is required for Polarion Integration.

9. Click **Next**.

Select the appropriate options until you reach the **Configure Linked Data Framework** panel.

10. In the **Configure Linked Data Framework** panel, update the following:

a. **4-tier Server URL**

Specifies the Teamcenter 4-tier URL.

The URL is in the format **http://***hostname:TC-deploy-port-number***/tc** where *TC-deploy-port-number* is the port number where the **tc.war** file is deployed in the Web Application Server.

b. **LIS Core Service Port**

Specifies the port used to access the core services of Linked Data Framework.

> Note:
>
> This port is only to be used for administration purposes and not by customizers or integrators.

c. If you want Linked Data Framework to use SSO, select the **Is SSO Enabled** check box and specify the following:

A. **SSO Login Service URL**

Specifies the SSO login URL that has a fully qualified domain name, for example, **http://***hostname.domain name:port number***/***ssoLoginService*.

B. **SSO Application ID**

Specifies the SSO application ID.

> Note:
>
> If you have enabled SSO, you must also update the LDAP repository with the **oauthproxy** user credentials.
>
> If you have installed or updated to Teamcenter version 12.x and Active Workspace 4.3, use the **SSO Application ID** of the Active Workspace Gateway.

11. Click **Next**.

Select the appropriate options until you complete the installation.

Since Linked Data Framework requires a 4-tier Teamcenter installation, you must generate and deploy the Teamcenter web application.

If you plan to install security services, ensure that you add the fully qualified domain name in the login and service URLs while generating the security services web application.

When configuring identity services using the Application Registry table for Teamcenter and Active Workspace, ensure that each **Application Root URL** consists of a fully qualified domain name.

Install Linked Data Framework using Deployment Center

Teamcenter and Polarion integration works only with the following security modes:

- Both Teamcenter and Polarion use the HTTP protocol.
- Both Teamcenter and Polarion use the HTTP protocol.

Linked Data Framework requires a 4-tier Teamcenter installation.

Procedure

1. Log on to Deployment Center and in the **Applications** step, select the following components:

   - **Teamcenter>Extensions>LDF Polarion Types integration**
   - **Teamcenter>Extensions>Linked Data Services>Java EE Based Linked Data Web Services**

     Installs the WAR file for Linked Data Framework and core services.
   - **Teamcenter>Extensions>Linked Data Services>LDF Embedded Software Management Integration**

     Installs Embedded Software Management Integration for Linked Data Framework.
   - **Teamcenter>Active Workspace LDF Embedded Software Management Integration**

     Installs Embedded Software Management Integration support for Active Workspace.
   - **Teamcenter>Extensions>Linked Data Services>LDF Foundation**

     Installs Linked Data Framework.
   - **Teamcenter>Active Workspace LDF Server Support**

     Installs Linked Data Framework for Active Workspace.
   - **Teamcenter>Extensions>Linked Data Services>LDF Requirements Management Integration**

     Installs Requirements Management Integration for Linked Data Framework.
   - **Teamcenter>Active Workspace LDF Requirements Management Integration**

     Installs Requirements Management Integration support for Active Workspace.
2. In the **Components** task, select **Linked Data Web Services (Java EE)**.
3. Update the **Linked Data Web Services (Java EE)** as follows:

Entry	Description
Machine Name	• In the **Machine Name** box, specify the machine where Linked Data Framework will be installed.

Entry	Description
	• In the **OS** list, select the operating system.
General Settings	• In the **Installation Path** box, specify the path where Linked Data Framework will be installed. • In the **OAuthorproxy User Password** and the **Confirm OAuthorproxy User Password** boxes specify the **oauthrproxy** password.
Login Account	In the **Login Account** section specify the username and password for the account that will be using the Linked Data Framework web services.
Server Configuration Settings	• In the **Teamcenter 4-tier URL** box, specify the Teamcenter server URL. • Enable the **Override connection** check box if you wish to override the default value of the Teamcenter URL.

4. Once you have completed all your tasks, you can deploy the environment.

Postrequisites

Since Linked Data Framework requires a 4-tier Teamcenter installation, you must generate and deploy the Teamcenter web application.

If you plan to install security services, ensure that you add the fully qualified domain name in the login and service URLs while generating the security services web application.

When configuring identity services using the Application Registry table for Teamcenter and Active Workspace, ensure that each **Application Root URL** consists of a fully qualified domain name.

Deploy Linked Data Framework

After installing Linked Data Framework using TEM, you must deploy the *lis.war* file to a J2EE-based Web Application Server.

It is recommended that you deploy *lis.war* and *tc.war* to the same J2EE-based Web Application Server. If you are using JAVA based Web Application Server such as Apache Tomcat or JBoss, you must always deploy the *lis.war* and *tc.war* on the same Web Application Server.

The *lis.war* is located at *TC_ROOT***/lis/stage/lis/lis.war**.

To check if the web application server you plan to use is certified with your current Teamcenter platform, refer to the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com.

Note:

OSLC integration only works when both applications are configured with https. For example, Teamcenter-Polarion integration works only when both the applications are configured with https. Perform the following configurations while deploying on J2EE-based Web Application Server.

Deploying on JBOSS

For supported version of JBOSS, see the Hardware and Software Certifications knowledge base article on Support Center:

https://support.sw.siemens.com

1. Open the *web.xml* file from the *lis.war/WEB-INF* directory.

2. Add the section **<session-config>** at the end of the *web.xml* file and inside the **<web-app>** tag as follows:

```
<web-app>
    <session-config>
        <session-timeout>1440</session-timeout>
        <cookie-config>
        <path>/</path>
            <secure>true</secure>
            <comment>"\"; SameSite=None; "</comment>
        </cookie-config>
    </session-config>
</web-app>
```

3. Deploy the *lis.war* file in JBOSS.

4. Restart the following:

   - JBOSS
   - Teamcenter Server pool manager
   - LIS service

Deploying on JBOSS 7.4 or WildFly web application server

After you have deployed *lis.war* file on JBOSS version 7.4 or WildFly web application server, do the following:

1. Create an XML file named *jboss-deployment-structure.xml* in the in *lis.war/WEB-INF* directory.

2. Paste the following content to the XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-deployment-structure>
  <deployment>
     <exclusions>
        <module name="org.apache.logging.log4j.api"/>
    </exclusions>
  </deployment>
</jboss-deployment-structure>
```

3. Deploy the *lis.war* file in JBOSS or WildFly web application server.

4. Restart the following:

   - JBOSS or WildFly web application server
   - Teamcenter Server pool manager
   - LIS service

Deploying on IBM WebShpere 8.5.5 and 9

1. Copy the *lis.war* file to the same directory where the WebSphere application is installed, for example, *E:\LIS_WAR*.

2. Using a command prompt, navigate to the directory where you copied the *lis.war* file, for example, *E:\LIS_WAR*.

3. Execute the following command:

   ```
   jar -xvf lis.war
   ```

   This extracts the *lis.war* file in the same directory that you navigated to from the command prompt.

4. Create another directory, for example, *E:\LIS_WAR\shared_lib*.

5. Copy the following JAR files from the new location where you extracted the contents of the *lis.war* file. For example, copy the jar files from *E:\LIS_WAR \lis\WEB-INF\lib* directory to the *E:\LIS_WAR\shared_lib* directory. In the following list, version numbers are represented by **nnn**.

   - *hk2-api-nnn-b32.jar*
   - *hk2-locator-nnn-b32.jar*

- *hk2-utils-nnn-b32.jar*
- *jersey-common.jar*
- *jersey-container-servlet-core.jar*
- *jersey-guava-nnn.jar*
- *jersey-media-jaxb.jar*
- *jersey-server.jar*
- *oauth-httpclient4-nnn.jar*
- *org.apache.httpcomponents.httpclient_nnn.jar*
- *org.apache.httpcomponents.httpcore_nnn.jar*
- *org.osgi.core-nnn.jar*
- *osgi-resource-locator-nnn.jar*

6. Delete the copied jar files from the *lib* directory. For example, delete the jar files from the *E:\LIS_WAR\lis\WEB-INF\lib* directory.

7. Using a command prompt, navigate to the directory that contains the *lis.war* file, for example, the *E:\LIS_WAR\lis* directory, and execute the following command:

   ```
   jar -cvf lis.war
   ```

   This creates a new *lis.war* file that does not have JAR files that we deleted in the previous step. Deploy the *lis.war* file into the WebSphere application as follows:

8. In the WebSphere navigation tree, expand **Environment** and click **Shared libraries**.

9. In the **Shared Libraries** pane, click **New**.

10. In the **Configuration** pane, update the information as follows:

    - **Name**: Specify the name of the library, for example, *shared_lis_lib*.
    - **Description**: Provide a description for the library, for example, Shared isolated library for Apache Http Client and Apache Wink.
    - **Classpath**: Specify the location that contains new LIS files. For example, change the classpath from the *E:\LIS_WAR\lis\WEB-INF\lib* directory to *E:\LIS_WAR \shared_lib*.

- Select the option **Use an isolated class loader for this shared library**.

Click **OK**.

The lis application is deployed on WebSphere.

11. Stop the lis application, if running.
12. On the Enterprise Applications page, click **Class loading and update Detection**.
13. Select the **Classes loaded with local class loader first (parent last)** check box and click **OK**.
14. On the Enterprise Applications page, click **Shared library references**.
15. Under **Application**, select **lis_war** and click **Reference shared libraries**.
16. In the Shared Library Mapping page, move the library from the **Available** list to the **Selected** list.
17. Click **OK**.
18. Disable the built in **JAX-RS** by updating the following JVM property to **true**:

    ```
    com.ibm.websphere.jaxrs.server.DisableIBMJAXRSEngine=true
    ```

    You can update this property in the admin console of WebSphere by going to **Servers→All Server→Server Infrastructure→Java and Process Management→Process Definition→Additional Properties→Java Virtual Machine→Additional Properties→Custom Properties**.
19. Save the configuration.
20. Restart WebSphere.
21. Check if your configuration is saved and then start the lis application.

Deploying on Apache Tomcat

For supported version of Tomcat, see the Hardware and Software Certifications knowledge base article on Support Center:

https://support.sw.siemens.com

After you deploy *lis.war* to Apache Tomcat, make the following configuration change:

1. Update the *context.xml* file located in *TOMCAT-INSTALL-DIR\conf* and add the following entries in the **context** section:

```
<Context sessionCookiePath="/" >

<CookieProcessor sameSiteCookies="None" />
```

```
<Context sessionCookiePath="/" >

    <CookieProcessor sameSiteCookies="None" />
    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
```

2. Restart the following:

   - Tomcat Apache server
   - Teamcenter Server pool manager
   - LIS service

Configurations when tc.war is deployed on IIS.NET

lis.war does not support the deployment on IIS.NET. If you have deployed *tc.war* on IIS.NET, update the default cross site cookie setting as follows:

1. Open the *web.config* file located in *SystemRoot\Microsoft.NET\Framework\latestversion\CONFIG*.

2. Update the value of the **sameSite** to **None**.

```
<httpCookies sameSite="None"/>
```

```
<location allowOverride="true">
    <system.web>
        <httpCookies sameSite="None"/>
        <sessionState cookieSameSite="None" />
        <securityPolicy>
            <trustLevel name="Full" policyFile=
            "internal" />
```

3. Restart the IIS server.

Configure Linked Data Framework web applications

Perform the following tasks after **deploying Linked Data Framework on a Web Application Server**.

1. Open the *config.json* file located in the Active Workspace Gateway installation directory, at the following location:

   TC_install_main_directory\Clients\aw\ms_node_TC\microservices\gateway-1.1.0

2. Update the **target** element in the **lis** section of the *config.json* file.

```
{
    "port": 3000,
    "index": "tc.html",
    "keyPath": "",
    "certPath": "",
    "maxAge": 15552000,
    "routes": {
        "tc": {
            "path": "/tc",
            "target": "http://vc6s004:7001/tc"
        },
        "vis": {
            "path": "/VisProxyServlet",
            "target": "http://VpaHost:8089/VisProxyServlet"
        },
      "lis": {
            "path": "/lis",
            "target":"http://<hostname>:<lis_war_port>/lis",
            "disableAuth": true
      }
    },
…
}
```

   Where:

   - **target** is the path where *lis.war* is deployed.
   - **lis_war_port** is the value of the port where the *lis.war* file is deployed in the Web Application Server.

3. If you have Security Services configured with a Teamcenter and Polarion integration, perform the folowing steps:

a. Create a Security Services **Application ID** for the Teamcenter server URL. The URL is the location where the **tc.war** file is deployed on the Web Application Server.

b. Open the *TC_DATA\tc_profilevars.bat* file and add the value of **Application ID** to the **TC_SSO_APP_ID**.

Example:

```
TC_SSO_APP_ID=AW_GW, Application-ID
```

c. Start the **Microservice Manager PoolA** Windows service.

4. Restart the **Microservice Manager PoolA** Windows service.

5. The rootservices URL changes to *http://hostname:gateway_port/lis/oslc/rootservices*.

   Check if you are able to access this URL.

6. Update the following Linked Data Framework preferences with the following values:

   a. **TC_LDF_host_url**: *http://hostname:gateway_port/lis/*

   b. **ActiveWorkspaceHosting.LIS.URL**: *http://hostname:gateway_port/tc.html?ah=true*

7. Start the following services:

   - Teamcenter Server pool manager and wait till it has spawned all Teamcenter server processes.
   - Application server (where **tc.war** and **lis.war** is deployed) and wait till it is in running mode.
   - Start the lis service.

When these changes take effect all requests to Linked Data Framework are routed through the Active Workspace Gateway, which is configured on port 3000 by default.

Configurations when using Teamcenter Security Services

Using Teamcenter Security Services

1. In the *config.json* file, located in the *TC_INSTALL_DIR\Clients\aw\microservices\gateway-x.x.x* directory, add the following two entries in the **csurfExclusions** section as follows:

```
"/lis/sso_login_callback",

"/lis/authenticated"
```

A sample section of the *config.json* file is as follows:

```
"csurfExclusions": [
            "/AWSSOLogin",
            "/graphql",
           "/lis/sso_login_callback",
           "/lis/authenticated"
        ],
```

Not using Teamcenter Security Services

1. In the *config.json* file, located in the *TC_INSTALL_DIR\Clients\aw\microservices\gateway-x.x.x* directory, add the following two entries in the **csurfExclusions** section as follows:

   ```
   "/lis/lisTCLogin"

   "/lis/authenticated"
   ```

   A sample section of the *config.json* file is as follows:

   ```
   "csurfExclusions": [
               "/AWSSOLogin",
               "/graphql",
               "/lis/lisTCLogin",
               "/lis/authenticated"
           ],
   ```

Starting Linked Data Framework services

To use Linked Data Framework, start services in the following order:

- Start the Teamcenter Server pool manager and wait till it has spawned all Teamcenter server processes.
- Start the application server (where *tc.ear* is deployed) and wait till it is in running mode.
- Start the lis service.

  When started, the service attempts to log on to the Teamcenter server.

Verify if the Linked Data Framework installation is successful

Check the following:

- The J2EE based Linked Data Framework services are installed at *TC_ROOT***/lis**.

- The core services of Linked Data Framework are installed either as a Windows service or a Linux daemon. On Windows, the service starts automatically, while on Linux, the system administrator must configure the daemon to start automatically.

- You are able to access the Linked Data Framework root services. This *rootservices* document shows root services with service catalogs, services, and OAuth authorization URLs.

  The root service url is in the format **http://***Web app host:Port***/lis/oslc/rootservices**, for example, http://lm6s003:7001/lis/oslc/rootservices.

5. Configuring integrations using Linked Data Framework

Checklist for configuring Linked Data Framework

Using Linked Data Framework, your application can run in two modes:

- Consumer: A consumer accesses services from the other(external) application. For example, when Teamcenter is set up as a consumer, it accesses services from the external application.
- Provider: A provider allows the other application to access its services. For example, when Teamcenter is set up as a provider, external applications can access its services such as change management.

You can print the following checklist and use it to verify if you have completed all the tasks required for configuring Linked Data Framework:

Teamcenter configurations

	Create a Linked Data Framework site in Teamcenter that represents the external application
	Add service providers
	Update service providers with semantic relations they can use
	Set mandatory preferences (if not already set) | | | |---|---| | | **ActivewWorkspaceHosting.LIS.URL** If the preference does not exist, create it. | | | **TC_LDF_host_url** | | | **LDF_FQDN** |
	Configure Relations Browser to show Linked Data Framework objects and relations

Polarion configurations

For more information about configuring Polarion and setting up Polarion in HTTP mode, see the Polarion Connector for Teamcenter Polarion Installation Guide and the Polarion Administrator and User Help. Also ensure that you refer to the latest configurations for Chrome browser in the Polarion documentation.

	Add Teamcenter server as a provider for Polarion OSLC
	Enable linking of Polarion work items to Teamcenter items
	Install and configure integration workflow extension for Teamcenter
	Install and configure integration workflow extension for Polarion

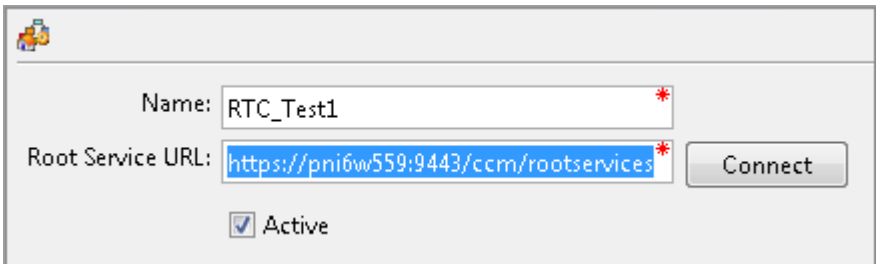
These basic configurations will allow you to use Linked Data Framework integration. To extend the Linked Data Framework functionality, you must perform the following advanced configurations and customizations:

- Update the semantic mapping file.
- Customize Relations Browser functionality to display Linked Data Framework objects and relations.
- Customize Active Workspace to display remote links.
- Expose additional services using Linked Data Framework.

Create a Linked Data Framework site in Teamcenter representing the external application

A site represents the tool that integrates with Teamcenter.

1. In Organization, select **Linked Data Framework Providers** from the **Organization List** tree.



2. Type a name for the site in the **Name** box.

> Tip:
>
> Provide a specific name so that it is easy to identify the external system for which you have created a site.

3. Type the root serives URL.

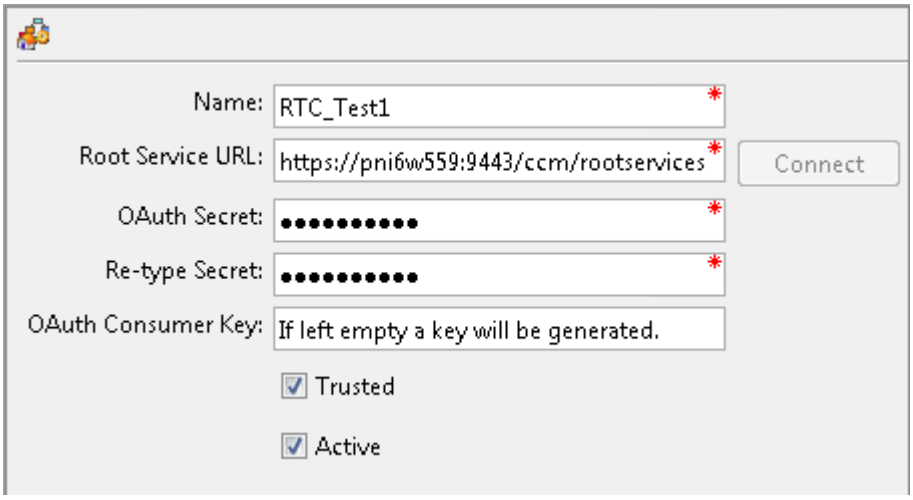
   The root services URL is the entry point to the site and helps find the available service providers.

> Note:
>
> Do not specify an IP address.

4. Click the **Connect** button.

   If this is a new session, a login window of the external site appears. You must log on to the external site with appropriate credentials.

   If the connection is successful, and if the site supports OAuth, the OAuth boxes appear.



5. Type the secret phrase to be associated with the OAuth Consumer key in the **OAuth Secret** box.

   The secret phrase can be any text.

6. Type the secret phrase, the one you provided above, again in the **Re-type Secret** box.

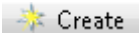
7. (Optional) Type the OAuth consumer key in the **OAuth Consumer Key box**. If this box is left blank, Teamcenter automatically generates a consumer key.

8. Select the **Trusted** check box to indicate that the site you are creating can trust Teamcenter. Trusted sites can share authorization data with other trusted sites.

9. Select the **Active** check box if it is not already checked to mark the site as active.

   Only an active site can integrate with other sites.

10. Provide a path to the catalog that contains services this site provides by adding the catalog URL to the **Add Catalog URL** list.

11. Click **Create** 

to create a Linked Data Framework site.

If Teamcenter is configured with SSO and you do not have an existing SSO session, a dialog box opens asking for login and password. Type your login and password after which the Site is created.

> Note:
>
> If you are using Microsoft Windows, for each client machine that leverages Security Services authentication, add the host URL of the Linked Data Framework site to the list of trusted sites in your web browser. You must also ensure that your web browser is configured to display popup messages.

Prerequisites before adding Polarion service providers for certain versions of Teamcenter

If your Teamcenter version is 13.1.x or higher and in the Teamcenter-Polarion integration, Polarion and Teamcenter and configured with https or configured behind https reverse proxy, import valid Polarion certificates in the JVM before adding Polarion service providers to Teamcenter:

Import valid Polarion SSL certificates in the JVM

1. Export Polarion SSL certificate from the browser or get the SSL certificate from the Polarion configuration.

   For more information about Polarion SSL certificate search the Polarion help or contact your Polarion administrator.

2. From the Teamcenter db console, check if the **TC_JRE_HOME** environment variable is set.

3. Import the Polarion SSL certificate in the JVM using the keytool utility.

   ```
   keytool -import -alias "alias-name" -keystore "%TC_JRE_HOME%
   \lib\security\cacerts" -file downloaded-polarion-certificate-file-path
   ```

   > Example:
   >
   > ```
   > keytool -import -alias "polarion-server" -keystore "%TC_JRE_HOME%
   > \lib\security\cacerts" -file C:\apps\polcert.cer
   > ```

For more information, search for the topic *Import a certificate to the Java Keystore* in the Polarion documentation.

After completing these steps, you can create a Polarion service provider in Teamcenter.

Add service providers

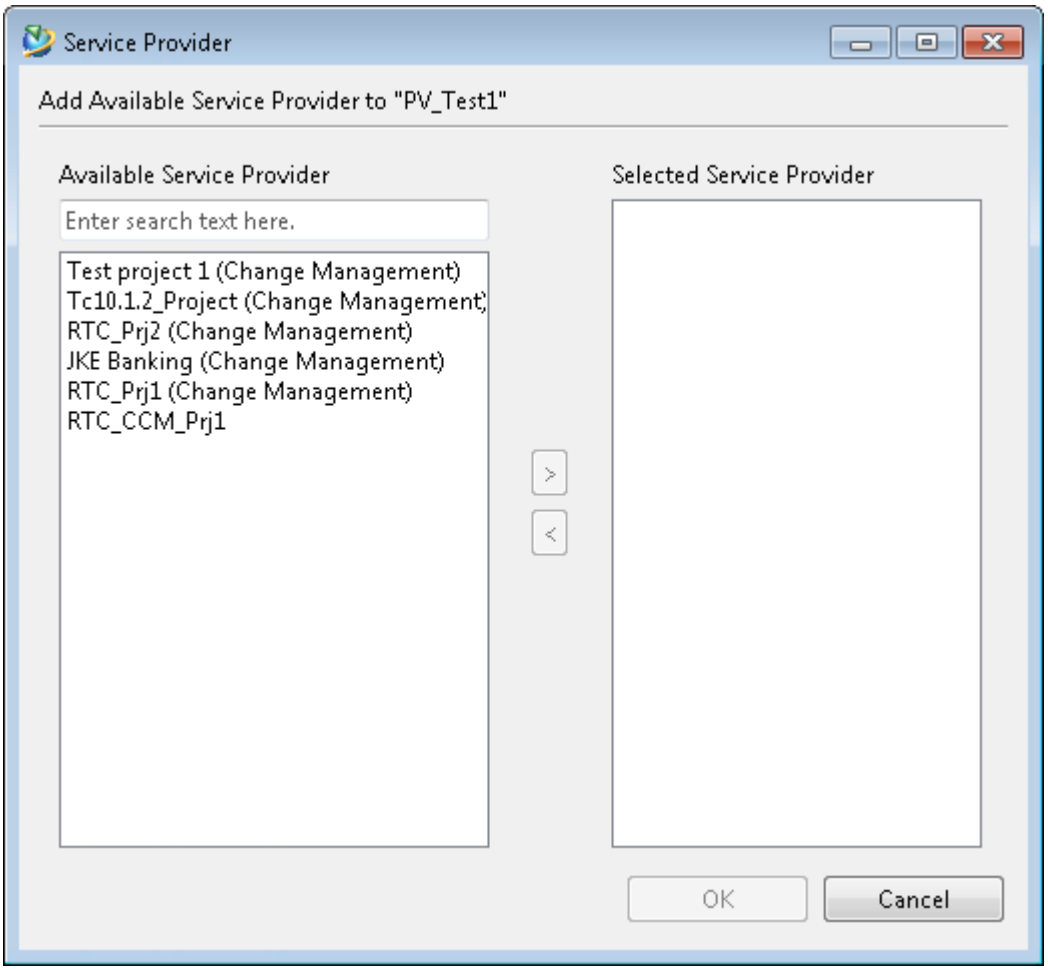
A service provider represents the services published by the external application.

1. In Organization, go to **Organization List→Linked Data Framework Providers** and select a site for which you want to add a service provider.

2. Click the **Add Service Provider** button.

   If this is a new session, a login window of the external site appears. You must log on to the external site with appropriate credentials.

3. In the **Service Provider** dialog box, select the service provider you want to add from the **Available Service Provider** list and add the service provider to the **Selected Service Provider** list.



> Note:
>
> Teamcenter allows only 128 characters in the service provider name. In case your service provider name is greater than 128 characters, modify the service provider name in the external application and limit the name to 128 characters.

4. Click **OK**.

If Teamcenter is configured with SSO and you do not have an existing SSO session, a dialog box opens asking for login and password. Type your login and password after which the service provider is created.

The service provider appears under the site name in the **Linked Data Framework Providers** node of the **Organization List** tree.

Update service providers with semantic relations they can use

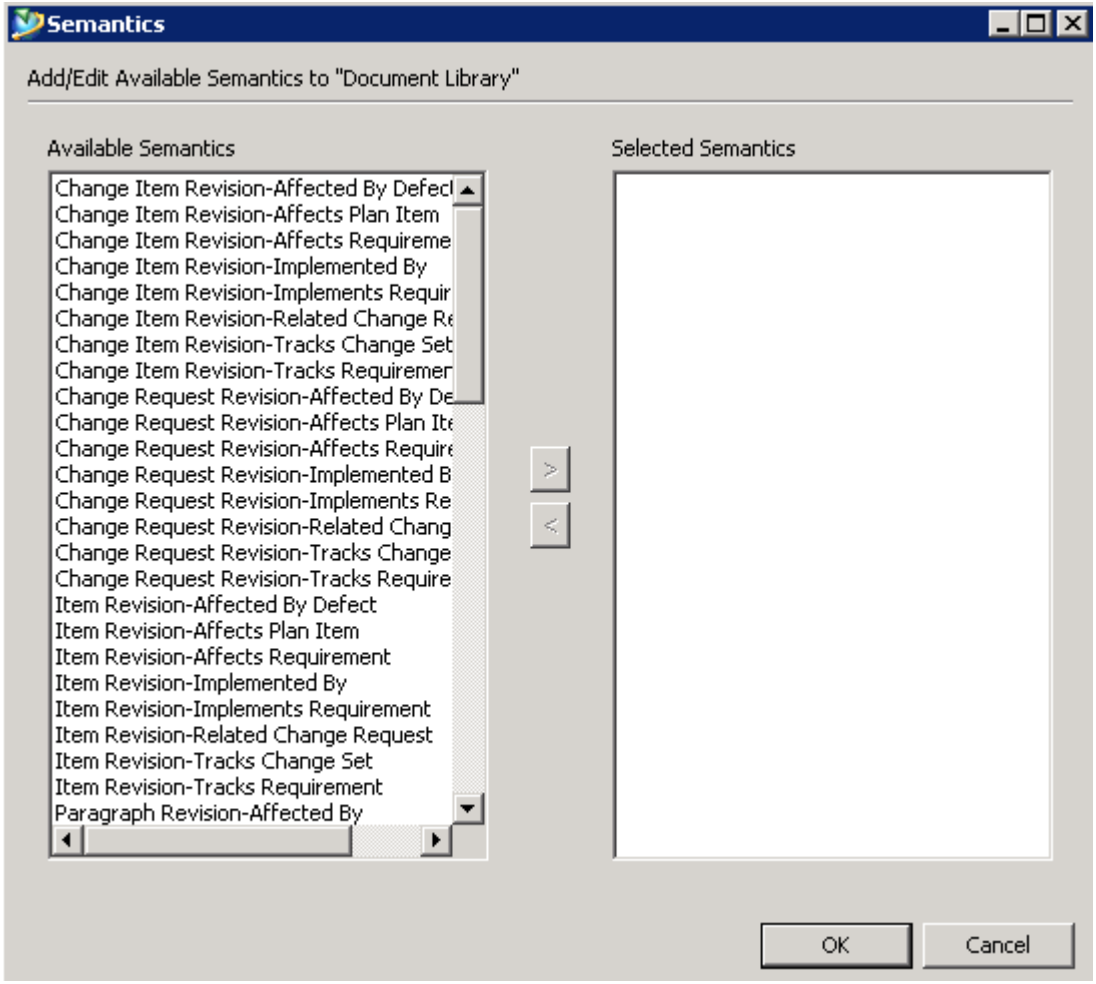
Semantics or Semantic relationships specify what relations to use when you create remote links. For example, when you create remote links between a Change Item Revision and an object in the external application, you can choose to apply relations defined here such as Affected By Defect, Affects Plan Item, and Affects Requirement.

To ensure that these semantic relations are available when you create Linked Data Framework links, you must add the semantic relations that you want to use for each service provider as follows:

1. In Organization, from the **Organization List** tree, select a site from the **Linked Data Framework Providers** node and then select a service provider.

2. In the **Semantics list**, click the **Add/Remove** button.

3. In the **Semantics** dialog box, select the semantic relations you want from the **Available Semantics** list and add them to the **Selected Semantics** list.



4. Click **OK**.

5. Click **Modify**.

You can also customize the semantic relationships.

Update the semantic mapping property files

After upgrading your Teamcenter and Linked Data Framework installation, you must update the semantic mapping property files, if you modified those files.

These files are:

- *oslc_cm_properties.xml* located in *TC_INSTALL_DIR/osl0lisoslc*.
- *oslc_core_properties.xml* located in *TC_INSTALL_DIR/osl0lisoslc*.

To update the files:

1. Take a backup of the files.

2. After you upgrade your Teamcenter and Linked Data Framework installation, run the **create_botype_reader** utility with the following arguments:

   - **-f=create**

   - **-filename=***name-of-the-property-file-you-backed-up*

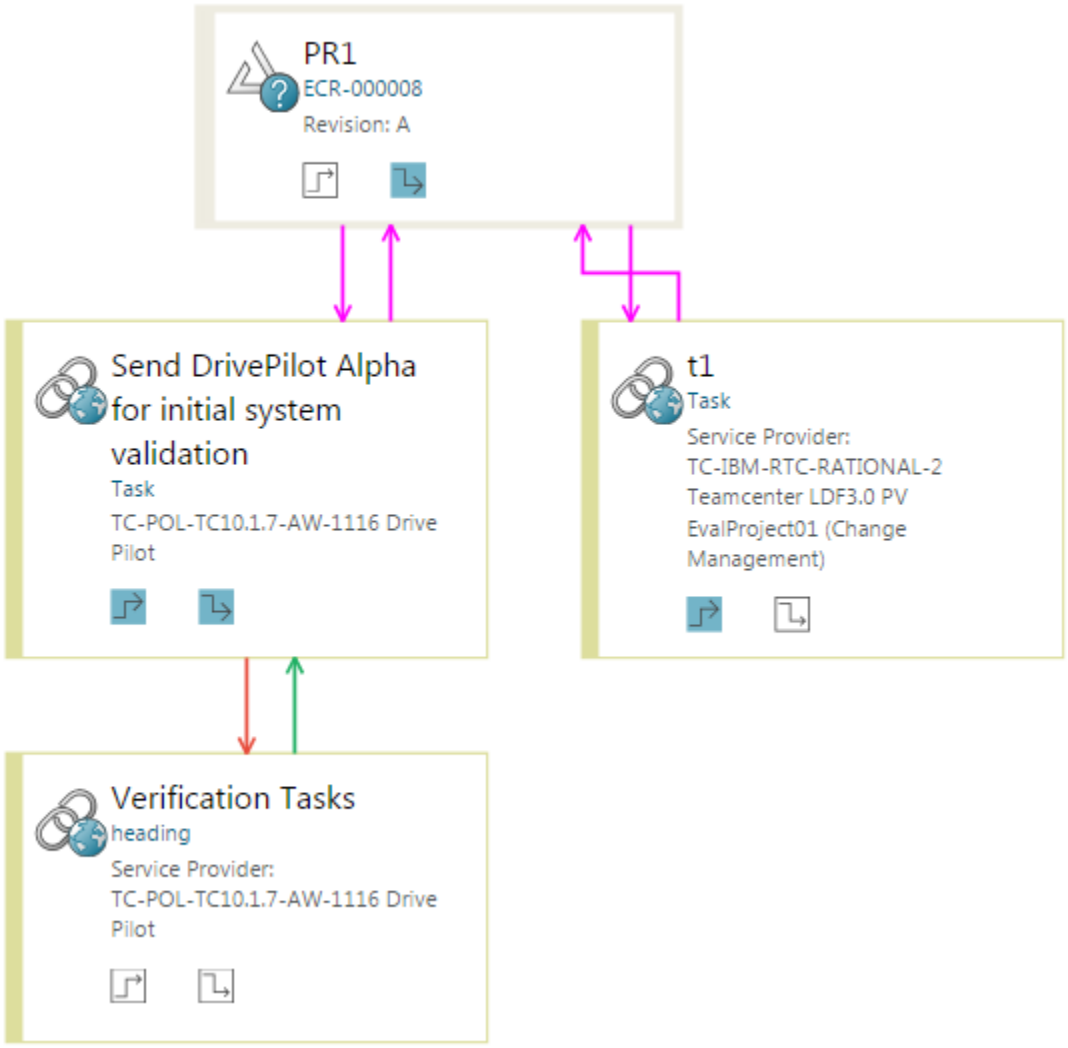
     For example, **-file= oslc_cm_properties.xml**

   Example:

   ```
   create_botype_reader -u=ed_admin -p=ed_admin -g=dba
   -mode=map_schema -domain=core -f=create -file=d:\oslc_cm_properties
   ```

Configure Relations tab to show Linked Data Framework objects and relations

The Relations tab in Active Workspace allows you to view the relationship between objects in Teamcenter. You can also configure it to show relations between Teamcenter objects and objects in an external or remote application such as Polarion.



To enable the **Relations** tab to show relations between Teamcenter objects and objects in a remote application, do the following:

1. In the Teamcenter rich client, search for the dataset **RelationB***.

2. In the **Search** pane, right-click **RelationBrowserConf** and choose **Named References**.

3. In the **Named References** dialog box, select *RB_UIConfigure.xml* and click **Download**.

4. Open the *RB_UIConfigure.xml* file and update the following four configurations:

   Add relations in the General view

   - Go to the section **view name="General"** and under the **group name="relations"** section, add the relations you want.

Use the following format:

```
<filter name="AffectedByDefectCM" parameterSet="AffectedByDefect"
color="(138,66,8)"/>
```

- **filter name** specifies the name of the relation.
- **parameterSet** specifies the name of the **parameterSet** element that is used by the relation. The **parameterSet** element defines how the relations and objects are mapped.
- **color** specifies the display color of the relation.

Add objects in the General view

- Go to the section **view name="General"** and under the **group name="objects"** section, add the objects you want.

  Use the following format:

```
<filter name="POL-ChangeRequest" parameterSet="
Ldf0TempLinkRuntimeObject"
color="(102,0,204)"/>
```

- **filter name** specifies the name of the object.
- **parameterSet** specifies the name of the **parameterSet** element that is used by the object. The **parameterSet** element defines how the relations and objects are mapped.
- **color** specifies the display color of the object.

Define parameterSet for the relations you added in the General view

- Go to the section **parameterSets** and under it, create a new **parameterSet=***relation-name* section.

  Example:

```
<parameterSet name="AffectedByDefect">
```

  Under the **parameterSet** section, add entries in the following format:

```
<parameterSet name="AffectedByDefect">
  <clipsfacts>

<fact>source=LDF,key=AffectedByDefectCM,relationType=AffectedByDefe
```

```
ct,
    targetDirection=forward,inputTypes=ItemRevision</fact>

<fact>source=LDF,key=AffectedByDefectCM,relationType=AffectedByDefe
ct,
    targetDirection=forward,inputTypes= Ldf0TempLinkRuntimeObject
</fact>
  </clipsfacts>
</parameterSet>
```

Define the **parameterSet** values in the **facts** elements of the **clipfacts** section.

- **source** specifies the query service. For relations to external systems, only the query **LDF** is supported.
- **key** specifies the remote relation type. This should match value of the **filter name** that you defined in the **group name="relations"** section.
- **relationType** specifies the name of the relation type used for Linked Data Framework linking.

  The Linked Data Framework relations that you can use are defined in the **RelationTypeMapping element of the semantic mapping file**. Use the relations defined in the **consumerRelation** and **providerRelation** elements. When using the relations, remove the prefix. For example, if the relation defined in the **consumerRelation** element is **oslc_cm:affectsRequirement**, remove the prefix **oslc_cm:** and use only **affectsRequirement**.

  You can also add relations that are not defined in the semantic mapping file. Refer to your external application for the names of the external relations.
- **targetDirection** specifies the direction of the relation. For remote systems, the direction value is **forward**.
- **inputTypes** specifies which objects to query.

Define parameterSet for the objects you added in the General view

- Go to the section **parameterSets** and under it, create a new **parameterSet=***object-name* section.

  For example

  ```
  <parameterSet name="Ldf0TempLinkRuntimeObject">
  ```

  Under the **parameterSet** section, add entries in the following format:

  ```
  <parameterSet name="AffectedByDefect">
  ```

```
    <clipsfacts>
      <fact>Ldf0TempLinkRuntimeObject </fact>
    </clipsfacts>
  </parameterSet>
```

Add the object names in the **facts** elements of the **clipfacts** section.

The fact value should be the values defined in the **RBTypeAttrMapping element of the semantic mapping file**.

5. Check out the *RelationBrowserConf* dataset.
6. Delete the original *RB_UIConfigure.xml* file from the **Named References** dialog box and import the new *RB_UIConfigure.xml* file.
7. Click **Close**.
8. Check in the *RelationBrowserConf* dataset.

Configure the integration with Polarion

For more information about configuring the integration with Polarion, see the Polarion documentation.

Set preferences

- Create a preference named **ActiveWorkspaceHosting.LIS.URL** and update it with the value of the Active WorkspaceClient URL.

  > Note:
  >
  > Ensure that the URL has the suffix **?ah=true**. For example **http://lm6s003:7001/awc/tc.html?ah=true**. This suffix is required for Active Workspace delegated UIs to work properly.

- To ensure that backlinks are created in the integrating application, create a preference named **TC_LDF_host_url** and for its value, specify the URL that points to the local Linked Data Framework web application.

  Example: http://lm6s003.net.plm.eds.com:7001/lis/

- To ensure that Linked Data Framework URLs contain fully qualified domain names, create a preference named **LDF_FQDN** and specify the fully qualified domain name of the machine where Linked Data Framework is deployed as the preference value.

  Example: **.net.plm.eds.com**

Note:

Ensure that the URL uses hostname and not IP-based URLs. Linked Data Framework does not support IP-based URLs.

6. Administering Linked Data Framework

Starting and stopping Linked Data Framework

Linked Data Framework is available as a Windows service and a Linux daemon. You must turn on this service for Linked Data Framework to work.

This service is named as **Teamcenter LIS Service LIS_***Host Name_User Name*.

On Windows, the service stars automatically, while on Linux, the system administrator must configure the daemon to start automatically.

The daemon script is located at *TC_ROOT***lis\rc.liswebservices.sh**

This script contains the following arguments:

-start

Starts the Linked Data Framework web services daemon.

-start_msg

Prints a message that the daemon is starting.

-stop

Immediately stops Linked Data Framework web services.

-stop_msg

Prints a message that the daemon is stopping.

-status

Prints a message indicating whether the Linked Data Framework web services daemon is running.

-log

Prints the entire output of the Linked Data Framework web services.

Creating or deleting the OAuth consumer keys

Use the **maintain_consumer_key** utility to create or delete oauth consumer keys.

You only need to use this utility when the remote application does not support automatic exchange of consumer keys with Teamcenter.

Example:

- Adding an OAuth consumer key:

```
maintain_consumer_key  -u=username -p=password -g=dba -mode=add
-consumer_name=RTC -consumer_secret=111 -isTrusted=Y
```

- Deleting an OAuth consumer key:

```
maintain_consumer_key  -u=username -p=password -g=dba -mode=delete
-consumer_name=RTC -consumer_secret=111
-consumer_key=12789259677583223
-isTrusted=Y
```

After you create a consumer key, run the system refresh operation from your browser by accessing the following url:

http://*machine_name*:*port_number*/lis/sys/refresh.

Here, *machine_name* is the machine where the *lis.war* file is installed and *port_number* is the web server port, for example, http://localhost:7001/lis/sys/refresh.

You must always run the system refresh operation after creating a consumer key and then use the consumer key to create a site.

Updating the oauthproxy password

Teamcenter creates the **oauthproxy** user when you install Linked Data Framework.

If you change the **oauthproxy** user password in Teamcenter, you must generate the password file again using the **encryptPass.bat** or **encryptPass.sh** file as follows:

```
encryptPass.bat new password
```

This file is located in the *TC_ROOT*/**lis** directory.

If Teamcenter SSO is enabled, you must update the oauthproxy password in the LDAP repository as well.

Remove unencrypted password from the log file

To ensure that no unencrypted passwords are retained in the log file for the **liswebservices** process, update the *log4j.properties* file located in the *TC_ROOT*/*lis* directory with the following entry:

```
log4j.logger.org.apache.http.wire = OFF
```

You log4j file entry looks like this:

```
You log4j file entry looks like this:
```

```
# ================================================================= #
# Set the Logging for SOA Communication between LIS Web Services and  #
# Teamcenter Server.                                                  #
# The Level can be DEBUG, INFO, WARN, ERROR, FATAL, or OFF            #
# ================================================================= #
log4j.logger.com.teamcenter.soa = OFF, SOARequestFileAppender
log4j.additivity.com.teamcenter.soa.client.Connection = false
log4j.additivity.com.teamcenter.soa = false

log4j.logger.org.apache.commons.httpclient = OFF
log4j.logger.httpclient.wire = OFF
log4j.logger.org.apache.http.wire = OFF
```

Set the Linked Data Framework service session time

The Linked Data Framework service times out after 1440 minutes. After this, you must log on again.

You can change the duration for which the service remains active by updating the **lisServiceSession.timeout** property in the *liswebservices.properties* file.

This file is located in the *TC_ROOT\lis* directory.

Chrome browser settings if you are not using Teamcenter Security Services

If you are using the *http* protocol to access Teamcenter, and if the version of Chrome browser is above version 80, disable the following flags by typing `chrome://flags` in the browser:

- **SameSite by default cookies**
- **Enable removing SameSite=None cookies**
- **Cookies without SameSite must be secure**

7. Exposing additional services using Linked Data Framework

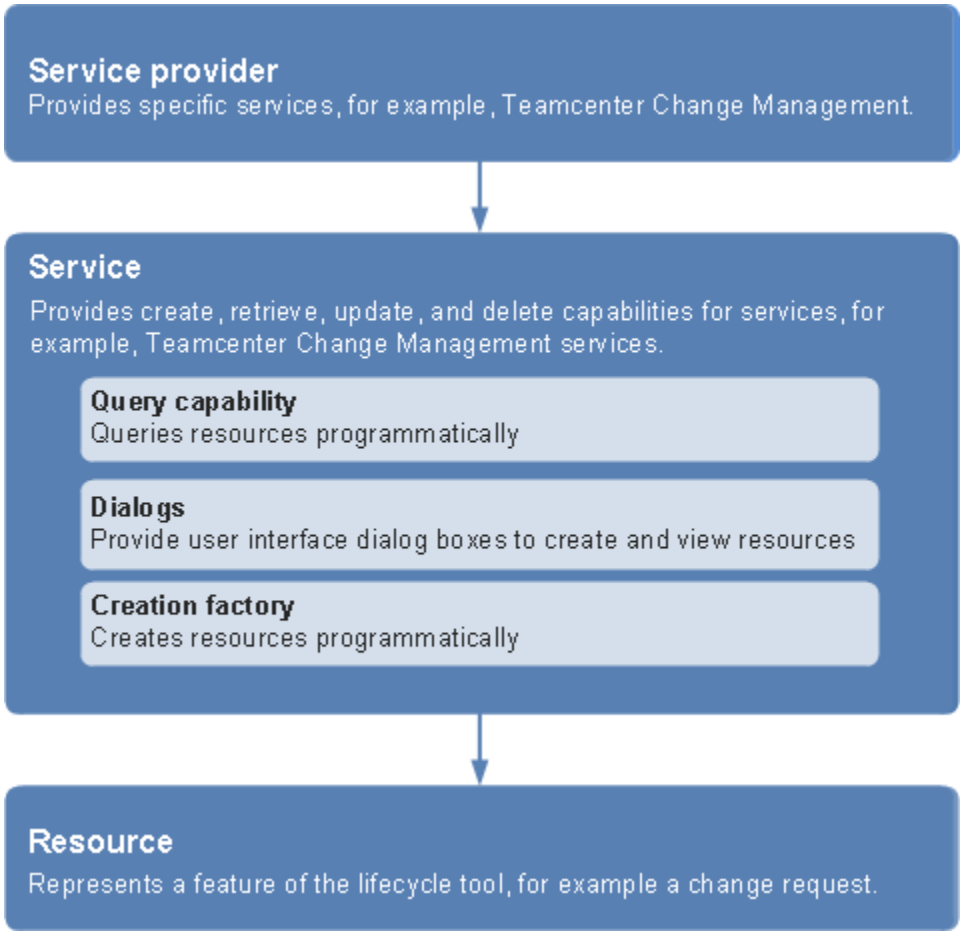
How does Linked Data Framework work?

Linked Data Framework supports Open Services for Lifecycle Collaboration (OSLC) specification. It uses open technologies such as Linked Data, RDF, and REST.

Representational State Transfer (REST)	Provides API specifications that allow you to use HTTP methods to create, retrieve, update, and delete resources.
Linked Data	Allows you to represent each resource as a URI and provides HTTP links to these resources.
Resource Description Framework (RDF)	Allows you to specify information about the resource.

Once Linked Data Framework is setup, you can create links between resources, view data, and update resources.

To support the integration, Linked Data Framework uses the following framework:



The Linked Data Framework framework contain the following components:

Framework elements	Description
Service provider	A service provider provides specific services, for example, Teamcenter Change Management.
Service	A service provides the capabilities to create, retrieve, update, and delete resources for a particular Linked Data Framework area, for example, Teamcenter Change Management. A service may offer the following capabilities:

Framework elements	Description	
	Query capabilities	Allows you to query a resource programatically
	Dialogs	Allows you to create UI dialogs to create and view resources
	Creation Factory	Allows you to create resources programmatically
Resource	A resource represents a feature of the application, for example Change Request.	

By default Change Management, Core Services, and Requirement Management domains are available. You can **add other domains** as per your requirements.

The Core Services catalog provides services to create, query, update, and delete generic Teamcenter objects such as item, document, parts using UI-based or API-based interfaces. Integrators can also add custom services for processing the core or custom business objects.

Overview of exposing additional services using Linked Data Framework

A third-party application can integrate with Teamcenter functionality such as Requirements Management only when the functionality is configured to support Linked Data Framework.

To enable Linked Data Framework support, you must perform the following in Business Modeler IDE:

- Create a service catalog.

  A service catalog acts as a container for a related set of functionality such as Requirement Management. It specifies the types of business objects that support Linked Data Framework.

- Create a Linked Data Framework service.

  A Linked Data Framework service specifies which services are supported. Services are of two types:

  - *Factory services* provide programmatic interfaces for performing operations such as creating and querying objects.

  - *Delegated user interface (UI) services* provide a UI-based service. The UI is rendered using Active Workspace. This service is defined with a URI pointing to an Active Workspace UI designed for that service.

- If you create a factory service, you must also specify service operations. A service operation is an ITK that is part of a library. For each factory service, all REST operations can be defined. For every service, you can associate four different REST-operation implementations: **GET**, **POST**, **PUT**, and **DELETE**.

- To support the new service catalog, create a new class based on the **Osl0ResourceAttrHelper** class, and register the class using the **create_botype_reader** utility.

- After you create service catalogs, services, service operations, and the new class in Business Modeler IDE, deploy the template to Teamcenter.

- Create a dataset in Teamcenter and attach an XML file that contains mapping information. Register the dataset using the **create_botype_reader** utility.

By default, Change Management and Core Services catalog are available. You can add other domains as per your requirements.

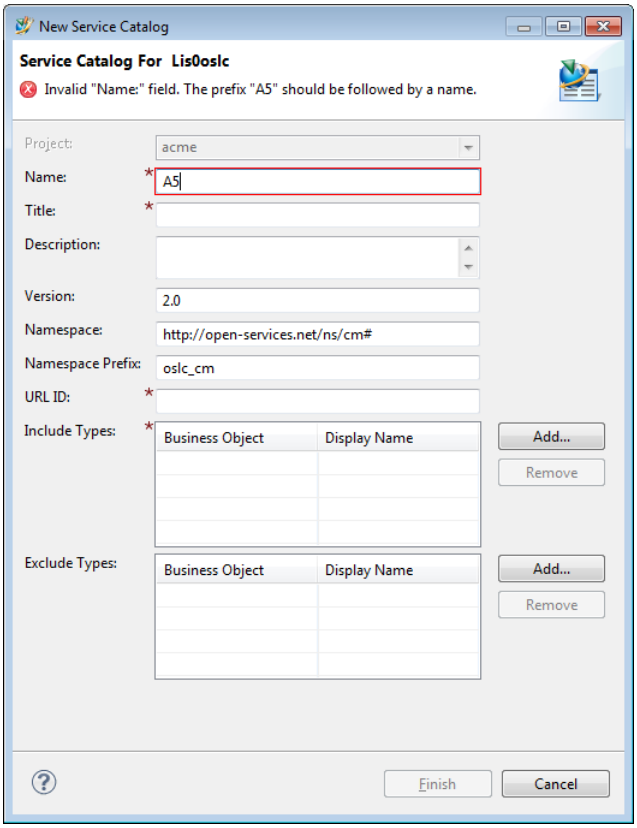
The Core Services catalog provides services to create, query, update, and delete generic Teamcenter objects such as items, documents, and parts by using UI-based or API-based interfaces. Integrators can also add custom services for processing the core or custom business objects.

Create a new service catalog

The Linked Data Framework provider catalog typically reflects specific functionality in Teamcenter, and it is referred to as a service catalog in the Business Modeler IDE Linked Data Framework registry. The change management domain is provided in Linked Data Framework by the **Lis0CM** catalog. You can create your own catalog to represent other areas of functionality in Teamcenter.

1. Open the **Extensions\Linked Data Services** folders, right-click an already-created protocol, and choose **New Service Catalog**.

   The **New Service Catalog** dialog box is displayed.



2. Enter the following information in the **Service Catalog** dialog box:

   a. In the **Name** box, type the name you want to assign to the new catalog.

   b. In the **Title** box, type the display name to the new catalog.

   c. In the **Description** box, state the purpose of the catalog.

   d. In the **Version** box, type the protocol version supported by the catalog.

   e. In the **Namespace** box, type the namespace path.

   f. In the **Namespace Prefix** box, type the prefix you want to assign.

   g. In the **URL ID** box, type the uniform resource identifier to form the URL for this resource.

   h. In the **Include Types** table, select the list of business objects that the catalog supports.

   i. In the **Exclude Types** table, select the list of business objects that the catalog does not support.

   j. Click **Finish**.

The new catalog appears under the selected protocol instance.

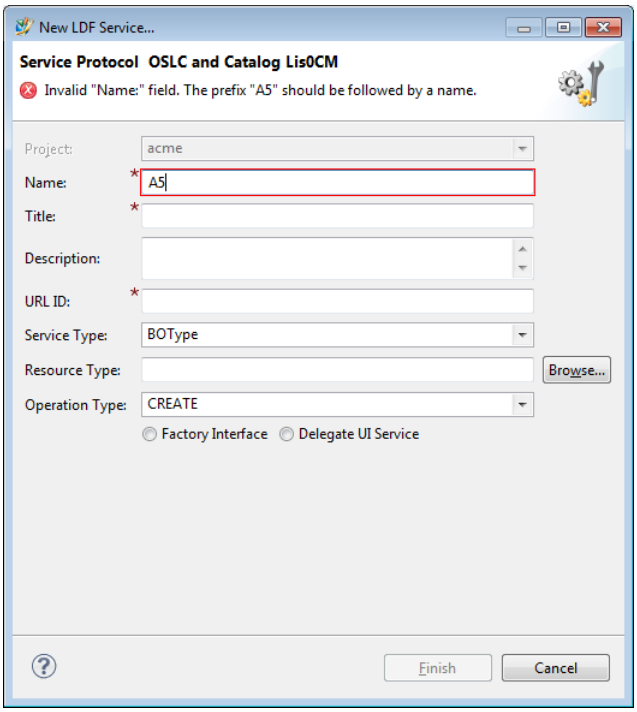
3. To save the changes to the data model, choose **BMIDE→Save Data Model** on the menu bar or click the **Save Data Model** button on the main toolbar.

Create a new Linked Data Framework service

The Linked Data Framework services are used to create factory APIs that are used programmatically to link to external applications. These services are defined for a given catalog. If you **create a catalog**, you must create the services comprising the catalog.

1. Open the **Extensions\Linked Data Services** folders, right-click the already-created catalog, and choose **New LDF Service**.

   The **New LDF Service** dialog box is launched.



2. Enter the following information in the **New LDF Service** dialog box:

   a. In the **Name** box, type the name you want to assign to the new service.

   b. In the **Title** box, type the display name to the new service. This appears in the service catalog.

   c. In the **Description** box, state the purpose of the service.

   d. In the **URL ID** box, type the uniform resource identifier. This forms the URL for the resource.

e. Click the arrow in the **Service Type** box to select the kind of service type, either a business object type, generic, or an instance type.

- **BOType**

  Specifies that the service being defined is based on a business object type.

- **Generic**

  Specifies that the service being defined does not apply to a specific business object type.

- **Instance**

  Specifies services for update, delete, and preview for a specified business object instance.

f. Click the **Browse** button in the **Resource Type** box, and select the business object type, such as **ChangeItem**.

g. Click one of the following buttons:

- **Factory Interface**

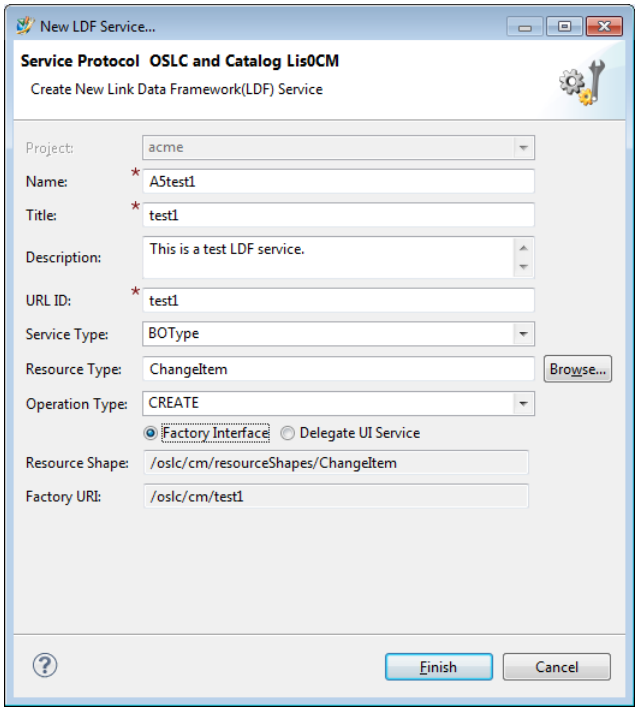
  Creates the service factory. When you click **Factory Interface**, the following boxes are populated:

  - **Resource Shape**

    Holds the URI required to get the resource shape (a data dictionary in RDF/JSON output format) for a given business object type.

  - **Factory URI**

    Holds the URI for invoking this factory service.



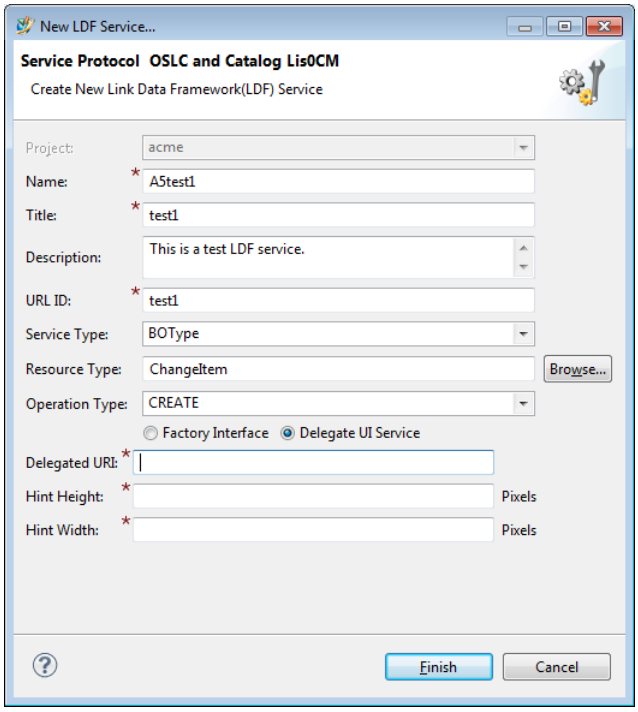
- **Delegate UI Service**

  Creates the service dialog box. Perform the following steps:

  A. In the **Delegated URI** box, specify the delegated user interface URI for this service implementation.

  B. In the **Hint Height** box, specify the value for the height of the **Delegated UI** dialog box in pixels.

  C. In the **Hint Width** box, specify the value for the width of the **Delegated UI** dialog box in pixels.



3. To save the changes to the data model, choose **BMIDE→Save Data Model** on the menu bar, or click the **Save Data Model** button on the toolbar.

Create a Linked Data Framework service operation

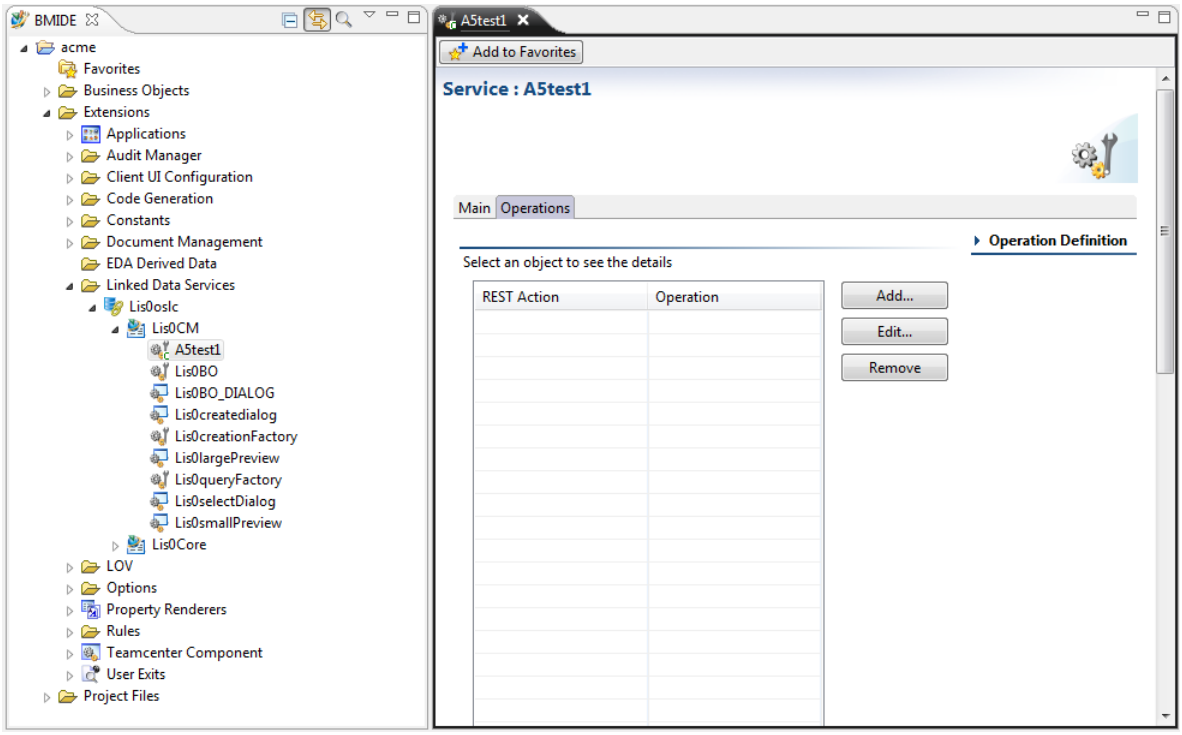
For every service, you can associate four possible different REpresentational State Transfer (REST) operation implementations: **GET**, **POST**, **PUT**, and **DELETE**.

1. Expand the **Extensions\Linked Data Services** folders.

2. Open the service factory to hold the new operation.

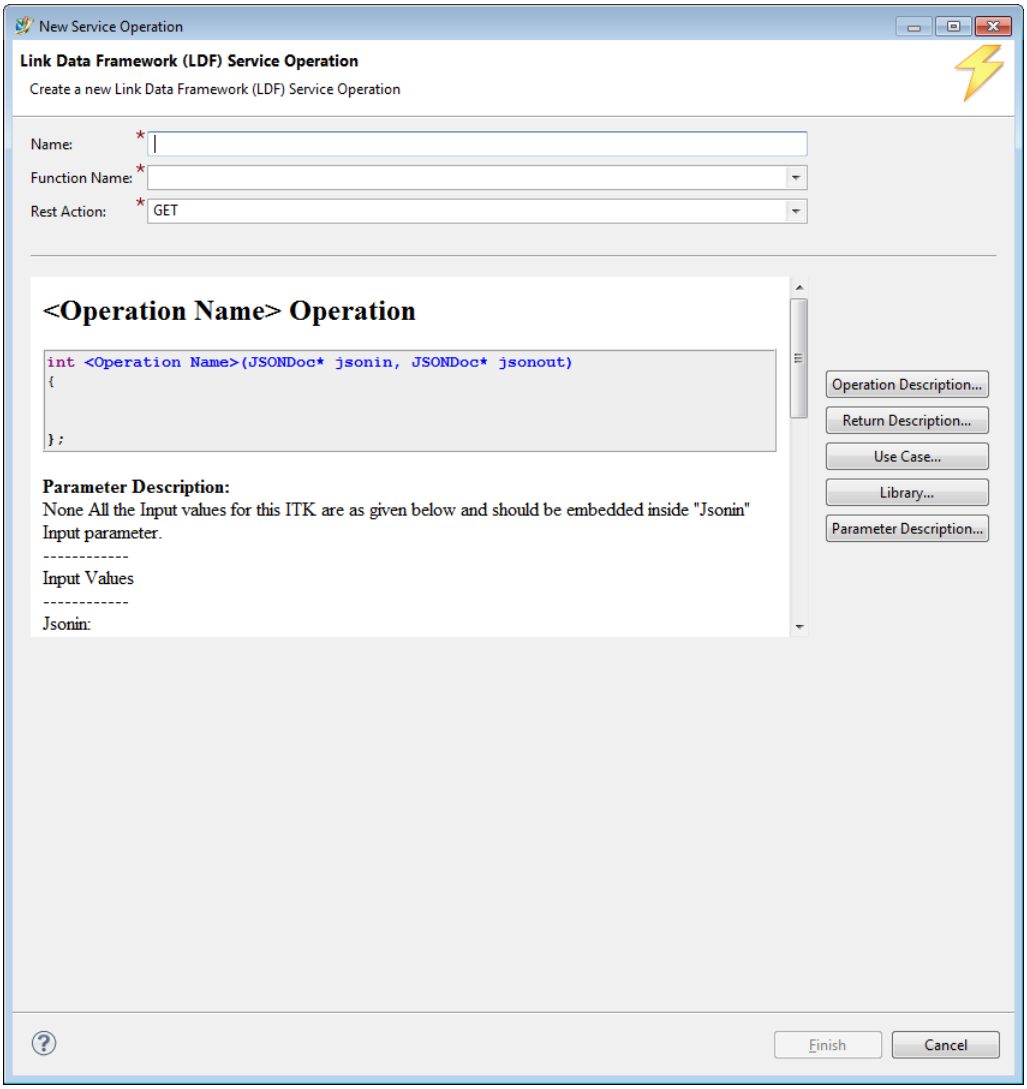
> Tip:
>
> You must create a service factory before you create a service operation.

3. Right-click the service factory in which you want to create the new operation, choose **Open**, and click the **Operations** tab.



4. Click the **Add** button on the **Operations** tab.

   The **New Service Operation** wizard is launched.



5. Perform the following steps in the **Link Data Framework (LDF) Service Operation** dialog box:

   a. In the **Name** box, type a unique name for that operation.

   b. In the **Function name** box, type the API you want to assign for executing the service operation.

   c. Click the arrow in the **REST Action** box to select the REST operation type: **GET**, **PUT**, **POST**, or **DELETE**. Only one operation with the selected REST action can be active.

   d. Click the **Operation Description** button to type a complete description of the functionality exposed through the service operation.

      The **Description Editor** is displayed.

      Follow these best practices:

- Describe what this operation does. Explain more than simply stating the method name.
- Make the description complete in its usefulness. Keep in mind the client application developer while writing the content.
- Whenever appropriate, describe how each argument interrelates when this operation completes.
- Use correct formatting with fixed and bold text where appropriate.
- Use correct Teamcenter terminology.
- Define acronyms before using them.

e. Click the **Return Description** button to type a complete description of what the service operation returns. Follow these best practices:

- Describe what the output represents and provide high-level details of the output data. Do not specify only the type of service data returned.
- Describe any partial errors returned.
- Specify returned objects that are created, updated, or deleted as part of service data.
- Use complete sentences.
- Use correct formatting with fixed and bold text where appropriate.
- Use correct Teamcenter terminology.
- Define acronyms before using them.

f. Click the **Use Case** button to describe how the user interacts with this operation to accomplish the operation's goal. Follow these best practices:

- Document when and why this operation can be consumed.
- Describe how operations interrelate to satisfy the use case (if there is interrelation between operations).
- Use complete sentences.
- Specify all possible use cases.
- Use correct formatting with fixed and bold text where appropriate.

- Use correct Teamcenter terminology.
- Define acronyms before using them.

g. Click the **Library** button to select the library where the API is defined, for example, **Lis0lisfmwrk**.

h. Examine the contents of the preview pane.

i. Click **Finish**.

The new service operation displays on the **Operations** tab. To see the characteristics of the operation, select it and click **Operation Definition** on the right side of the editor.

6. To save the changes to the data model, choose **BMIDE→Save Data Model** on the menu bar, or click the **Save Data Model** button on the main toolbar.

7. You must also implement the operation in C++. A sample implementation of a GET operation is as follows:

```
#include <fclasses/tc_string.h>
#include <tc/tc.h>
#include <fclasses/tc_string.h>
#include <base_utils/ScopedPtr.hxx>
#include <base_utils/ScopedSmPtr.hxx>
#include <base_utils/SharedPtr.hxx>
#include <base_utils/Mem.h>
#include <tc/tc_util.h>
#include <fclasses/tc_types.h>
#include <tccore/aom_prop.h>
#include <Lis0lisfmwrk/LisfmwrkUtils.hxx>
#include <Lis0lisfmwrk/lis_defines.h>
#include <Lis0lisfmwrk/Lis0lisfmwrk_errors.h>
#include <Lis0lisfmwrk/toolkit/AbstractResource.hxx>
#include <Lis0lisfmwrk/lis_factory.h>
#include <Lis0lisfmwrk/Lis0JsonWriter.hxx>
#include <metaframework/BusinessObject.hxx>
#include <metaframework/BusinessObjectRef.hxx>


using namespace Teamcenter;
using namespace std;
using namespace lis0lisfmwrk;

/**
Retrieves an object factory Lifecycle Interoperability Services (LIS) service
<br/>It will retrieve the object information (attributes names and values) based on
resourceshape.
<br/>Open Services for Lifecycle Collaboration (OSLC) defines the ResourceShape
resource
 to allow the specification of a list of properties with allowed values.
In teamcenter, OSLC resource will be the object type and its properties.
<br/>This object information is returnnd in JSON format.
```

```
<br/>This method is invoked in case of get object factory request given from REST
client
 for GET rest action.
<br/>e.g. REST client request.
<br/>Rest action - GET.
<br/>URL http://localhost:8080/lis/oslc/cm/BO?uid=g0bRF0$RI_JGwA

@returns
<ul>
<li>#ITK_ok on success.
<li>#LIS0LISFMWRK_valid_attr_helper_not_found if the LisAttributeHelper is not valid.
<br/>The LisAttributeHelper maps the Teamcenter properties with Open Services for
Lifecycle
 Collaboration (OSLC) core properties. It can be created through the utility
'create_botype_reader'
<li>#LIS0LISFMWRK_valid_writer_not_found if valid media writer not for generating
output JSON
 xml.
</ul>
*/
int LIS_get_object_factory( const char *request_body,   /**< (I) Request BODY in
JSON format. */
                            char **response             /**< (OF) Response in JSON
format. */
                          )
{
    int ifail = ITK_ok;
    ResultStatus rstat;
    PomResultStatus pomStat;
    std::string responseStr;
    try
    {
        std::string reqMediaType =  "application/json";
        tag_t mediaWriterTag =
lis0lisfmwrk::Lis0MediaWriterImpl::getMediaWriter( reqMediaType,
         true );
        if ( mediaWriterTag == NULLTAG )
        {
            rstat = LIS0LISFMWRK_media_writer_not_exist;
        }
        POMRef<lis0lisfmwrk::Lis0MediaWriterImpl> mediaWriterImpl( mediaWriterTag );
        std::string reqBody = request_body;
        lis0lisfmwrk::AbstractResource *resourceModel = new
lis0lisfmwrk::AbstractResource();
        scoped_ptr <lis0lisfmwrk::AbstractResource> freeResource1( resourceModel );
        if ( mediaWriterImpl != 0 )
        {
             // write the document
             rstat = mediaWriterImpl->lis0ReadFrom( &reqMediaType, &reqBody,
&resourceModel );
        }
        //Get the urlParameter attribute(UID=XXX) here from JSON
        std::string uidParam = resourceModel-
>getSimpleStringPropertyValue(URL_PARAMETER);
        std::string catalogTitle = resourceModel-
>getSimpleStringPropertyValue(CATALOG_TITLE);
        std::string queryParam = uidParam;
        std::string objUid = "";
        LisDataUtils::parseRequestParamaterToGetUid(queryParam, &objUid);
        tag_t objTag = NULLTAG;
```

```
        lis0lisfmwrk::AbstractResource *outResourceModel = 0;
        Lis0ResourceShapeUtils *resourceShapeUtils = new
lis0lisfmwrk::Lis0ResourceShapeUtils();
        scoped_ptr <lis0lisfmwrk::Lis0ResourceShapeUtils>
freeUtil( resourceShapeUtils );
        if(!objUid.empty())
        {
            bool tag_valid = false;
            rstat = LisDataUtils::checkObjectUIDIsValidOrNot(objUid, tag_valid,
objTag);
            if(tag_valid)
            {
                 rstat = resourceShapeUtils->getBoResourceShape( objTag, objUid,
catalogTitle,
                 &outResourceModel );
            }
        }
        else
        {
            throw IFail( LIS0LISFMWRK_input_invalid_request);
        }
        std::string requiredMediaType =  "application/json";
        // write the resource model to document
        rstat = LisDataUtils::addEtagToResourceModel(objTag, &outResourceModel);
        rstat = LisDataUtils::generateOutputFromResourceModel( outResourceModel,
        &requiredMediaType, &responseStr );
        scoped_ptr<lis0lisfmwrk::AbstractResource>
freeoutResourceModel( outResourceModel );
        *response = SM_string_copy(responseStr.c_str());
    }
    catch(const IFail& ex)
    {
        ifail = ex.ifail();
        Teamcenter::Lis0lisfmwrk::logger()->error( ERROR_line, ifail,
        "LIS_get_object_factory failed with error[%d]\n",
        ifail );
    }
    return ifail;
}
```

Create a new class to support the newly created service catalog

To support the new service catalog:

1. Create a new class as a subclass of the **Osl0ResourceAttrHelper** class.

   For example, if the new service catalog supports Requirement Management, the class name could be **Cus0RMResourceAttrHelper**.

2. Implement the **lis0updateCompundAttributesBase()** API if you require custom processing when retrieving or updating properties on a business object that is supported by the service catalog.

3. To register this class in Teamcenter, run the **create_botype_reader** utility with information about the new class.

For example:

```
create_botype_reader -u=Tc-admin-user -p=password -g=group
-mode=resource
-type=Cus0RMResourceAttrHelper -protocol=oslc -versions=2.0
-domain=rm
```

where

-domain specifies the URL identifier of the service catalog, and

-type specifies the name of the newly created class.

After you create service catalogs, services, service operations, and the new class in Business Modeler IDE, deploy the template to Teamcenter.

8. Customizing Linked Data Framework

Customize mapping between Teamcenter and external applications using the semantic mapping file

For data linking to work between Teamcenter and the external application, information on how domains, objects, object properties, and relations are mapped between the two application need to be defined (also known as semantic mapping). This information is defined for each domain as XML files.

For example, by default Teamcenter supports the core, and change management domains. Their semantic mapping files are available in the *TC_ROOT\install\osl0lisoslc* directory and are named as follows:

- *lis_core_properties.xml*
- *lis_cm_properties.xml*

When you create new domains you need to create a new semantic mapping file for that domain.

The structure of a semantic mapping file

```
<DomainProperties>

  <NamespaceURI>
    <!-- Specifies the namespaces for Teamcenter and OSLC services -->
  </NamespaceURI>

  <TypeURIMapping>
      <!-- Specifies the URI for a particular semantic type. For example
URI for
           Change Request -->
  </TypeURIMapping>

  <ExternalSemanticType>
    <!-- Specifies details for a particular semantic type. -->
  </ExternalSemanticType>

  <TypeMapping>
      <!-- Specifies the Teamcenter object types that are mapped to a
particular
           semantic type -->
  </TypeMapping>

  <TypeAttrMapping>
      <!-- Specifies the mapping between attributes of Teamcenter object
```

```
types and the
           attributes of the semantic type -->
  </TypeAttrMapping>

  <RelationTypeMapping>
      <!-- Specifies the mapping between relations of Teamcenter object
types and the
           relations of the semantic type. -->
  </RelationTypeMapping>

</DomainProperties>
```

NamespaceURI element

Specifies namespace information. Namespaces allow organization of data into separate categories such for change management or requirements management.

```
<NamespaceURI>
  <Namespace namespacePrefix="tc"
   namespaceValue="http://www.plm.automation.siemens.com/ldf#"/>
  <Namespace namespacePrefix="oslc"
   namespaceValue="http://open-services.net/ns/core#"/>
  <Namespace namespacePrefix="oslc_cm"
   namespaceValue="http://open-services.net/ns/cm#"/>
  <Namespace namespacePrefix="oslc_rm"
   namespaceValue="http://open-services.net/ns/rm#"/>
  <Namespace namespacePrefix="dcterms"
   namespaceValue="http://purl.org/dc/terms/"/>
</NamespaceURI>
```

This element contains the following tag:

Tag name	Tag attribute	Attribute description
Namespace	**namespacePrefix**	Name of the namespace
	namespaceValue	Value or URI of the namespace

TypeURIMapping element

Specifies the URI of the semantic type. Semantic type refers to an OSLC resource such as Change Request or Requirement.

```
<TypeURIMapping>
```

```
  <SemanticType name="ChangeRequest"
   uri = "http://open-services.net/ns/cm#ChangeRequest" />
  <SemanticType name="Requirement"
   uri = "http://open-services.net/ns/rm#Requirement" />
</TypeURIMapping>
```

This element contains the following tag:

Tag name	Tag attribute	Attribute description
SemanticType	**name**	Name of the semantic type
	uri	Unique resource identifier of the semantic type

ExternalSemanticType element

Specifies details for a specific semantic type such as Change Request.

```
  <ExternalSemanticType domain="cm" name="ChangeRequest"
extends="CoreBO">
    <CreationDelegatedUI urlId = "creator"/>
    <SelectionDelegatedUI urlId = "selector"/>
    <AttrDef name="oslc_cm:closeDate"
     ldfOccurs="zero_or_one" ldfValueType="String" tcValueType="string"
         ldfDescription="The date at which no further activity or work
         is intended to be conducted."
         ldfRepresentation="na" ldfRange="na" ldfTitle="CloseDate"
         ldfReadOnly="true" />
    <AttrDef name="oslc_cm:status" ldfOccurs="zero_or_one"
     ldfValueType="String" tcValueType="string"
         ldfDescription="Used to indicate the status of the change
request
         based on values defined by the service provider. Most often a
         read-only property. Some possible values may include:
         'Submitted', 'Done', 'InProgress', etc."
         ldfRepresentation="na" ldfRange="na" ldfTitle="Status"
         ldfReadOnly="true" />
  </ExternalSemanticType>
```

This element contains the following tags:

Tag name	Tag attribute	Attribute description
ExternalSemanticType	**domain**	The domain name. This value must match the service catalog name in Business Modeler IDE.
	name	Name of the semantic type
	extends	The semantic type from where this semantic type is derived
Creation-DelegatedUI	**urlId**	The URL ID of the Creation Delegated UI. This value must match the value in the URL ID field of the create dialog service in Business Modeler IDE.
Selection-DelegatedUI	**urlId**	The URL ID of the Selection Delegated UI. This value must match the value in the URL ID field of the select dialog service in Business Modeler IDE.
AttrDef	**name**	Name of the attribute, for example, **oslc_cm:closeDate**.
	ldfOccurs	The number of times the attribute occurs. Valid values are: • **exactly_one** • **zero_or_one** • **zero_or_many** • **one_or_many** • **any**
	ldfValueType	The type value of the attribute based on OSLC specifications. Valid values are: • **boolean** • **decimal** • **double** • **float** • **integer** • **date_time** • **string** • **xml_literal** • **resource** • **local_resource** • **typed_reference** • **untyped_reference** • **external_reference** • **typed_relation** • **untyped_relation** • **untyped**

Tag name	Tag attribute	Attribute description
	tcValueType	Teamcenter type values that are mapped to OSLC types. Valid values are: • **logical** • **short** • **double** • **float** • **integer** • **date_time** • **string** • **char** • **typed_reference** • **untyped_reference** • **external_reference** • **typed_relation** • **untyped_relation** • **untyped** • **note**
	ldfDescription	Description of the property
	ldfRepresentation	Specifies how the attribute is represented as per OSLC specifications. Valid values are: • **reference** • **inline** • **either** • **na**
	ldfRange	Specifies the range of the property as per OSLC specifications. Valid values are: • **na** • **any**
	ldfTitle	Specifies the title of the attribute
	ldfReadOnly	Specifies whether clients can change attribute value after the resource is created. Valid values are: • **true** • **false**

TypeMapping element

Specifies the Teamcenter object types that are mapped to a semantic type.

```
<TypeMapping targetType="ChangeRequest">
    <TcTypeMapping srcTcType="ChangeRequest"/>
    <TcTypeMapping srcTcType="ProblemReport"/>
</TypeMapping>
```

This element contains the following tag:

Tag name	Tag attribute	Attribute description
TypeMapping	**targetType**	Name of the semantic type
TcTypeMapping	**srcTcType**	Teamcenter object type. This name must be the same as the name of the object type in Business Modeler IDE. Case sensitive.

TypeAttrMapping element

Maps the attributes of the Teamcenter object types to the semantic types. The Teamcenter object types were define in the **TypeMapping** element and the semantic types were defined in the **ExternalSemanticType** element.

The attributes of a semantic type can be mapped to multiple Teamcenter object types.

```
<TypeAttrMapping targetType="ChangeRequest">
    <AttrMapping srcTcType="ChangeRequest" srcTcAttr="releaseDate"
     targetSemanticAttr="oslc_cm:closeDate"/>
    <AttrMapping srcTcType="ProblemReport" srcTcAttr="status"
     targetSemanticAttr="oslc_cm:status"/>
</TypeAttrMapping>
```

This element contains the following tags:

Tag name	Tag attribute	Attribute description
TypeAttrMapping	**targetType**	Name of the semantic type
AttrMapping	**srcTcType**	Teamcenter object type.
	srcTcAttr	Teamcenter object type attribute
	targetSemanticAttr	Attribute of the semantic type

Note:

The **releaseDate** attribute from **srcTcType** is mapped to **oslc_cm:closeDate** attribute. This mapping is applicable only for the Teamcenter type Change Request and its subclasses.

The **status** attribute from **srcTcType** is mapped to **oslc_cm:status** attribute. This mapping is applicable only for the Teamcenter type ProblemReport and its subclasses.

RBTypeAttrMapping element

Maps semantic types to the runtime Teamcenter object types. These runtime Teamcenter object types are used to represent the semantic types in the Relations application.

By default, the **Ldf0TempLinkRuntime** object type is mapped to the **CoreBO** semantic type and the **Lcm1ChangeRequest** object type is mapped to the **ChangeRequest** semantic type.

You must map other custom object types to the semantic types.

```
<RBTypeAttrMapping  SemanticType="ChangeRequest"
RBType="Lcm1ChangeRequest">
     <RBAttrMapping SemanticAttr="status" RBTypeAttr="lcm1Status"/>
     <RBAttrMapping SemanticAttr="closeDate"
RBTypeAttr="lcm1CloseDate"/>
</RBTypeAttrMapping>
```

This element contains the following tags:

Tag name	Tag attribute	Attribute description
RBTypeAttr-Mapping	**SemanticType**	Name of the semantic type
	RBType	Name of the Teamcenter runtime object type
RBAttrMapping	**SemanticAttr**	Attribute name of the semantic type
	RBTypeAttr	Attribute name of the runtime Teamcenter object type

RelationTypeMapping element

Maps Teamcenter relation type with the relation type of the semantic element.

```
<RelationTypeMapping targetType="CoreBO">
    <TcRelationMapping srcTcRelationType="Lcm0AffectsRequirement"
```

```
     consumerRelation="oslc_cm:affectsRequirement"
     providerRelation="oslc_cm:relatedChangeRequest"/>
    <TcRelationMapping srcTcRelationType="Lcm0ImplementsRequirement"
     consumerRelation="oslc_cm:implementsRequirement"
     providerRelation="oslc_cm:relatedChangeRequest"/>
</RelationTypeMapping>
```

If the external application uses a custom relation that is not part of the OSLC specification, ensure that you update this section, else backlinks may not be created.

If you have created a custom object, ensure that the relations are defined as properties of the custom object in Business Modeler IDE. If you do not do this, the links will not appear in the **Remote Links** table after they are created.

This element contains the following tags:

Tag name	Tag attribute	Attribute description
RelationType-Mapping	**targetType**	Name of the semantic type
TcRelation-Mapping	**srcTcRelationType**	Teamcenter relation type
	consumerRelation	Specifies the relation to use when Teamcenter as a consumer creates remote links in the external application. In case of backlink creation from the external application, this will create the relation specified in the **srcTcType** attribute.
	providerRelation	Specifies what relation to use when creating backlinks to the external application.

Post update steps

After updating the semantic mapping file perform the following steps:

1. Validate the semantic mapping file using the **create-botype_reader** utility. When you use the **-f=validate** argument, this utility validates the Teamcenter types as well as the types specified in the **extends** clause.

   ```
   create_botype_reader -u=Tc-admin-user -p=${TC_USER_PASSWD} -g=group
   -mode=map_schema
   -domain=core -file=${TC_INSTALL_DIR}/osl0lisoslc/
   lis_custom_properties.xml
   -f=validate
   ```

The output of this utility run is a file named *semantic_data.csv*. This file lists the **TypeMapping** and **RelationTypeMapping** entries in your semantic mapping file. Check if the mapping is correct.

2. Store the semantic mapping file to the database using the **create-botype_reader** utility. When you use the **-f=create** argument, this utility stores the semantic mapping file to the database.

```
create_botype_reader -u=Tc-admin-user -p=${TC_USER_PASSWD} -g=group
-f=create
-mode=map_schema -domain=core
-file=${TC_INSTALL_DIR}/osl0lisoslc/lis_custom_properties.xml
```

3. (Optional) List the semantic types: Using the **create-botype_reader** utility you can list the semantic mapping entries that were saved to the database. Use the **-f=list** argument.

```
create_botype_reader -u=Tc-admin-user -p=${TC_USER_PASSWD} -g=password
-mode=map_schema
-domain=core -f=list -file=d:\sematic_data.csv
```

Configuring the Creation Delegated UI dialog

When Teamcenter acts as a *Provider*, the creation dialog (Teamcenter interface) rendered in the external application shows the list of Teamcenter types. To make these types appear, do the following:

- Update the **TypeMapping** element of the semantic mapping file.

  Ensure you update the correct semantic mapping file. The semantic types are available in the following location:

  - Core: *TC_ROOT\install\osl0lisoslc\lis_core_properties.xml*
  - Change management: *TC_ROOT\install\osl0lisoslc\lis_cm_properties.xml*
  - Requirement management: *TC_ROOT\install\ lrm0lisrm \lis_rm_properties.xml*

  The **TypeMapping** element must have the **targetType** and **srcTcType**.

  - **targetType**: Specifies semantic type
  - **srcTcType**: Specifies Teamcenter types

  For example, while adding links to the remote system, if you want to add *Part* and *System Block* types in addition to the already existing Item type as part of the CoreBO semantic type, update the **TypeMapping** element as follows:

```
<TypeMapping targetType="ExtendedChangeRequest">
    <TcTypeMapping srcTcType="CustomChangeRequest"/>
    <TcTypeMapping srcTcType="CustomProblemReport"/>
</TypeMapping>
```

> Note:
>
> Abstract Teamcenter types must not be mapped to the semantic types.

- If you are updating existing semantic mapping files, delete the named references of these semantic mapping files that are associated with the following datasets:
  - Core: Dataset = *oslc_core_attr_ds*, Named reference = *lis_core_properties.xml*
  - Change management: Dataset = *oslc_cm_attr_ds* Named reference = *lis_cm_properties.xml*
  - Requirement management: Dataset = *oslc_rm_attr_ds* Named reference = *lis_rm_properties.xml*
- Run the **create-botype_reader** utility using the **–f=create** argument to update the semantic mapping file.

Create custom icons for semantic types

You can customize the type icons for semantic types using Active Workspace configuration methods. See the *Adding new type and tile icons* topic in the Active Workspace *Configuration and Extensibility* documentation for information on how to add custom icons for semantic types.

Define the relations to apply when creating remote links

You can define what relations to apply when you create a remote link between Teamcenter and the external application. For example, you can define that when you create a remote link between a change request in Teamcenter and an issue in the external application, the relation can be either **Lis0Reference** or **Lis0Specification**. You can apply these relation definitions for each service provider. To create the relation definitions:

1. Define the Teamcenter objects, objects in the external application, and the relation between them in an XML file. Give a meaningful name to the file, such as *semantic_data.xml*. Specify the definitions in the **SemanticKey** element of the XML file as follows:

```
<SemanticKey>
    <TcSemanticType>ChangeRequest</TcSemanticType>
    <RemoteSemanticType>CoreBO</RemoteSemanticType>
    <SemanticRelationType>Lis0Reference</SemanticRelationType>
```

```
        <SemanticRelationType>Lis0Specification</SemanticRelationType>
    </SemanticKey>
```

Consider the following when creating the XML file:

- The **TcSemanticType** is the Teamcenter object type.
- The **RemoteSemanticType** is the object type in the external application.
- The **SemanticRelationType** is the Teamcenter relation type.
- If you do not create a definition for a Teamcenter object type, the parent object definition is used when you create a remote link. For example, if no definition is created for change notice, the change item definition is used when creating remote links. Similarly, if no definition is created for the change item, the item definition is used when creating remote links.

2. Add these definitions to the database using the **maintain_ldf_semantics** utility, using the **–create** argument.
3. Apply the definitions for each service provider as follows:
    a. In Organization, from the **Organization List→Lifecycle Data Framework Providers**, select a service provider.
    b. In the **Semantics** list, click **Add/Remove**.
    c. In the **Semantics** dialog box, select the semantic relations you want from the **Available Semantics** list and add them to the **Selected Semantics** list.
    d. Click **OK**.
    e. Click **Modify**.
4. Verify if the relation types appear when you create a remote link.

Example of the XML file containing definitions:

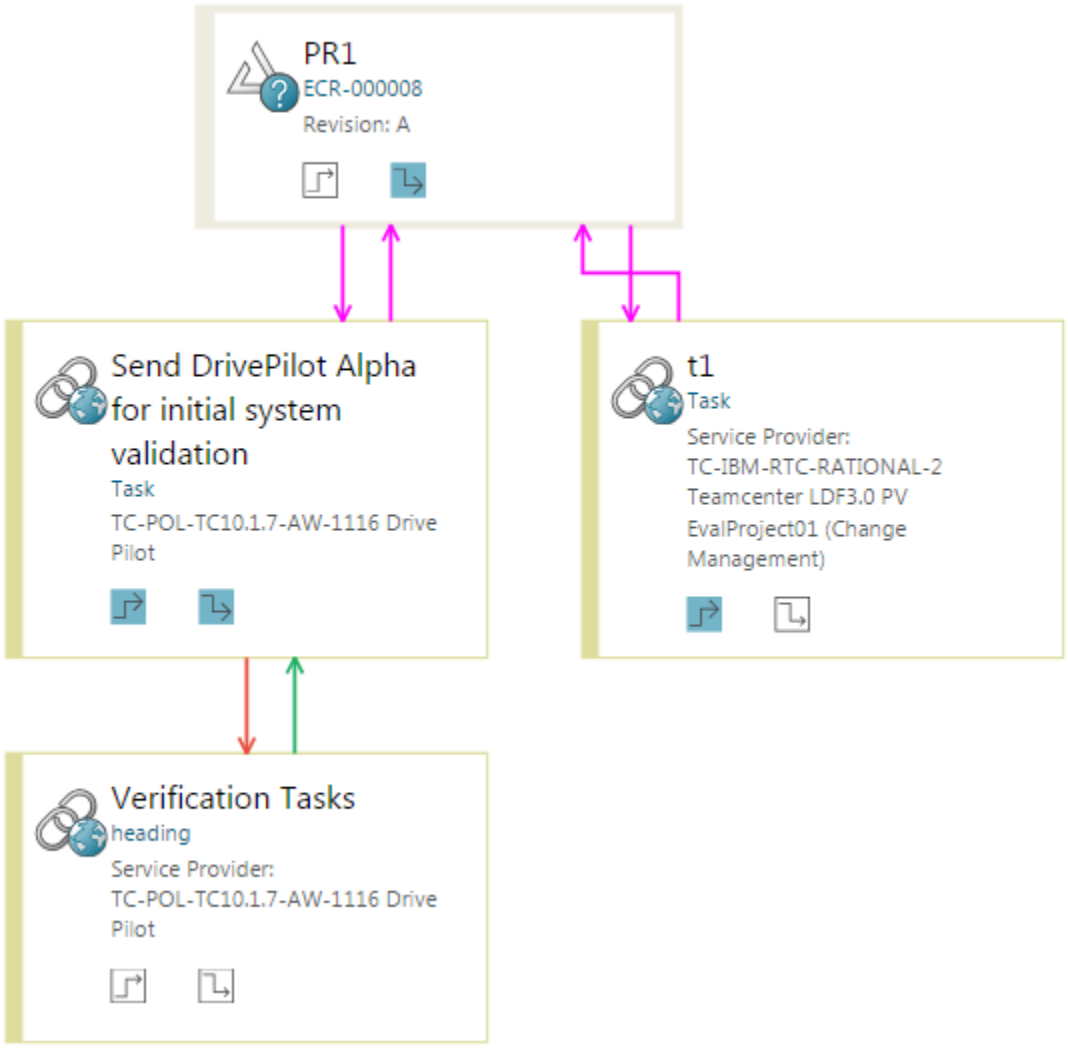
```
<?xml version="1.0" encoding="UTF-8"?>
<SemanticKey>
    <TcSemanticType>CoreBO</TcSemanticType>
    <RemoteSemanticType>ChangeRequest</RemoteSemanticType>
    <SemanticRelationType>Lis0Reference</SemanticRelationType>
    <SemanticRelationType>Lis0Specification</SemanticRelationType>
</SemanticKey>
<SemanticKey>
    <TcSemanticType>CoreBO</TcSemanticType>
```

```
    <RemoteSemanticType>Requirement</RemoteSemanticType>
    <SemanticRelationType>Lis0Reference</SemanticRelationType>
    <SemanticRelationType>Lis0Specification</SemanticRelationType>
</SemanticKey>

<SemanticKey>
    <TcSemanticType>ChangeRequest</TcSemanticType>
    <RemoteSemanticType>CoreBO</RemoteSemanticType>
    <SemanticRelationType>Lis0Reference</SemanticRelationType>
    <SemanticRelationType>Lis0Specification</SemanticRelationType>
</SemanticKey>
<SemanticKey>
    <TcSemanticType>Requirement</TcSemanticType>
    <RemoteSemanticType>CoreBO</RemoteSemanticType>
    <SemanticRelationType>Lis0Reference</SemanticRelationType>
    <SemanticRelationType>Lis0Specification</SemanticRelationType>
</SemanticKey>
```

Customizations to show Linked Data Framework objects and relations in the Relations tab

The **Relations** tab in Active Workspace allows you to view the relationship between objects in Teamcenter.



For more information about configuring the Relations application, see the Relations configuration.

You can add objects and relations supported by Linked Data Framework to views in the Relations application as follows:

1. Create runtime object types in Business Modeler IDE that correspond to the object types in the external application by extending the existing Teamcenter runtime objects. For example, for a change request in the external application, extend the runtime **Lcm1ChangeRequest** object in Business Modeler IDE, which maps to the defect object. For Polarion objects, corresponding Teamcenter objects are created by default.

   If you have added a custom domain, extend the **Ldf0TempLinkRuntimeObject** runtime business object to represent the custom semantic types.

2. Update the **RBTypeAttrMapping** element in the semantic mapping file to map the semantic type to the Teamcenter runtime business object type.

3. Update the Relations configuration file with information about the extended semantic types and semantic relations. The **RV1_DARB_UI_configuration_file_name** preference specifies the name of this configuration file.

   For more information about updating the Relations configuration file, see the Active Workspace documentation.

4. **Configure Relations tab for remote system objects**.

Customize Active Workspace page to create links

In Active Workspace, by default, the **Create Link** user interface section is only available in the **Reference** tab of a change. If you want the **Create Link** user interface section to appear in other pages of Active Workspace, you must customize the stylesheets of these objects.

An example of creating Linked Data Framework links for an item revision and displaying the links in the **Attachments** tab is as follows:

1. Create am XML file that contains the relations and the properties to be displayed in the **Create Link** user interface section.

   For example, create an XML file called *coreRemoteLinksSummarySection.xml* as follows:

```
<subRendering>
<section title="Remote Links" titleKey="tc_xrt_RemoteLinks" >
        <objectSet
source="Lrm0MasterRelation.Lis0Link,IMAN_reference.Lis0Link,
        IMAN_specification.Lis0Link,Lcm0AffectsRequirement.Lis0Link,
        Lcm0ImplementsRequirement.Lis0Link,Lcm0TracksRequirement.Lis0Link,
        Lcm0AffectedByDefect.Lis0Link,Lcm0AffectsPlanItem.Lis0Link,
        Lcm0TracksChangeSet.Lis0Link,Lcm0RelatedChangeRequest.Lis0Link,
        Lcm0ImplementedBy.Lis0Link" defaultdisplay="tableDisplay"
        sortby="object_string" sortdirection="ascending">
        <tableDisplay>
            <property name="object_name"/>
            <property name="relation"/>
            <property name="lis0SecondaryRemoteObjType" modifiable="false"/>
        </tableDisplay>
        <thumbnailDisplay/>
        <listDisplay/>
        <command actionKey="newBusinessObjectContextualAction"
        commandId="com.teamcenter.rac.ldf.createNew"
renderingHint="commandbutton"/>
```

```
        <command actionKey="cutAction" commandId="org.eclipse.ui.edit.cut"
        renderingHint="commandbutton">
            <parameter name="localSelection" value="true"/>
        </command>
        </objectSet>
    </section>
    <command actionKey="deleteLinkObject"
    commandId="com.teamcenter.rac.ldf.deleteLink"
renderingHint="commandbutton" />
</subRendering>
```

2. Create a dataset with the same name as the XML file you created in the previous step and associate the XML file to this dataset as a named reference.

   For example, create a dataset named *coreRemoteLinksSummarySection* and associate the *coreRemoteLinksSummarySection.xml* file to this dataset as a named reference.

3. Add the XML file you created in the previous step to the stylesheet of the business object using the **inject** argument.

   For example, to add the **Create Link** user interface in the **Attachments** tab of an Item Revision, do the following:

   a. In My Teamcenter, open the summary stylesheet for the object. For the item revision, the name of the summary stylesheet is *Awp0ItemRevSummary*.

   b. Add the **inject** statement where you want the **Remote Links** user interface to appear.

      For example, to make the **Remote Links** user interface to appear in the **Attachments tab**, update the **Attachments** page element with the following inject statement:

      ```
      <inject type="dataset" src="coreRemoteLinksSummarySection"/>
      ```

      You can see the complete XML code of the **Attachments** page element at the end of this topic:

4. Save the changes to the stylesheet.

5. Restart the Teamcenter pool manager.

6. Clear the browser cache and log on to Active Workspace.

Attachments page element example

```
    </page>
```

```
    <page titleKey="attachments">
        <section titleKey="tc_xrt_Files">
            <objectSet
source="IMAN_specification.Dataset,IMAN_reference.Dataset,

IMAN_manifestation.Dataset,IMAN_Rendering.Dataset,TC_Attaches.Dataset,

IMAN_UG_altrep.Dataset,IMAN_UG_scenario.Dataset,IMAN_Simulation.Dataset"
            defaultdisplay="listDisplay" sortby="object_string"
            sortdirection="ascending">
                <tableDisplay>
                    <property name="object_string"/>
                    <property name="object_type"/>
                    <property name="relation" modifiable="true"/>
                    <property name="release_status_list"/>
                    <property name="date_released"/>
                    <property name="owning_user"/>
                </tableDisplay>
                <thumbnailDisplay/>
                <listDisplay/>
                <command actionKey="newBusinessObjectContextualAction"
                commandId="com.teamcenter.rac.common.AddNew"
                renderingHint="commandbutton"/>
                <command actionKey="pasteAction"
                commandId="com.teamcenter.rac.viewer.pastewithContext"
                renderingHint="commandbutton"/>
                <command actionKey="cutAction"
commandId="org.eclipse.ui.edit.cut"
                renderingHint="commandbutton">
                    <parameter name="localSelection" value="true"/>
                </command>
            </objectSet>
        </section>
        <section titleKey="tc_xrt_Documents">
            <objectSet source="IMAN_specification.DocumentRevision"
            sortdirection="ascending" sortby="object_string"
            defaultdisplay="listDisplay">
                <tableDisplay>
                    <property name="object_string"/>
                    <property name="object_type"/>
                    <property name="release_status_list"/>
                    <property name="date_released"/>
                    <property name="owning_user"/>
                </tableDisplay>
                <thumbnailDisplay/>
                <listDisplay/>
                <command actionKey="newBusinessObjectContextualAction"
                commandId="com.teamcenter.rac.common.AddNew"
                renderingHint="commandbutton"/>
```

```
                <command actionKey="pasteAction"
                commandId="com.teamcenter.rac.viewer.pastewithContext"
                renderingHint="commandbutton"/>
                <command actionKey="cutAction"
commandId="org.eclipse.ui.edit.cut"
                renderingHint="commandbutton">
                    <parameter name="localSelection" value="true"/>
                </command>
            </objectSet>
        </section>
        <inject type="dataset" src="coreRemoteLinksSummarySection"/>
    </page>
```

Querying attributes of remote objects and receiving notifications from remote systems

Querying attributes of remote objects

Suppose you have started a change workflow and the object in the external application is related to that workflows Targets, References, or is itself a target. The workflow can only proceed forward if the object in the external application is in a particular state, for example status is closed. To check the state of the object in the external application, you can use the **LDF-sync-ldf-status** rule handler in your workflow.

Receiving notifications from remote systems

Remote systems can use the Linked Data Framework Notify service to notify Teamcenter about changes. If you want the remote system to advance a workflow, the remote system should call the notify service which then calls the **EPM_trigger_advancer** workflow API. This workflow API can then advance the workflow.

Workflow handlers for performing operations on remote objects

You can use the following workflow handlers to perform operations on objects in remote systems:

- The LDF-create-object workflow handler creates an object in the remote system and relates it to the workflow attachment.
- The LDF-set-task-result-to-property handler reads the specified property from the remote object and updates its task result.
- The LDF-sync-ldf-status handler queries attributes of remote objects to determine the state of the remote object.

Embedded Software Management customizations

The Embedded Software Management functionality uses Linked Data Framework to create links between Embedded Software Management components and artifacts in external applications. You must follow configuration and customization steps for Linked Data Framework to extend the Embedded Software Management linking functionality. This topic give you additional information about customization options:

- By default, only the Software component of Embedded Software Management is mapped to the Software Release artifact in Polarion. You can map other Embedded Software Management components to Polarion components by updating the *lis_esm_properties.xml* file in the *oslc_esm_attr_ds* dataset.

  Example:

  ```
  <TypeMapping targetType="SoftwareRelease">
    <TcTypeMapping srcTcType="Software"/>
    <TcTypeMapping srcTcType="AppSoftware"/>
    <TcTypeMapping srcTcType="Calibration"/>
    <TcTypeMapping srcTcType="ConfigFile"/>
    <TcTypeMapping srcTcType="Ess0License"/>
    <TcTypeMapping srcTcType="PriBootloader"/>
    <TcTypeMapping srcTcType="SecBootloader"/>
  </TypeMapping>
  ```

- Update the **RevisionSummary** and **RevisionSummaryForShowObjectLocation** stylesheets of the Embedded Software Management components so that the **Remote Links** table is enabled.

  Use the following inject statement:

  ```
  <inject type="dataset" src="les1RemoteLinksSummarySection"/>
  ```

- By default, the **Architecture** tab in Active Workspace is enabled only for the Software Architecture component. To enable it for other components, update the **Awb0AvailableFor** constant with the appropriate value of the component.

- To restrict what type of relations you can create between two occurrences, create a preference in the format *primary_type_name_secondary_type_name_allowed_relations*.

  Example:

  To restrict the relation between Software Revision and Processor Revision to **Compatible Link**, create a preference named **Software Revision_Processor Revision_allowed_relations** and in the value section add **Esw0CompatibleLink**.

To find the names of the Embedded Software Management components, load the *esw0esmgmt_schema.xml* template in Business Modeler IDE.

9. Using Linked Data Framework

For more information about how you can create links to external applications using Linked Data Framework, see the Active Workspace Change Management documentation.

You can also upload software binary or software package files from a software artifact to Teamcenter.

10. Using Linked Data Framework REST APIs to create and update Teamcenter resources

Linked Data Framework allows external applications to programmatically create, update, and query Teamcenter resources using REST action. These REST actions are HTTP based interfaces that use*http* methods such as GET, POST, PUT, DELETE, HTTP response codes, content type handling, and resource formats.

You can use a REST client to access the URLs for creating, querying, and updating the supported Teamcenter resources as follows:

- Use the **GET** method and the root services URL to obtain an XML file. In the XML file, search for the Service Provider Catalog URL.
- Using this URL, log on to Teamcenter.
- In the XML file that appears, find the Creation Factory and Query Factory URLs.
- Use the Creation Factory URL to create a Teamcenter resource and the Query Factory URL to query Teamcenter resources.

Use the following methods along with the relevant URLs:

Method name	Description
GET	Queries a resource
POST	Creates a resource
PUT	Updates a resource
DELETE	Deletes a resource

The following header information is required for the REST method invocation:

Method name	Header information
GET	`Accept: application/rdf+xml OR application/json` `OSLC-Core-Version: 2.0`
POST	`Content-Type: application/rdf+xml OR application/json` `OSLC-Core-Version: 2.0`

PUT	`Accept: application/rdf+xml`
	`Content-Type: application/rdf+xml OR application/json`
	`OSLC-Core-Version: 2.0`
	`If-Match:`

DELETE	`Accept: application/rdf+xml`
	`OSLC-Core-Version: 2.0`
	`If-Match:`

Where:

`application/rdf+xml`	Providers must respond with RDF/XML representation.
`application/json`	Providers must respond with JSON representation.
`If-Match`	Contains the eTag that determines the state of the resource.
`Content-Type`	Specifies the input content type to all OSLC providers.
`OSLC-Core-Version`	Specifies the supported version of the OSLC specification.

You must pass the RDF/XML or JSON as input when you use the REST APIs.

GET action

Description

Retrieves all information about a specific Teamcenter resource by making an HTTP GET request to the Query Factory URL.

Headers	`Accept: application/rdf+xml OR application/json` `OSLC-Core-Version: 2.0`
Resource URI	URL of the Teamcenter resource that contains its UID.
Method	**GET**

Response information

Response headers

Status code 200: Successful. Response body contains all the information of the resource.

Status code 500: Resource does not exist. Response body will show error message from Teamcenter. For example, `The specified tag has been deleted, cannot find requested object`

Response body

JSON or RDF+XML.

Example

Method: GET

Resource URI: http://pni6w2071.net.plm.eds.com:7001/lis/oslc/cm/BO?uid=AaYhvJClYj1eHD

Headers:

```
OSLC-Core-Version: 2.0
Accept: application/json
```

Response header

```
Status Code: 200 OK
Content-Type: application/json;charset=UTF-8
Date: Tue, 18 Apr 2017 17:43:55 GMT
Etag: AmXhvJClYj1eHD-AaYhvJClYj1eHD
OSLC-Core-Version: 2.0
Transfer-Encoding: chunked
X-Powered-By: Servlet/2.5 JSP/2.1
```

Response body

```
{
  "dcterms:contributor": "18-Apr-2017 14:31",
  "dcterms:created": "18-Apr-2017 14:31",
  "dcterms:creator": "tcdba",
  "dcterms:description": "ECR-000035",
  "dcterms:identifier": "ECR-000035",
  "dcterms:modified": "18-Apr-2017 14:31",
  "dcterms:subject": "ECR-000035",
  "dcterms:title": "ECR-000035",
  "etag": "AmXhvJClYj1eHD-AaYhvJClYj1eHD",
  "oslc:instanceShape": "http:\/\/pni6w2071.net.plm.eds.com:7001\/
lis\/oslc\/
  resourceShape\/ChangeRequest",
```

```
  "oslc:shortTitle": "ECR-000035",
  "oslc_cm:affectedByDefect":
  [
    {
      "dcterms:title": "http:\/\/pnv6s282.net.plm.eds.com:82\/
polarion\/oslc\/
      services\/projects\/elibrary\/workitems\/EL-204",
      "rdf:resource": "http:\/\/pnv6s282.net.plm.eds.com:82\/
polarion\/oslc\/
      services\/projects\/elibrary\/workitems\/EL-204",
      "rdf:type":
      [
      ]
    }
  ],
  "oslc_cm:closeDate": "18-Apr-2017 14:31",
  "prefixes":
  {
    "dcterms": "http:\/\/purl.org\/dc\/terms\/",
    "oslc": "http:\/\/open-services.net\/ns\/core#",
    "oslc_cm": "http:\/\/open-services.net\/ns\/cm#",
    "rdf": "http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-ns#",
    "rdfs": "http:\/\/www.w3.org\/2000\/01\/rdf-schema#",
    "tc": "http:\/\/www.plm.automation.siemens.com\/ldf#"
  },
  "rdf:about": "http:\/\/pni6w2071.net.plm.eds.com:7001\/lis\/
oslc\/cm\/
  BO?uid=AaYhvJClYj1eHD",
  "rdf:type":
  [
    {"rdf:resource": "http:\/\/open-services.net\/ns\/
cm#ChangeRequest"}
  ]
}
```


POST action

Description

Creates a resource in Teamcenter resource by making an HTTP POST request to the Creation Factory URL.

Headers

```
OSLC-Core-Version:2.0
Content-Type: application/json
Content Body: It contain the resource which need to be
created.
```

Resource URI URL of the Creation Factory.

Method **POST**

Response information

Response headers

Successful: Status code 200 with etag and location that contains the Reource URL.

Content body invalid: Status code 500. Response body will show error message from Teamcenter. For example, `The Media Reader has failed to parse the input JSON envelop/document. Check the Teamcenter server syslog for more details.`

Response body

JSON or RDF+XML.

Example

Method: POST

Resource URI: http://pni6w2071.net.plm.eds.com:7001/lis/oslc/cm/createfactory

Headers:

```
OSLC-Core-Version:2.0
Content-Type: application/json
```

Content body

```
{
  "dcterms:description": "Test_factory_create1",
  "dcterms:title": "Test_factory_create1",
  "dcterms:type": "ChangeRequest",
  "prefixes":
  {
    "dcterms": "http:\/\/purl.org\/dc\/terms\/",
    "oslc": "http:\/\/open-services.net\/ns\/core#",
    "oslc_cm": "http:\/\/open-services.net\/ns\/cm#",
    "rdf": "http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-ns#",
    "rdfs": "http:\/\/www.w3.org\/2000\/01\/rdf-schema#",
    "tc": "http:\/\/www.plm.automation.siemens.com\/ldf#"
  },
  "rdf:about": "http:\/\/pni6w2071.net.plm.eds.com:7001\/lis\/
oslc\/cm\/
  createfactory",
  "rdf:type":
  [
```

```
    {"rdf:resource": "http:\/\/open-services.net\/ns\/
cm#ChangeRequest"}
  ]
}
```

Response header

```
Status Code: 200 OK
Content-Length: 0
Content-Type: application/json; charset=UTF-8
Date: Wed, 19 Apr 2017 08:18:16 GMT
Etag: QtWhvlNkYj1eHD-QhYhvlNkYj1eHD
Location: http://pni6w2071.net.plm.eds.com:7001/lis/oslc/cm/BO?
uid=QhYhvlNkYj1eHD
OSLC-Core-Version: 2.0
X-Powered-By: Servlet/2.5 JSP/2.1
```

PUT action

Description

Updates content of a specific resource in Teamcenter by making an HTTP PUT request to the resource URL.

Headers	`Content-Type: application/rdf+xml, application/json`
	`OSLC-Core-Version: 2.0`
	
	If-Match = Eta, make sure that changes are applied to the state of the resource that has been fetched from the server.

Resource URI	Reource URL that contains the Teamcenter resource UID.
Method	**PUT**
Content body	Contains the resource that needs to be updated. Before using PUT, use the GET method on the Teamcenter resource, which will return the response body and Etag in response header. Use the Etag in the IF-match of the PUT request.

Response information

Response headers

Successful: Status code 200 with Etag containing the state of the resource.

Note:

Avoid overwriting changes done by someone else. A resource may have changed in the time between fetching it, doing modifications and sending it back. Clients must not blindly overwrite work item contents but ensure that the changes are applied to the state that they fetched from the server. This is done by including the Etag received on load in the If-Match header when updating the work item back to the server using PUT. If the state of the server is still the same, the update will succeed with a 200OK status. If the work item was modified, it will fail with a 412 Precondition Failed status.

Response body

JSON or RDF+XML.

Example

Method: PUT

Resource URI: http://pni6w2071.net.plm.eds.com:7001/lis/oslc/cm/BO?uid=AaYhvJClYj1eHD

Headers:

```
Content-Type: application/rdf+xml
OSLC-Core-Version: 2.0
If-Match = QxWhfFXKYjj4JA-QlXhfFXKYjj4JA
```

Content body

```
<rdf:RDF
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:oslc="http://open-services.net/ns/core#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:j.0=http://open-services.net/ns/cm#
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <j.0:ChangeRequest    rdf:about="http://
pni6w2071.net.plm.eds.com:7001/lis/oslc/
 cm/BO?uid=AaYhvJClYj1eHD">
<dcterms:title>Test_factory_create1_update</dcterms:title>
<dcterms:description>Test_factory_create1_update</dcterms:description>
</j.0:ChangeRequest>
</rdf:RDF>
```

Response header

```
Status Code: 200 OK
```

```
Content-Length: 0
Content-Type: application/json; charset=UTF-8
Date: Tue, 18 Apr 2017 17:56:23 GMT
Etag: gdRhvlMUYj1eHD-AaYhvJClYj1eHD
OSLC-Core-Version: 2.0
X-Powered-By: Servlet/2.5 JSP/2.1
```

DELETE action

Description

Deletes a resource in Teamcenter by making an HTTP DELETE request to the resource URL.

Headers

```
Accept : application/rdf+xml
OSLC-Core-Version :2.0
Content-Type : application/rdf+xml OR application/json
```

Resource URI Reource URL that contains the Teamcenter resource UID.

Method **DELETE**

Response information

Response headers

Successful deletion: Status code 200

Response body

JSON or RDF+XML.

Example

Method: DELETE

Resource URI: http://pni6w2071.net.plm.eds.com:7001/lis/oslc/cm/BO?uid=AaYhvJClYj1eHD

Headers:

```
Accept : application/rdf+xml
OSLC-Core-Version :2.0
```

Response header

```
Status Code: 200 OK
Content-Length: 0
Content-Type: application/json; charset=UTF-8
```

```
Date: Wed, 19 Apr 2017 06:21:48 GMT
OSLC-Core-Version: 2.0
X-Powered-By: Servlet/2.5 JSP/2.1
```

QUERY action

Description

Queries resources in Teamcenter by making HTTP GET requests to the query URL. Optionally, this can be filtered with a where clauses.

Headers

```
Accept: application/rdf+xml OR application/json
OSLC-Core-Version: 2.0
```

Resource URI URL of the Query Factory.

Method **GET**

Query Parameters

Parameter	Type	Description
oslc.where	string	Name of the oslc properties that must be queried. Supported properties include: • dcterms:type • dcterms:description • dcterms:title • dcterms:subject • dcterms:identifier • oslc:shortTitle

Response information

Response headers

Successful: Status code 200: List of resources.

Status code 404: Not Found: The Registry Helper has failed to process the request, because no valid Provider Catalogue exists.

Status code 500: Internal Server Error: The specified type does not exist.

Response body

JSON or RDF+XML.

Example 1

Method: GET

Query URL: http://pni6w11193:7001/lis/oslc/core/queryfactory

Headers:

```
OSLC-Core-Version: 2.0
Accept: application/json
```

Response header

```
Status Code: 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 06 Jun 2018 07:18:40 GMT
OSLC-Core-Version: 2.0
Transfer-Encoding: chunked
X-Powered-By: Servlet/3.0 JSP/2.2
```

Response body

```
{
  "oslc:responseInfo":
  [
    {
      "oslc:totalCount": 3,
      "rdf:type":
      [
        {"rdf:resource": "http:\/\/open-services.net\/ns\/
core#ResourceInfo"}
      ],
      "rdfs:member":
      [
        {"rdf:resource": "http:\/\/pni6w11193.net.plm.eds.com:7001\/
lis\/oslc\/
                          core\/BO?uid=A3Yp8rIQ6jRptC"},
        {"rdf:resource": "http:\/\/pni6w11193.net.plm.eds.com:7001\/
lis\/oslc\/
                          core\/BO?uid=A5QpbDN06jRptC"},
        {"rdf:resource": "http:\/\/pni6w11193.net.plm.eds.com:7001\/
lis\/oslc\/
                          core\/BO?uid=A5bpsI2E6jRptC"},
            ]
    }
  ],
  "prefixes":
```

```
    {
      "dcterms": "http:\/\/purl.org\/dc\/terms\/",
      "oslc": "http:\/\/open-services.net\/ns\/core#",
      "rdf": "http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-ns#",
      "rdfs": "http:\/\/www.w3.org\/2000\/01\/rdf-schema#",
      "tc": "http:\/\/www.plm.automation.siemens.com\/ldf#"
    },
    "rdf:about": "http:\/\/pni6w11193.net.plm.eds.com:7001\/lis\/oslc\/
                  core\/queryfactory",
    "rdf:type":
    [
    ]
  }
```

Example 2:

Method: GET

Query URL with where clause: http://pni6w11193:7001/lis/oslc/core/queryfactory?oslc.where=dcterms:type=Item

Headers:

```
OSLC-Core-Version: 2.0
Accept: application/json
```

Response header

```
Status Code: 200 OK
Content-Type: application/json;charset=UTF-8
Date: Wed, 06 Jun 2018 07:26:47 GMT
OSLC-Core-Version: 2.0
Transfer-Encoding: chunked
X-Powered-By: Servlet/3.0 JSP/2.2
```

Response body

```
{
  "oslc:responseInfo":
  [
    {
      "oslc:totalCount": 3,
      "rdf:type":
      [
        {"rdf:resource": "http:\/\/open-services.net\/ns\/
core#ResourceInfo"}
```

```
      ],
      "rdfs:member":
      [
        {"rdf:resource": "http:\/\/pni6w11193.net.plm.eds.com:7001\/
lis\/oslc\/
                          core\/BO?uid=A3Yp8rIQ6jRptC"},
        {"rdf:resource": "http:\/\/pni6w11193.net.plm.eds.com:7001\/
lis\/oslc\/
                          core\/BO?uid=A5QpbDN06jRptC"},
        {"rdf:resource": "http:\/\/pni6w11193.net.plm.eds.com:7001\/
lis\/oslc\/
                          core\/BO?uid=A5bpsI2E6jRptC"},
            ]
    }
  ],
  "prefixes":
  {
    "dcterms": "http:\/\/purl.org\/dc\/terms\/",
    "oslc": "http:\/\/open-services.net\/ns\/core#",
    "rdf": "http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-ns#",
    "rdfs": "http:\/\/www.w3.org\/2000\/01\/rdf-schema#",
    "tc": "http:\/\/www.plm.automation.siemens.com\/ldf#"
  },
  "rdf:about": "http:\/\/pni6w11193.net.plm.eds.com:7001\/lis\/oslc\/
               core\/queryfactory\/?&oslc.where=dcterms:type=Item",
  "rdf:type":
  [
  ]
}
```

NOTIFY action

Description

Notifies Teamcenter about a change associated with the linked object, for example, a remote system such as Polarion sends URLs of Teamcenter objects associated with the Polarion work items whose status is changed.

Headers

```
Accept: application/xml
Authorization: OAuth realm="Teamcenter",
oauth_token="bdce003d-5199-48d5-85fe-4c71f3e64cbd",
oauth_consumer_key="f013abe9-7258-42da-98da-5c1373c46546",
oauth_signature_method="HMAC-SHA1",
oauth_timestamp="1526898982",
oauth_nonce="4589207281453928", oauth_version="1.0",
oauth_signature="vCUmLISgIgvxXEoR2Msi%2FQxmXGg%3D"
Content-Type: application/json
OSLC-Core-Version: 2.0
```

```
User-Agent: Wink Client v1.1.2
Content-Length: 1708
Host: pni6w11193.net.plm.eds.com:7001
Connection: Keep-Alive
Cookie: JSESSIONID=lQT8bCpGfgBjS2GW8LG8hpTB57kvr1YSW
                    cRqvPfsPgjZ2PLcSP2t!201170707
Cookie2: $Version=1
```

URI http://hostname/lis/oslc/custom/notify

Method **POST**

Content Body

```
{
."dcterms:created": "2018-05-21T16:04:33.073+05:30",
."dcterms:description": "<span style=\"font-size: 10pt;
  line-height: 1.5;\">Nitin_CR_21May_003<\/span>",
."dcterms:identifier": "EL-333",
."dcterms:modified": "2018-05-21T16:05:45.240+05:30",
."dcterms:subject": [
.],
."dcterms:title": "Nitin_CR_21May_003",
."dcterms:type": "Change Request",
."oslc:serviceProvider": {
.."rdf:resource": "http:\/\/pnv6s435:82\/polarion\/oslc\/
  services\/projects\/elibrary"
.},
."oslc:shortTitle": "EL-333",
."oslc_cm:closeDate": "2018-05-21T16:06:21.550+05:30",
."oslc_cm:closed": false,
."oslc_cm:fixed": true,
."oslc_cm:inprogress": false,
."oslc_cm:relatedChangeRequest": [
..{
..."dcterms:title": "Nitin_CR_21May_002_TC4",
..."rdf:resource": "http:\/\/
pni6w11193.net.plm.eds.com:7001\/
   lis\/oslc\/cm\/BO?uid=wldtg4FJ6jRptC"
..}
.],
."oslc_cm:status": "Approved",
."oslc_cm:verified": false,
."pol1:categories": [
.],
."pol1:previousStatus": "Reviewed",
."pol1:priority": "Medium",
."pol1:resolution": "Valid",
."pol1:severity": "Should Have",
."pol1:status": "Approved",
."pol1:type": "Change Request",
```

```
."prefixes": {
.."dcterms": "http:\/\/purl.org\/dc\/terms\/",
.."ldf": "http:\/\/www.plm.automation.siemens.com\/ldf#",
.."oslc": "http:\/\/open-services.net\/ns\/core#",
.."oslc_cm": "http:\/\/open-services.net\/ns\/cm#",
.."oslc_rm": "http:\/\/open-services.net\/ns\/rm#",
.."pol1": "http:\/\/polarion.plm.automation.siemens.com\/
oslc#",
.."siemens_esm": "http:\/\/www.plm.automation.siemens.com\/
   ldf\/esm#"
.},
."rdf:about": "http:\/\/pnv6s435:82\/polarion\/oslc\/
  services\/projects\/elibrary\/workitems\/EL-333",
."rdf:type": [
..{
..."rdf:resource": "http:\/\/open-services.net\/ns\/
   cm#ChangeRequest"
..}
.]
}
```

Response Header

```
HTTP/1.1 201 Created
Date: Mon, 21 May 2018 10:36:22 GMT
Transfer-Encoding: chunked
Content-Type: application/xml; charset=UTF-8
OSLC-Core-Version: 2.0
X-Powered-By: Servlet/3.0 JSP/2.2
```

11. Tips, notes, and warnings

- If you receive a CSRF attack warning in your browser when connecting to Teamcenter, clear the browser cache and log on again.
- When viewing Teamcenter information from the external application, the **Show More** and **Show Less** links show the same content because they point to the same URL.
- If one user from the external application maps to more than one Teamcenter user, Teamcenter will not be able to identify the correct user. This is because Teamcenter uses the same *Oauth* user authentication that the external application uses to map its users.
- Linked Data Framework and Active Workspace must be deployed to the same domain of the web app server. An inconsistency results in issues while rendering the delegated UI dialogs. This is due to the Same Origin Policy that the browser implements.
- Although change management and core service catalogs are available by default, external applications cannot identify the core service catalogs because the external applications do not consume any Teamcenter namespaces.
- When using the **PUT** and **DELETE** REST operations, you must specify the Accept Header as `application/rdf+xml` or `application/json`, even though these operations do not return any output.
- If you are unable to see specific types of dataset when you upload files from either Polarion or Teamcenter using **Remote File Upload** functionality, then update the preference **TYPE_DISPLAY_RULES_list_types_of_subclasses** with the value **Dataset**.
- Refer to the Active Workspace Customization Guide for information about customizing the delegated UIs.
- Using the root services URL, you can discover other URLs that Linked Data Framework uses. The following table lists URLs for the Change Management service, which is available by default:

Name	Linked Data Framework URL
Root services	http://<hostname>:portno/lis/oslc/rootservices
CM Catalog	http://<hostname>:portno/lis/oslc/cm/catalogs
CM Service Provider	http://<hostname>:portno/lis/oslc/cm/providers
CM Creation Factory	http://<hostname>:portno/lis/oslc/cm/createFactory
CM Query Capability	http://<hostname>:portno/lis/oslc/cm/queryFactory?... where **...** represents the query where clause. For example:

Name	Linked Data Framework URL
	?oslc.where=dcterms:identifier="<ItemID>" ?oslc.where=dcterms:type="<Type>" ?oslc.where=dcterms:title="<ObjectName>" ?oslc.where=dcterms:type="<Type>"&dcterms:title="<Object Name>" ?oslc.prefix=tc=<http://www.plm.automation.siemens.com/lis>&oslc.where=tc:item_id="<Item ID>" ?oslc.prefix=tc=<http://www.plm.automation.siemens.com/lis>&oslc.where=dcterms:type="<Type>"&tc:item_id="<Item ID>"
CM Creation Dialog	http://<hostname>:portno/lis/oslc/cm/creator
CM Selection Dialog	http://<hostname>:portno/lis/oslc/cm/selector
Resource	http://<hostname>:portno/lis/oslc/cm/largepreview?uid=<UID>
Resource Small Preview	http://<hostname>:portno/lis/oslc/cm/smallpreview?uid=<UID>
Resource Large Preview	http://<hostname>:portno/lis/oslc/cm/largepreview?uid=<UID>
Resource Editor	http://<hostname>:portno/lis/oslc/cm/editor?uid=<UID>
CM BO Preview Factory	http://<hostname>:portno/lis/oslc/cm/BO?uid=<UID>
CM BO Editor Factory	http://<hostname>:portno/lis/oslc/cm/BO?uid=<UID>
CM BO Delete Factory	http://<hostname>:portno/lis/oslc/cm/BO?uid=<UID>
Listing resource data model of given BO Type	http://<hostname>:port/oslc/cm/resourceShapes/<BOType>