TEAMCENTER

Security Services Configuration

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**

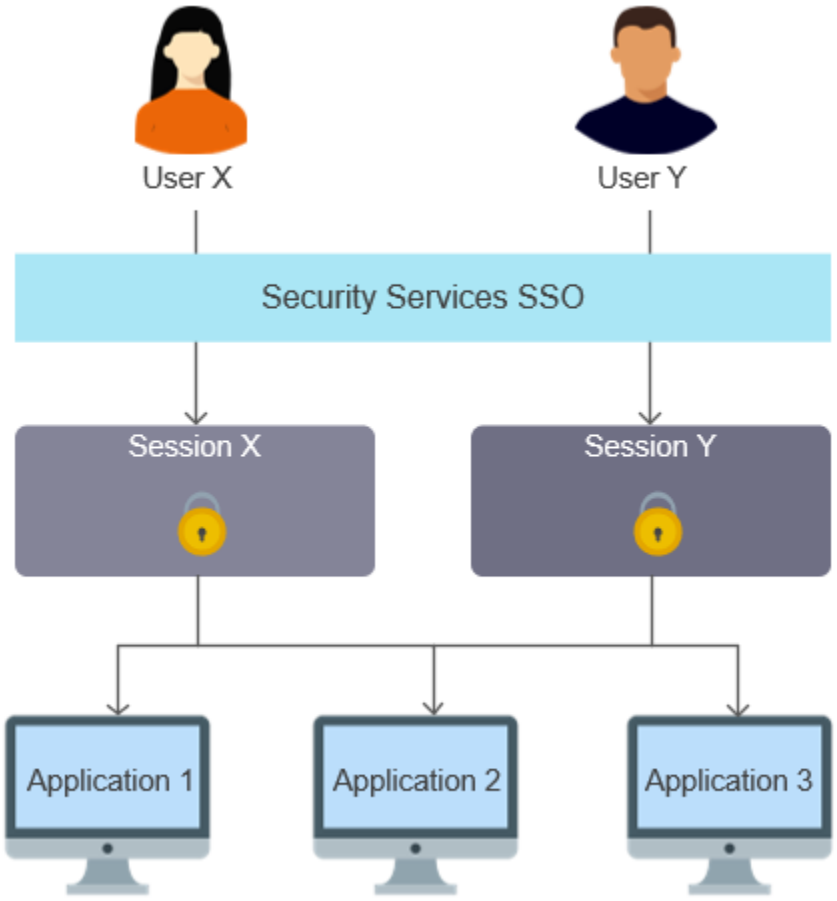
Contents

Reverse proxy support for external identity providers, WebSEAL, SiteMinder, File Management System (FMS), and two-way SSL

1. Getting started with Security Services

What is Security Services?

Security Services is a software solution that integrates with a site's single sign-on solution to provide single sign-on connections to multiple Siemens Digital Industries Software products. When a user initially signs on to a Teamcenter application, a Security Services logon authentication establishes a session that eliminates the need for multiple authentication challenges for opening additional Teamcenter applications as long as the session remains in effect.



A Security Services session expires if there is no interaction with Security Services over a configured span of time, even if the user is actively using one or more Teamcenter applications. However, Security Services session expiration does not affect a user's current sessions with Teamcenter applications; it simply means they receive a logon challenge if they start another Teamcenter application.

System requirements

Security Services requires a web application server, LDAP v3-compliant identity provider, and Java runtime extension (JRE). For information about versions of operating systems, third-party software, and Teamcenter software that are certified for your platform, see the *Hardware and Software Certifications* knowledge base article on Support Center.

Web application servers

The Security Services web components (Login Service and Identity Service) can be deployed on the following web application servers:

- IBM WebSphere application server
- Tomcat application server
- JBoss application server

Web browsers

Security Services is supported by the following web browsers:

- Microsoft Edge
- Mozilla Firefox

> Caution:
>
> Mozilla Firefox 30+ ignores the **autocomplete=off** attribute for password fields.

- Google Chrome

LDAP directories

Security Services can use any LDAP v3-compliant identity provider.

Java environment

In addition to installing one of the supported web browsers, you also must install the Java Runtime Environment (JRE) Java Plug-in on each client machine.

Teamcenter Security Services product support

> Note:
>
> You can install Security Services as part of a Teamcenter installation, or you can install it separately to use with any of the following products that support Security Services:

- Teamcenter

- Teamcenter Enterprise
- Teamcenter engineering process management (Engineering Process Management)
- Teamcenter portfolio, program and project management (Portfolio, Program and Project Management)
- Teamcenter systems engineering and requirements management (Systems Engineering and Requirements Management)
- Teamcenter community collaboration (Community Collaboration)
- Teamcenter lifecycle visualization (Lifecycle Visualization)

How Security Services works

Security Services consists of two services and a session agent:

Login Service

The Login Service is the Security Services component that interacts with Teamcenter client applications. On behalf of those clients, it challenges the user with a logon prompt and collects the supplied user ID and password. Following authentication of those credentials, it returns a Teamcenter Security Services application token to the Teamcenter client application. The Login Service is also the repository for active Security Services sessions. That is, it holds the state information essential to the single sign-on capability of Security Services. In the Web Application Manager, this service appears as **Teamcenter Security Services Login Service Web Application**.

The Login Service is a web application that controls logon challenges. Teamcenter web applications interact with the Login Service through a web redirection protocol. Teamcenter rich clients interact with the Login Service through the Security Services client library and the Security Services Session Agent. The Login Service interacts with the Identity Service to authenticate users and to generate Security Services tokens.

Identity Service

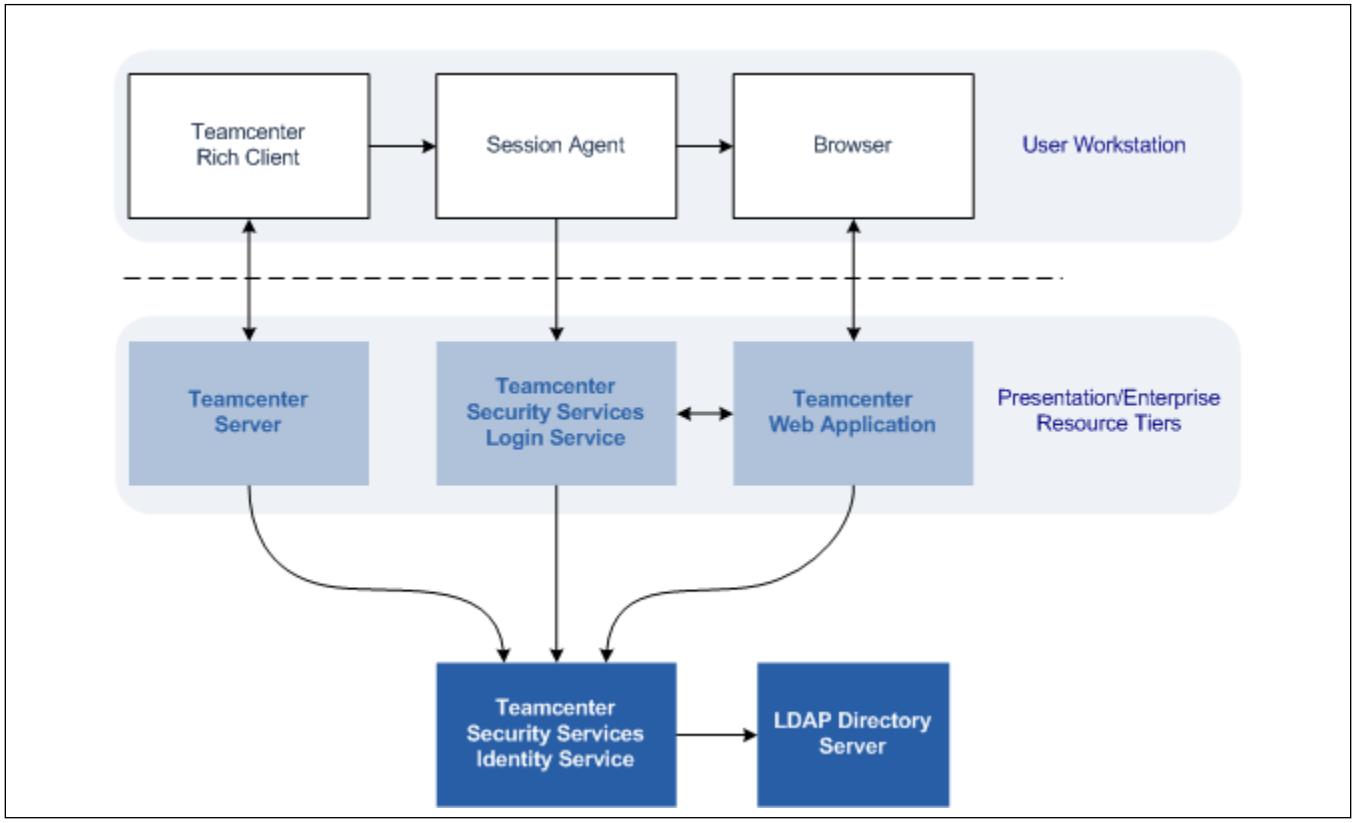
The Identity Service authenticates user credentials, meaning it verifies a user ID and password against an underlying identity provider. That provider can be an LDAP directory or a customer-provided facility. The Identity Service also interacts with Teamcenter server applications to validate Teamcenter Security Services application tokens. In a typical single sign-on deployment, user credentials are collected and submitted by the Login Service. However, the Identity Service is independent of the Login Service. Other applications can furnish user credentials directly for authentication, using the Identity Service simply as an interface to an identity provider. In the Web Application Manager, this service appears as **Teamcenter Security Services Identity Service Web Application**. The Identity Service is configured for a default LDAP identity provider implementation, but it can be configured to work with a variety of implementations.

The Identity Service is a web application. Teamcenter servers and web applications interact with the Identity Service interfaces to an identity provider (for example, an LDAP repository) to authenticate users and to determine the user's alias for a specific Teamcenter application.

Security Services Session Agent

The Security Services session agent must be installed on client machines to support the Security Services single sign-on experience.

The following figure illustrates the interaction between these components.



Security Services components

To enable LDAP integration, deploy both the Security Services Login Service and Identity Service with Teamcenter and NX clients. Security Services normally requires each Teamcenter user to have a single Teamcenter user identifier and associated password established in the underlying identify provider, aside from **special cases** where an SSO gateway is used without aliasing user names. If single sign-on capability is not needed, the Identity Service can be deployed without the Login Service. This is known as *authentication-only* mode.

You can deploy both the Login Service and Identity Service in a clustered web server environment. This enables load balancing and failover. Some Teamcenter applications can be configured to use the Identity Service directly for authentication.

When a user launches a first Security Services-enabled Teamcenter application, that application invokes the Login Service. Because there is no active Security Services session, the Login Service challenges

the user with a logon window. In that window, the user should enter a Teamcenter user name and password, and, if desired, a locale from the displayed list. If Security Services is configured appropriately, the user can enter a user ID in *domain\username* format, for example, **acme.com\john**. If domain is omitted, the domain is assumed to be the default base DN configured in the Identity Service.

The Login Service authenticates the provided user ID and displays the Security Services Session Agent window in the browser. It is through this window that subsequent Teamcenter applications join the Security Services session. The window should remain open unless the user is ready to end the session; however, the user can minimze the window.

Security Services draws a distinction between an authenticating user ID and a Teamcenter user ID. The authenticating user ID is the one provided by the user to log on to Security Services. That user ID may be authenticated by an LDAP lookup through the Identity Service or through an authenticating gateway deployed in front of the Login Service. That user ID is also a Teamcenter user ID, unless a mapping is defined between the authenticating user ID and a Teamcenter user ID. If so, that mapping is performed by an LDAP, which must be configured within the Identity Service.

Legacy Security Services

Prior to Security Services 11.3, the Login Service loaded applets for rich client on the client machine. These applets were written in Java and used Java Object Signing.

Using applets for these purposes avoided any need to install executables on client machines to support Security Services.

> Note:
>
> The following applets are deprecated as of Security Services 11.3.

- Teamcenter Security Services session agent applet

  This applet, which lasts the lifetime of the Security Services session, represents the session to the user. It contains no Security Services session information directly, but it connects the Teamcenter client to the Login Service. This applet appears as a small browser on the your desktop, which you can minimize without affecting the Security Services session. Clicking the **Logout** button in this dialog box ends the Security Services session. You must enter your logon credentials to launch another client.

- Teamcenter Security Services session detector applet

- Teamcenter Security Services status reporter applet

  This applet assists the internal establishment of a Security Services session with the rich client.

The Java applets use Javascript Object Signing. With Javascript Object Signing, the Security Services applets do not directly read any certificate store. On the clients, access to certificate stores is handled by the Java browser plug-in run-time environment (JRE).

Starting at Security Services 11.3, the Login Service no longer loads applets on the client computer. This is true regardless of the version of a rich client. However, for backward compatibility, Login Services from Security Services versions prior to 11.3 will continue to load applets for a Security Services 11.3 rich client.

Session management

Logon credentials

Users log on to Security Services through a browser. The user's logon credentials are transmitted from the workstation using HTTP or HTTPS (selected during Security Services configuration) to the Login Service, and subsequently to the Identity Service and the LDAP repository. After the user's credentials are authenticated, the credentials are not accessed or transmitted as users launch subsequent Teamcenter applications that join their Security Services session.

Application IDs

An application is represented within Security Services as a unique text string known as an application ID. An administrator must define an application ID for each Teamcenter application in the Security Services domain. You use these application IDs when completing the Application Registry table.

Security Services application tokens

Teamcenter Security Services application tokens represent authenticated users within Security Services. These tokens essentially replace the user's original credentials for Teamcenter applications joining the user's Security Services session. Each time a user launches a Teamcenter application, Security Services creates a token and delivers it to the application's client. The client forwards this token to the application's underlying server, which then submits the token back to the Security Services Identity Server for validation.

Using Teamcenter Security Services application tokens avoids the obvious security issue that would be present if the user's credentials were passed among Teamcenter clients and servers. Teamcenter Security Services tokens security is achieved using the following mechanisms:

- Tokens are never stored on the client workstation. They are passed from the browser to rich clients, but they do not persist on the workstation as cookies, in files, or any other form.
- Tokens are specific to each installed Teamcenter product. Each token incorporates an application ID indicating the installed Teamcenter product to which it applies. This ID is established for each Teamcenter product during installation. No other application can submit a token unless its ID matches the token's ID.
- Tokens have a configurable (typically short) lifetime.
- Tokens are encrypted. Encryption keys are maintained exclusively within the Login and Identity Services.

Asynchronous Security Services application tokens

Asynchronous Security Services application tokens are used only in conjunction with mediating applications. Mediating applications, such as Teamcenter Integration Framework and Global Services, can assume the role of a Security Services session agent and submit special Security Services logon requests, which include the usual target application ID plus an additional pseudo application ID.

All logon requests to Security Services return a Teamcenter Security Services application token built for the target application, but the token Teamcenter Security Services returns to the mediating application has a special structure: It contains an inner token, which is intended for the target application and contains an alias user ID. Security Services looks up that alias user ID for the target application in the LDAP identity provider, keying off the actual user ID and pseudo application ID. That token is returned to the mediating application wrapped in an outer token that is separately encrypted. The mediating application decrypts the outer token and extracts and forwards the inner token to the target application, which subsequently validates that token back with Security Services, conferring the alias user ID's level of authentication for the real user on the target application.

Configuring asynchronous Security Services application tokens involves:

- Setting a mediator password.
- Configuring pseudo application IDs in the Application Registry table.
- Adding LDAP entries for the pseudo application IDs.

For more information about configuring asynchronous Security Services application tokens, see the *Dispatcher — Deployment and Administration*.

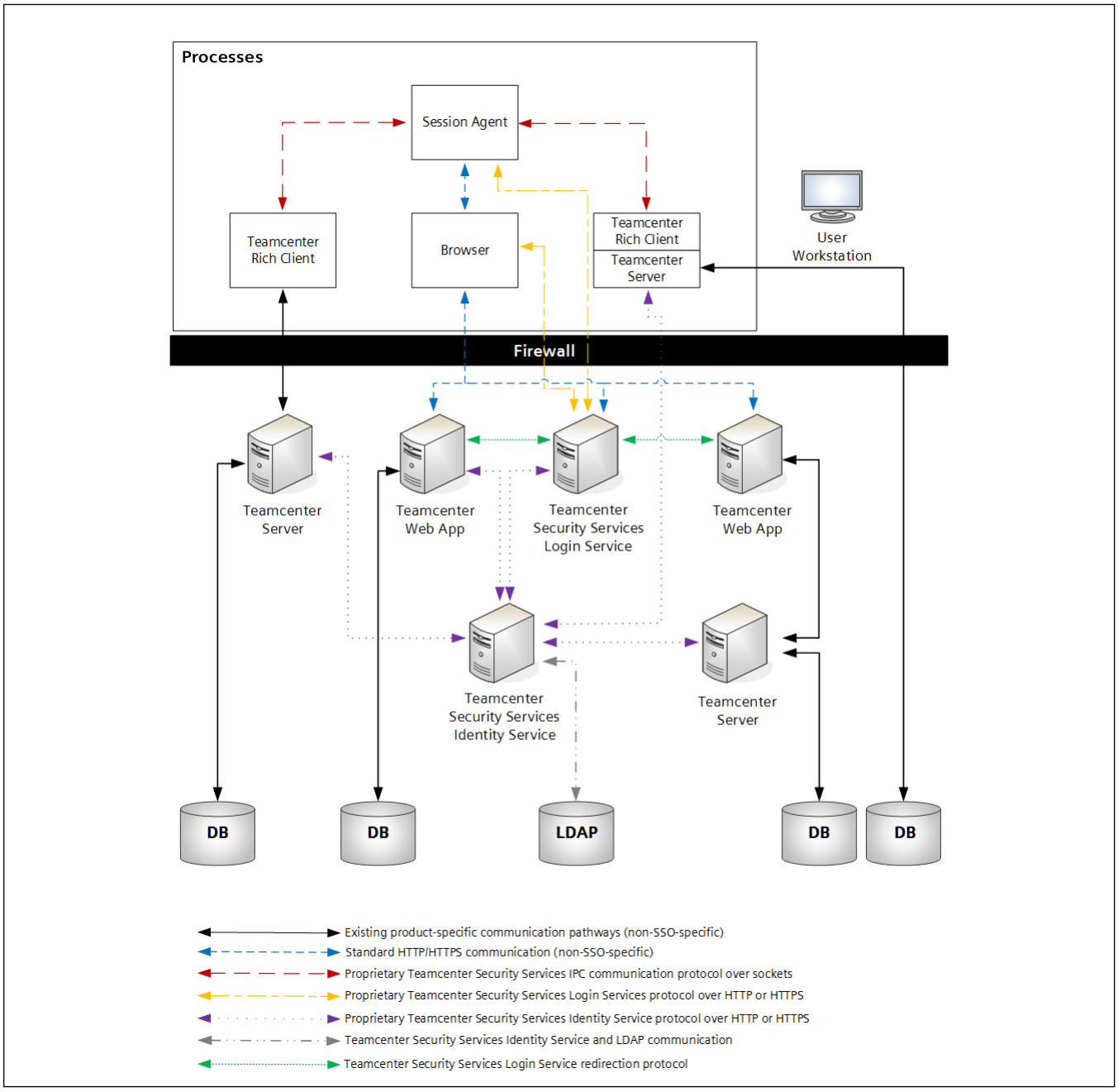
Teamcenter Security Services session lifetime

Once authenticated, a Teamcenter Security Services session exists until the user deliberately ends the session or the session times out. Any Teamcenter application launched by the user after the initial logon joins the user’s existing session without presenting another logon window to the user. Therefore, security of the session assumes the user’s workstation is secure, for example, that it is not left unattended.

Security Services includes a configuration setting that defines the maximum Teamcenter Security Services session idle time. If a Teamcenter Security Services session is found to be idle for longer than the specified time, the session is automatically closed. If a new Teamcenter application is launched, the user receives a new logon window.

Security Services communication channels

Security Services transmits SSO credentials (logon credentials and Teamcenter Security Services tokens) over a number of communication channels. Some of the pathways and protocols employ proprietary Security Services protocols. The following figure shows the various communication channels.



Security Services communication flow

Existing communication links in Teamcenter products

Many of the communication links over which Teamcenter Security Services tokens are transmitted are specific to individual Teamcenter products. Several typical deployments of both web-based and non-web-based Teamcenter applications appear in the diagram. Teamcenter Security Services tokens are transmitted from clients (rich clients or web applications) to servers as part of the protocol explained in the previous section.

For information regarding the security of data transmitted between those clients and servers, see your Teamcenter product documentation.

Communication links between browsers and Teamcenter web applications

Some Teamcenter applications offer web tier interfaces, including the Login Service. This browser communication is not Security Services specific, but user logon credentials are transmitted using web protocols. Siemens Digital Industries Software recommends deploying these and other web applications behind a firewall and on web servers that employ secure communication protocols (for example, SSL).

Security Services session agent proprietary protocol over sockets for rich clients

Part of the Security Services solution involves communication between rich clients and the Security Services Session Agent. This is a proprietary protocol based on sockets. The communication over this channel includes transmission of Security Services tokens, and its security is based on the following mechanisms:

- It uses an unpublished protocol specific to Security Services.
- It is confined to a single user workstation.
- It employs local port numbers.
- It incorporates the application's unique tag defined during installation.
- It leverages OS security mechanisms such that only the OS user can initiate the protocol.

Login Service proprietary protocol over HTTP for rich clients

In parallel with the normal browser communication, the Security Services Session Agent communicates with the Login Service using a proprietary protocol over HTTP. This communication channel is involved in the transmission of Security Services tokens to rich clients. The security of this channel is based on the following mechanisms:

- It uses an unpublished protocol specific to Security Services.
- It is built on top of a secure underlying communication protocol. That is, it is assumed that the Login Service is on a web server employing secure communications, such as SSL.

Identity Service proprietary protocol over HTTP

Teamcenter applications interact with the Identity Service to validate Security Services tokens by using a proprietary protocol over HTTP. Generally, this communication occurs behind security boundaries. Some Teamcenter deployments involve the installation of servers outside the security boundary. The following mechanisms ensure the integrity and security of this communication:

- It uses an unpublished protocol specific to Security Services.
- Callers of the Identity Service must provide the unique application tag (defined during installation) in each transmission.
- The Identity Service can be deployed on a web server configured to use SSL.

Communicating with LDAP servers

Security Services leverages LDAP connections for its communication with an underlying LDAP server. If desired, these can occur over **SSL**.

Context-sensitive rights management

In conjunction with Teamcenter Global Services, **Security Services** introduces a facility to provide access between pairs of PDM applications where both applications (home and target) are members of a local PDM domain. The access between the home and the target applications is mediated by Global Services, so we refer to Global Services in this role as the *mediating application*. The level of authorization associated with an instance of this access depends on the identities of the initiating Teamcenter user and each of the participating PDM applications and the nature of the access. Global Services distills those attributes into an pseudo application ID and requests an application token for the pseudo application from the Login Service. Security Services returns a token to the mediating application containing an alias user ID that is recognized by the target application.

This access between the mediating application and the target application is referred to as a *trust relationship*. The target application does not authenticate the underlying user but trusts that the mediating application has authenticated and authorized that user for the activity implied by the alias user ID.

Configuring trust relationships requires additional **LDAP table configuration**, an additional Identity Service context parameter, and additional entries in the Identity Service Application Registry table.

Install Security Services

Install Security Services as appropriate for the installation tool you use to install Teamcenter:

Deployment Center

If you use Deployment Center, install Security Services using Deployment Center.

Teamcenter Environment Manager (TEM)

If you use TEM, install Security Services using Web Application Manager and TEM:

1. Install the Security Services web applications (Login Service and Identity Service) using Web Application Manager on a Windows or Linux machine.

2. Record information about the Teamcenter applications you want to connect to Security Services in the *Teamcenter Security Services Application Registry worksheet*. This worksheet shows the Teamcenter applications, their IDs, URLs, LDAP attributes, and trusted settings.

3. If you use LDAP, **configure an LDAP server for Security Services**.

4. Configure the Security Services **Login Service** and **Identity Service** components.

5. In the **Features** panel in TEM, make sure you select the **Teamcenter Security Services** feature under **Server Enhancements**.

Verify the Security Services installation:

1. **Verify your Security Services installation**. You should be able to log on to Security Services, even if no other Teamcenter products have been installed.

2. Configure and deploy your Teamcenter products, one at a time, **verifying that they work with Security Services**. If you are using SSL and are experiencing issues, try configuring without SSL first (if possible), and then change back.

If you encounter problems during Security Services installation and configuration, see the *Troubleshooting* for possible solutions.

Guidelines for Security Services deployment

Following are some basic guidelines relative to a Teamcenter deployment that includes Security Services:

- User workstations must be secure. Once signed on, a single sign-on session exists on the user's workstation, which other Teamcenter applications can join without another logon window being displayed. Furthermore, the Security Services Session Agent functionality depends on the user's operating system security.

- Deploy the Login Service on a web server using secure communication, for example, Secure Sockets Layer (SSL).

- If the Security Services deployments are not behind a firewall, deploy the Identity Service on a web server using **SSL**. This requires server certificates where the Teamcenter applications are deployed.

- If the Security Services Login Service and Identity Service are deployed behind a firewall, no SSL is needed between them. Clients do not see this URL traffic and the firewall guarantees communication between them is secure.

- Set the token lifetime configuration setting low, typically 1 to 5 minutes.

- Configure the session lifetime long enough to avoid the user inconvenience caused by multiple sign-ons but short enough to provide a measure of security in case users inadvertently leave their

machine logged on but unattended. Typically, other mechanisms (for example, operating system security features) are in place to address this latter issue, in which case the session lifetime can be set from several to many hours.

2. Setting up an LDAP server for Security Services

Record Teamcenter application information for Security Services

To prepare for configuring your LDAP server, enter information about the Teamcenter applications you intend to deploy in the *Teamcenter Security Services Application Registry worksheet*. Refer to this information when you configure LDAP for the Identity Service web application.

Note:

Consider the following when completing the *Teamcenter Security Services Application Registry worksheet*:

- The application ID entered for each Teamcenter application installation must correspond to the appropriate environment variable in each product. For example, Teamcenter 2412 has a **TC_SSO_APP_ID** environment variable.
- The application root URL must be specified for any Teamcenter client. This URL can be in either IPv4 or IPv6 format.
- The application user name attribute is the attribute name in **LDAP** that holds the user's alias.
- The **Trusted Application** box is set to **false** unless you are configuring a Global Services trusted application.
- The **Strip Domain Name** box is set to **false** unless all user IDs are checked for embedded domain names.
- A deployment can include multiple instances of a specific Teamcenter product. Each instance must have its own application ID.
- The Login Service web application is a required entry in this table. Its application root URL box can contain any value, or it can be left empty.
- Two or more web tier instances with distinct application IDs can be configured with a single Teamcenter server instance. This requires no special configuration in Security Services.

Teamcenter Security Services Application Registry worksheet

Application ID	**Application root URL** *(can be in either IPv4 or IPv6 format)*	**LDAP user name attribute**	**Trusted application**	**Strip domain name**

Choose an LDAP server

Use an LDAP browser to verify proper LDAP settings for the Security Services Identity Service configuration. Several (for example, JXplorer) are freely available.

Use the browser to connect to an LDAP server, examine naming contexts and DN entries, and verify that the connection settings are valid. Tools of this type can also authenticate users and examine their attributes to verify that the attribute names and values are correct.

Security Services uses an LDAP version 3 directory server for user authentication and application-level authorization. Normally, each Teamcenter user must have a user ID and password set in the directory server, since it is against this directory server that Security Services attempts to authenticate the user's logon credentials.

Some Security Services configurations do not require changes to the LDAP schema. Specifically, if all users are authorized to launch all Teamcenter applications (note that each application does its own authorization checks, so Security Services cannot override the application's authorization settings) and the user aliases within all Teamcenter applications match the user logon IDs, no schema changes are necessary.

Further, some Security Services configurations do not require LDAP. Specifically, if the above is true and an SSO gateway is used, no LDAP is required. Otherwise, your deployment uses LDAP, and you can modify the schema to achieve certain features, specifically application authorization (the ability to launch a specific Teamcenter application) and aliasing (mapping the logon ID to an application-specific user alias).

To use application authorization and aliasing using Security Services for a Teamcenter application, you must define a user ID attribute for that Teamcenter application in the LDAP server schema. Then, set the value of that attribute to their alias for the specific application. If the user is not authorized for that application, do not set that attribute for that user. Because Teamcenter applications can share a single attribute, this implies that if a user is authorized for one application, they are authorized for the others and that they share the same alias for each user.

Note:

Because the Login Service is a Teamcenter application, it has a corresponding attribute like any other Teamcenter application. Because every Teamcenter user must have a value set for this attribute, setting the value on this attribute for a user authorizes the user to logon to Teamcenter.

LDAP configuration examples

Example 1 – Defining multiple attributes

If you install Engineering Process Management and Systems Engineering and Requirements Management and each user has a distinct alias in each application, then:

1. Create three attributes in the LDAP repository:

   - **uid** for the Login Service
   - **TcEngUserName** for Engineering Process Management
   - **TcReqUserName** for Systems Engineering and Requirements Management

2. Create (or modify) an object class in the LDAP repository to hold the three attributes, and attach that object class to each Teamcenter user entry in the repository.

3. When you configure the Identity Service, configure the Application Registry table with these values:

Application ID	Attribute name
TCSSOLoginService	**uid**
TcEngineering	**TcEngUserName**
TcRequirements	**TcReqUserName**

4. For Teamcenter user **JHill** who is authorized for all Teamcenter applications, the attribute values in the LDAP directory server might be:

   - **uid** = **Joe**
   - **TcEngUserName** = **Joey**
   - **TcReqUserName** = **Joseph**

5. For Teamcenter user **FSmith** who is only authorized to use Engineering Process Management, the attribute values in the LDAP directory server might be:

- **uid** = **Fred**
- **TcEngUserName** = **Freddy**

Example 2 – Defining a single shared Teamcenter attribute

If you install Engineering Process Management and Systems Engineering and Requirements Management and all users share a single Teamcenter user alias that is distinct from their logon ID, and all Teamcenter users are authorized to launch all Teamcenter applications, then:

1. Create one attribute in the LDAP repository, for example, **TcUserName**. This is shared by all Teamcenter applications, including the Login Service.
2. Create (or modify) an object class to hold this attribute, and attach that object class to each Teamcenter user entry in the repository.
3. When configuring the Identity Service, configure the Application Registry table with these values:

Application ID	Attribute name
TCSSOLoginService	**TcUserName**
TcEngineering	**TcUserName**
TcRequirements	**TcUserName**

Example 3 – Defining pseudo application IDs

As in Example 1, if you install Engineering Process Management and Systems Engineering and Requirements Management and each user has a distinct alias in each application, and a pseudo application ID, **TcPseudoEngineering**, is defined in the Application Registry table intended for user **FSmith**, then:

1. Create four attributes in the LDAP repository, for example:
   - **uid** for the Login Service
   - **TcEngUserName** for Engineering Process Management
   - **TcReqUserName** for Systems Engineering and Requirements Management
   - **TcPseudoUserName** for Engineering Process Management, but with enhanced user rights
2. Create (or modify) an object class in the LDAP repository to hold the four attributes, and attach that object class to each Teamcenter user entry in the repository.

3. When you configure the Identity Service configuration, configure the Application Registry table with these values:

Application ID	Attribute name
TCSSOLoginService	**uid**
TcEngineering	**TcEngUserName**
TcRequirements	**TcReqUserName**
TcPseudoEngineering	**TcPseudoUserName**

4. For Teamcenter user **JHill** who is authorized for all Teamcenter applications, the attribute values in the LDAP directory server are the same as in Example 1.

5. For Teamcenter user **FSmith** who is authorized to use Engineering Process Management both directly and via a mediating application with enhanced rights, the attribute values in the LDAP directory server might be:

   - **uid** = **Fred**
   - **TcEngUserName** = **Freddy**
   - **TcPseudoUserName** = **AdminFred**

Example 4 – No schema changes

If you install several Teamcenter applications and the user alias within each application is the same as the user's logon ID, and every user is authorized to launch each Teamcenter application (or each Teamcenter application does its own authorization), then:

- No new attributes (or object classes) are needed in the LDAP repository. Use the existing user ID attribute (for example, **uid**) for all the Teamcenter applications, including the Login Service.
- When you configure the Identity Service, configure the Application Registry table with these values:

Application ID	Attribute name
TCSSOLoginService	**uid**
TcEngineering	**uid**
TcRequirements	**uid**
TcEnterprise	**uid**

How to enable encrypted LDAP

> Note:
>
> If you use **LDAPS** protocol but do not configure the Active Directory server to use **LDAPS**, the connection to the LDAP server fails.
>
> If you use **TLS** protocol but do not configure the Active Directory server to use **TLS**, Active Directory defaults to a non-SSL connection without notifying the user. (You can, however, verify the connection protocol by viewing Security Services log files.)
>
> Of these two protocols, the **TLS** protocol is considered superior. The **LDAPS** protocol was an experimental, and now deprecated, protocol.

1. Configure the Active Directory server with Certificate Services.

2. Extract the certificates for the primary domain controller (PDC) and the domain using Base64 x509 encoding. Save these certificates (for example, as *filename***.cer**) in the file system for the web application server where Security Services is deployed.

3. Add both of the certificates to the **cacerts** file of the keystore in the **jre\lib\security** folder for the Java installation used by the web application.

   > Note:
   >
   > For convenience, you can manage certificates using a certificate management tool like Keystore Explorer.

4. Save the **cacerts** file.

5. Define the **ConnectionType** parameter in the identity server with one of the following sets of values:

   - Protocol: **TLS**

     Port: **389**

   - Protocol: **LDAPS**

     Port: **636**

   > Note:
   >
   > These ports are the default values used for Active Directory. If you use a different LDAP tool or different ports for LDAP, adjust the port values accordingly.

6. Save the WAR file for the identity service.

7. Redeploy the Teamcenter web application deployable file for the identity service.

8. Test the client by logging on to verify the encrypted LDAP configuration is successful.

3. Configuring Security Services

Context parameter worksheets

After **installing the Login Service and the Identity Service** and configuring your LDAP Identity Provider, you can now configure the Login and Identity Services for your site using the Teamcenter Web Application Manager.

> Note:
>
> Siemens Digital Industries Software recommends you configure and verify Security Services on your system before you install and configure other Teamcenter applications.

Before you begin configuring Security Services, complete the *Login Service context parameter worksheet* and the *Identity Service context parameter worksheet* context parameter worksheets. Your Security Services product configuration will proceed quickly if you have this information ready.

There are context parameters associated with both the Login Service and the Identity Service. Enter the value in the **Value** column of these tables.

> Note:
>
> Default values appear in parentheses at the end of each description.

Login Service context parameter worksheet

Context parameter name	Description	Value
webmaster	Specifies the email address of the administrator to whom questions and comments about the application should be directed (**change_me_webmaster_name@change_me_email_domain**).	
tcsso.login_service.appid	Specifies the Teamcenter ID of the Login Service. This value should match the corresponding entry in the Application Registry table (**TCSSOLoginService**).	
tcsso.login_service.http_connection_close	(Deprecated) Indicates whether to close or keep alive the Session Agent	

Context parameter name	Description	Value
	HTTP connection following each request to the Login Service. Change this value only to solve connection issues between the Session Agent Applet and the Login Service (**keep-alive**).	
tcsso.login_service.rp_cookieNamePattern	Specifies a pattern or set of patterns describing the names of cookies used by reverse proxy servers protecting Teamcenter applications (**PD-H-SESSION-ID, PD-S-SESSION-ID, SMSESSION**).	
tcsso.login_service.proxyURL	Specifies the **protocol:// host:port** URL for the Login Service when used with load balancing or SSO gateway proxies. If the value is empty, Security Services uses the HTTP request to retrieve the **protocol://host:port** information needed for the Login Service information (*blank*).	
tcsso.login_service.sso_service_url	Specifies the URL of the Identity Service (**change_me**). Note: The URL must be an IPv4 address.	
identityServicePassword	Specifies a password shared between the Login Service and Identity Service to sign and encrypt security information to prevent it from being forged or viewed. It can be any value, but Siemens Digital Industries Software recommends that	

Context parameter name	Description	Value
	it contain both letters and numbers and be at least eight characters long. The password must be the same as that specified in the Identity Service configuration (**change_me_password**).	
tcsso.behind_sso_gateway	Indicates the presence of a third-party SSO solution (**false**).	
tcsso.gateway.field.type	Indicates how the gateway transmits credential information (Teamcenter user ID) in the HTTP request to the Login Service (**header**). Choose from the following values: • **header** • **cookie** • **principal** • **parameter** • **remote_user** • **client_certificate** • **filter_class** This value is ignored if **tcsso.behind_sso_gateway** is false.	
tcsso.gateway.field.name	Indicates the name of the chosen field in **tcsso.gateway.field.type** (**COMMSSOCRED**). If **tcsso.gateway.field.type** is set to **client_certificate**, the valid values are: • **CN** (common name); the default value if set to **client_certificate**. • **SN** (surname) • **G** (given name)	

Context parameter name	Description	Value
	• **SERIALNUMBER** (serial number) This value is ignored if **tcsso.behind_sso_gateway** is false.	
tcsso.username.filter.class	Specifies the fully qualified class name that is responsible for extracting the Teamcenter user name from the client certificate other than the **CN** (common name), **SN** (surname), and **G** (given name) fields (*blank*). This value should only be used when **tcsso.gateway.field.type** is set to **filter_class**.	
tcsso.client.enable.notice.consent.logon.banner	Indicates whether a notice and consent logon banner are displayed to the user (**false**).	
tcsso.forgotten.password.URL	Specifies the URL to associate with the hypertext link for forgotten passwords on the logon window. If blank, no link appears. If the URL is valid, it is assumed that it links to a site that provides the capability for recovering or regenerating user passwords (*blank*).	
tcsso.online_help.enable	Enables Security Services online help for users if set to **true**. If **false**, no online help is available (**true**).	
tcsso.login_service.enable_session_agent_applet	(Deprecated) Indicates whether the single sign-on applets are enabled. If **false**, applets are disabled and single sign-on between rich client and other clients does not function (**true**).	

Context parameter name	Description	Value
tcsso.login_service.force_web_browser_login	Indicates whether single sign-on among browser instances on the user's workstation is disabled. If true, every new browser instance receives a logon challenge (**false**).	
tcsso.frame_ancestors	Allows sites to disallow hosting a Security Services logon page (**none**): • On any domain by retaining the default value **none**. • On some domains but not on others. If you elect to not use this parameter, leave the value empty.	
tcsso.federation_type	When set to **UMC** to use the Siemens User Management Control (UMC) software, the Login Service relies on the federation identity provider to perform user authentication and Security Services performs as a service provider, thereby authorizing users for Teamcenter applications (**none**).	
tcss.federation_url	When set, the TCSS Login Service redirects the user to the URL for the Federation Identity Provider to perform user authentication and return the user to the Teamcenter application after successful authentication (*blank*). The URL needs to be a single sign-on (SSO) URL that conforms to the Security	

Context parameter name	Description	Value
	Assertion Markup Language (SAML) standard, that is, a service provider-initiated login URL provided by the SAML identity provider (IdP).	
tcsso.federation_reply_url	When set, the Login Service provides this URL to the federation identity provider to redirect the user following authentication (*blank*).	
tcsso.federation_logout_url	When set, the Login Service provides this URL to the federation identity provider to log the user out from both the identity and service providers (*blank*).	
tcsso.cors_whitelist	When set, the list of domains to be white listed are honored for cross-origin resource sharing (CORS) support (*blank*).	
DEBUG	Note: • Siemens Digital Industries Software recommends you use **Log4j2** for debugging, as the **DEBUG** option is deprecated. • Using the **DEBUG** option produces a voluminous amount of logging information. (Deprecated) Indicates whether the Login Service prints debug information (**warn**). If set to **true**, the Login Service prints **info**, **warn**, **error**, and **fatal** information.	

Context parameter name	Description	Value
	Choose from the following values: • **warn** • **false** • **true** • **debug** • **info** • **error** • **fatal**	
tcsso.login_service.enableCsrf	Enables cross-site request forgery (CSRF) protection and allows compatibility with clients on older releases (**false**).	

The *Identity Service context parameter worksheet* lists context parameters set for the Identity Service.

Identity Service context parameter worksheet

Context parameter name	Description	Value
webmaster	Specifies the email address of the administrator to whom questions and comments about the application should be directed (**change_me_webmaster_name @change_me_email_domain**).	
identityProvider	Specifies the class of identity provider used by the Identity Service (**com.teamcenter.ss.identity.spi. LDAPIdentityProvider**).	
identityServicePassword	Specifies a password shared between the Login Service and Identity Service to sign and encrypt security information to prevent it from being forged or viewed. It can be any value, but Siemens Digital Industries Software recommends that it contain both letters and numbers and be at least eight characters long. The password must be the same as that specified	

Context parameter name	Description	Value
	in the Login Service configuration (**change_me_password**).	
passwordLifetime	Specifies the lifetime, in seconds, for an SSO gateway attempt. This time limits a replay attack. This is configurable to accommodate latency in deployments (**30**).	
mediatorPassword	Specifies a password shared between the Identity Service and a mediating application. Used to encrypt tokens passed to the mediator for later distribution to target applications participating in trust relationships (**change_me_password**).	
tokenLifetime	Specifies the lifetime, in seconds, of a Teamcenter Security Services token. This short-lived, one-time-use credential, is a secure substitute for the user's real credentials (**600**).	
sessionLifetime	Specifies the number of minutes that an Teamcenter Security Services session can be idle before it is terminated, where idle means no logon or logoff events. Generally, this should be several hours. If a Teamcenter Security Services session expires, it does not harm Teamcenter application sessions the user has open. The user gets a new challenge if they start a new application (**600**).	
tcsso.LogLevel	Specifies the Teamcenter Security Services events to be logged. Use one of the following values: • **None** – No Teamcenter Security Services events are logged. • **Authentication failures** – Only authentication failures or unauthorized access events are logged. • **Authentication successful** – Authentication successes are logged.	

Context parameter name	Description	Value
	• **All authentication events** – All positive and negative security-related events are logged. The Teamcenter Security Services authentication log is written to the **%APPDATA%\teamcenter\sso\sso_Authentication.log** (Windows) or the *$HOME***/.teamcenter/sso/sso_Authentication.log** (Linux) file (**Authentication failures**).	
tcsso.AuthLogDir	Specifies the destination directory for the authorization log file. If left blank, the log file is written to the same directory as the servlet debug file (*blank*).	
LDAPVersion	Specifies the minimum LDAP version used for connections (**3**).	
PasswordResetEnabled	Enables display of the change password logon window. This window provides fields for users to enter their new password. This capability is available only for deployments where the user repository is Active Directory. If set to **true**, there must also be an entry in the **LDAP Configuration table** listing the **QueryDN** and **QueryDNPassword** parameters granting administrative permissions. These credentials are used for password update operations. If **false**, and a reset or expired password is detected, the user receives an error message (**false**).	
PasswordResetMessage	Displays additional information when prompted to change password. This can be a link to a change password service (*blank*).	
GatewayAliasingEnabled	Specifies that user ID aliasing for Teamcenter applications is enabled and always performed unless Security	

Context parameter name	Description	Value
	Services is configured in gateway mode. If this parameter is **true**, user ID aliasing is performed in gateway mode as well. Unless a valid LDAP repository is configured in Security Services, this parameter must be set to **false** (**false**).	
ReferralsEnabled	Specifies LDAP referrals, where one LDAP server references another, are enabled if set to **true**. If you enable this feature, you must have entries for the referred-to LDAP servers in the LDAP Configuration table (**false**).	
ReferralHopLimit	Specifies the maximum number of hops (jumps) to follow in sequence during a referral. This value is ignored if **ReferralsEnabled** is **false** (**5**).	
DEBUG	Note: Using the **DEBUG** option produces a voluminous amount of logging information. (Deprecated) Specifies the conditions for debugging output files (**false**). If set to **true**, the Login Service prints **info**, **warn**, **error**, and **fatal** information. Choose from the following values: • **warn** • **false** • **true** • **debug** • **info** • **error** • **fatal**	

Debugging Security Services

Using Log4J2 for debugging

Security Services now uses Apache Log4J2 for debugging. Log4J2 behavior is driven by a configuration file, **log4j2.xml**. Security Services has four versions of that file for the various components.

Server	For the Login Service and Identity Service web applications, the **log4j2.xml** configuration file is in the *LoginServiceWebAppName***webapp_root\WEB-INF\classes** of the Security Services **insweb** folder. Modify this file prior to using the Web Application Manager (**insweb**) to **Generate Deployable File** for each web application.
Client	The Session Agent **log4j2.xml** file is located in Session Agent install location folder, for example, **C:\Users\uhyxm9\AppData\Teamcenter\SecurityServicesSA**.

Configuring Apache Log4J2 for debugging

When configuring Apache Log4J2 for debugging, the allowable logging levels are:

- **ALL** or **TRACE**

  Specifies the most fine-grained informational events, including function call entry and exit.

- **DEBUG**

  Specifies fine-grained informational events that are useful to debug Security Services.

- **INFO**

  Specifies informational messages that highlight the progress of Security Services at the coarse-grained level.

- **WARN** (default)

  Specifies potentially harmful situations and possible misconfigurations.

- **ERROR**

  Specifies error events that may still allow Security Services to continue running.

- **FATAL**

  Specifies very severe error events that may lead Security Services to abort.

- **OFF**

Specifies no output to the debug files.

Each of the levels includes all the debugging of the levels below it.

The **log4j2.xml** file has a **<Loggers>** section where these flags are set. For example, the server version contains:

```
<Loggers>
   <Root level="ERROR">
       <AppenderRef ref="STDOUT"/>
   </Root>
   <Logger name="com.teamcenter._ss" level="WARN" additivity="true">
       <AppenderRef ref="serverFileWriter"/>
   </Logger>
   <Logger name="com.teamcenter.ss" level="WARN" additivity="true">
       <AppenderRef ref="serverFileWriter"/>
   </Logger>
   <Logger name="org.apache.xmlrpc" level="WARN" additivity="true">
       <AppenderRef ref="serverFileWriter"/>
   </Logger>
</Loggers>
```

To change the log level, replace each instance of the default **WARN** level as appropriate.

Log file locations

To synchronize the log file locations for Teamcenter components, Teamcenter Security Services log files can be found in both server and client locations.

Server	The server log files are created under: **Windows systems**: *base-log-file-location***server** **Linux systems**: *base-log-file-location*/**server** These log files are identified by web application name.
Client	The client log files for the Session Agent and rich client libraries are created under: **Windows systems**: *base-log-file-location***client** **Linux systems**: *base-log-file-location*/**client**

Adjacent to each of those directories, for both the client and server components, is an archive folder. Once daily, or when the log file reaches a size of 32MB, the current log file is compressed, date-stamped, and moved to the archive folder. A fresh log file is then initialized.

Specify the log file locations with the SIEMENS_LOGGING_ROOT environment variable

Siemens Digital Industries Software recommends you set the **SIEMENS_LOGGING_ROOT** environment variable to specify the Teamcenter Security Services (TcSS) log file locations.

By default, since the environment variable is not set, the log files are in the user home directory.

Host OS	Default base log file location
Windows	**%USERPROFILE%\Siemens\logs\TcSS**
Linux	**$HOME/Siemens/logs/TcSS**

If the **SIEMENS_LOGGING_ROOT** environment variable is set, the TcSS files are created according to that environment variable:

Host OS	Default base log file location
Windows	**SIEMENS_LOGGING_ROOT\TcSS**
Linux	**SIEMENS_LOGGING_ROOT/TcSS**

> Note:
>
> The user home directory depends on which user profile is used to run the TcSS component. For the Login and Identity Service, this means it depends on which user profile is used to run the Java web application server. When the application server is run as a Windows service, this profile usually defaults to the Local Service account. In this case, the user home directory is **%systemroot%\ServiceProfiles\LocalService**.

Configuring the Login Service

Modify a web application

To configure the web application information, context parameters, and the Login Input Definitions context table, open the **Modify Web Application** dialog box in Web Application Manager.

1. Launch the Web Application Manager.

   Windows: Browse to the *WEB_ROOT* directory and double-click the **insweb.bat** program icon.

   Linux: Change to the *WEB_ROOT* directory and type the **insweb** command.

2. In the **Web Applications** list, select the Login Service application name, and then click **Modify**.

   This opens the **Modify Web Application** dialog window.

Modify web application information

In the **Modify Web Application** dialog box, click **Modify Web Application Information**. The Web Application Manager displays the **Modify Web Application Information** dialog box for the Login Service. Modify the following information:

1. If you are using load balancing, you must select the **Distributable** ☑ check box.

2. Enter a time-out value in minutes in the **Session Timeout** box. This indicates how many minutes of inactivity are permitted before the web server terminates the session and effectively ends the user's Security Services session.

   > Note:
   >
   > Ensure that you set the load balancer's session timeout interval to a value equal to or greater than the Teamcenter session timeout values.

What is the difference between session timeout and the value of the sessionLifetime context parameter?

The default value for the web application *session timeout* kept by the web application server is 30 minutes; the Security Services **sessionLifetime** kept by Teamcenter is 600 minutes.

Session timeout is set when you create the Login Service using the Web Application Manager. If the Login Service web application *session timeout* occurs, the web container purges the Login Service; this can result in login challenges, even though the underlying Security Services **sessionLifetime** has not timed out.

Configuring load balancer timeouts

If you use a third-party load balancer, ensure its time-out setting values are equal to or greater than the timeout settings for Teamcenter entered in the Web Application Manager **Session Timeout** box. This indicates how many minutes of inactivity are permitted before the web server terminates the session on the web server. From the **Modify Web Application** dialog box, click **Modify Web Application Information**. The Web Application Manager displays the **Modify Web Application Information** dialog box for the **Session Timeout** box.

The Teamcenter web tier and Teamcenter Security Services Login Service maintain client session information. When deployed behind a load balancer, it is important that all requests from a given client are routed to the same back-end web tier or Login Service instance once a session has been established for that client. For that purpose, load balancers typically have a stickiness or affinity setting. This must be set in the load balancer configuration for these Teamcenter web applications. Also, ensure the load

balancer's session time-out interval is set to a value equal to or greater than the Teamcenter session time-out values. Otherwise, the load balancer time-out eclipses the Teamcenter time-out. Neglecting either of these considerations can lead to apparently random and unexpected behavior as the load balancer switches between active web application instances.

Modify context parameters

From the **Modify Web Application** dialog box, click **Modify Context Parameters**. To modify context parameters for the Login Service, do the following:

1. In the **Name** column, select the context parameter you want to change.

2. In the **Value** column, click the current value and change the value. The **Description for Selected Parameter** box contains a description of the context parameter you selected.

   Note:

   - The **Req** (required) box indicates whether the selected context parameter is required. Required context parameters must contain a value. However, Security Services does not currently use this feature. So, none of the boxes is checked.

   - The default value for the Login Service application ID, **TCSSOLoginService**, is valid. However, this ID must also appear in the Application Registry table of the Identity Service.

   - For many deployments, the only value requiring modification is the **tcsso.login_service.sso_service_url**. This typically invokes a URL of the form:

     `http://`*webserverhost*`:`*webserverport*`/sso_service`*WARfilename*

     For example: **http://mspm011:7001/tcssoidentityservice**.

3. When you are finished modifying the context parameters, click **OK**.

   Note:

   Changes to context parameters do not take effect until you generate a WAR file and deploy the WAR file on your web application server.

Values found in the Login Input Definitions table

The Login Input Definitions table specifies how user credentials are sent to the Login Service and, in turn, the Identity Service and identity provider. By default, the Login Service expects a user name and password to be sent in form fields named **tcsso_username**, **tcsso_password**, and **tcsso_locale**, respectively; the Identity Service expects the user name and password values to be associated with keys named **User** and **Password**, respectively.

> Note:
>
> Do not modify these values unless the corresponding values are changed in the Security Services JSP pages. The following changes are only necessary in cases where gateway integration is not possible using the **tcsso.gateway.field.type** and **tcsso.gateway.field.name** context parameters.

The following table describes the values for the **Modify Table – Login Input Definitions** dialog box fields.

Field	Description
Name	Specifies the name of the field, for example, **tcsso_username**.
Where	Specifies where the name field is found. The choices are **form**, **cookie**, and **header**.
Required	Specifies whether or not the field is displayed; the choices are **true** and **false**.
Idpkey	Specifies the **key** associated with the underlying identity provider.

The following table describes the login input configuration values.

Parameter	Description
tcsso_username	Specifies the user name of the user.
tcsso_password	Specifies the user's password.
tcsso_newpassword	Specifies the user's new password. If **PasswordResetEnabled** is **true**, the **tcsso_newpassword** value is used to set the new password.
tcsso_confirmpassword	Confirms the user's new password is entered correctly. Note: The **tcsso_confirmpassword** parameter is not associated with an **IdProvider** key.
tcsso_locale	Specifies the user's locale.

To add a row to a table, click **Add Row**. To remove a row from a table, click **Remove Row**.

When you finish modifying the table, click **OK**.

Note:

Changes do not take effect until you generate a WAR file and deploy it on your web application server.

Changing the login input definitions is very seldom necessary. Most authenticating gateways are easily accommodated with the provided context parameters.

Configuring a load balancer, reverse proxy, or SSO gateway

During authentication, Security Services references several files on the Login Service deployment, such as page backgrounds and style sheets. Those references are generated dynamically with Java Server Pages executed on the Security Services Login Service. Therefore, the Login Service needs its own URL from the client's perspective. If the client machine does not directly access the server where the Login Service is deployed (because of an SSO gateway, reverse proxy, or load balancer), configure the **tcsso.login_service.proxyURL** parameter. The **tcsso.login_service.proxyURL** parameter must be the URL users enter when accessing their application and/or Security Services.

The **tcsso.login_service.proxyURL** parameter is configured in two ways depending on the intermediary server.

- Behind an SSO gateway:

  Syntax:

  protocol://*host*:*port*

- Behind a reverse proxy or load balancer:

  Syntax:

  protocol://*host*:*port*/*cpath*

  Where *protocol* = **http** or **https**; *host*:*port* = intermediary server; *cpath* = **Login Service servlet**

Customizing the logon window

The **Security Services** Login Service provides a default logon window containing three fields: user name, password, and locale. If your site requires the user to enter additional information, for example, an account number, you can add an additional field to the logon window.

Configuring the Identity Service

Overview of the Identity Service configuration

The Security Services Identity Service authenticates Teamcenter users against the LDAP repository or repositories configured within it. However, when the Security Services Login Service is configured behind an authenticating gateway, the Security Services Identity Service does not authenticate ordinary Teamcenter users that are channeled through the Security Services Login Service. It requires no configured LDAP repository for that purpose. However, the Identity Service may still authenticate the user credentials for server-side applications, such as ITK commands or the FTS indexer. Those authentications either come directly from **tcserver**, or are submitted as SOA requests to the Teamcenter webtier, bypassing any authenticating gateways and the Login Service. That authentication requires appropriate LDAP configuration of the Identity Service. Such configuration does not interfere with gateway authentication of ordinary Teamcenter users.

Modify context parameters for the Identity Service

> Note:
>
> Identity Service configuration is essentially the same whether deploying in single sign-on mode or authentication-only mode.

1. Launch the Web Application Manager.

   Windows: Browse to the *WEB_ROOT* directory and double-click the **insweb.bat** program icon.

   Linux: Change to the *WEB_ROOT* directory and type the **insweb** command.

2. In the **Web Applications** list, select the Identity Service application name, and then click **Modify**.

   This opens the **Modify Web Application** dialog window.

3. Click **Modify Context Parameters**. The Web Application Manager displays the **Modify Context Parameters** dialog box for the Identity Service.

4. Fill in each parameter with **values appropriate to your site**.

   > Note:
   >
   > To help determine the proper LDAP settings, use an LDAP browser utility. This helps you set, verify, and capture the LDAP settings you need for the Identity Service.

5. After you modify the context parameters, click **OK**.

Note:

Changes to the context parameters do not take effect until you generate a WAR file and deploy the WAR file on your web application server.

Modifying Identity Service tables

Modifying Identity Service tables

The following tables are associated with the Identity Service:

- SSO Token Specification
- Application Registry
- LDAP Domain map
- LDAP Configuration

From the **Modify Web Application** dialog box, click **Modify Tables**.

Application Registry table configuration values

To modify the Application Registry table, select Application Registry from the **Modify Tables** dialog box and click **Modify**.

This displays the **Modify Table** dialog box.

Each Teamcenter solution installed on your system has configuration values associated with the Application Registry table. Set the configuration values using the **Modify Table** dialog box with the values in the *Teamcenter Security Services Application Registry worksheet* you compiled at the end of the Security Services installation process.

The following table describes the columns in the Application Registry table.

Field	Description
Application ID	Specifies the ID associated with the Teamcenter application installed on the system, for example, **TCEnterprise**.
Application Root URL	For Teamcenter applications participating in the Teamcenter Security Services web redirection protocol, this required field is the URL pointing to the root path of that application.

Field	Description
	Note: This URL can be in either IPv4 or IPv6 format.
LDAP UserName Attribute	Specifies the LDAP attribute for the application-specific user name, for example, **TCEnterpriseUserName**.
Trusted Application	If **true**, this entry is a pseudo application. Teamcenter users do not log into this application directly, but instead use it to look up an alias user ID for an actual Teamcenter application in a trust relationship established on their behalf via a mediating application (**false**).
Strip Domain Name	If **true** for this application, all user IDs are checked for embedded domain names of the form *Domain\UserID*. When generating Security Services tokens for that application, the domain name and separator are stripped, leaving only the user ID (**false**).

Note:

- Include a row for each distinct Teamcenter application deployed in the installation, including the Login Service (the default is **TCSSOLoginService**).
- If you add or remove an application ID, you should update your *Teamcenter Security Services Application Registry worksheet*.
- Include a URL in the **Application Root URL** column for each application in the table (other than the Login Service). This is generally the base URL for the application. The application root URL for all Teamcenter clients (except Teamcenter Enterprise) is of the format:

  `http://node:portnumber`

  For Teamcenter Enterprise, this base URL is of the format:

  `http://node:portnumber/tcent`
- The application's LDAP user name attribute value must denote some attribute associated with users in your LDAP server.

SSO Token Specification table configuration values

To modify the SSO Token Specification table, select SSO Token Specification table from the **Modify Tables** dialog box and click **Modify**.

This displays the **Modify Table** dialog box.

Each token type is represented by an instance any time an array of tokens is generated.

The following table describes the columns in the SSO Token Specification table.

Field	Description
Token type	Specifies the token type (**HttpCookieToken** or **HttpHeaderToken**) to be generated.
Token name	Specifies the token name.
Cookie path	Specifies the cookie path.

LDAP Domain map

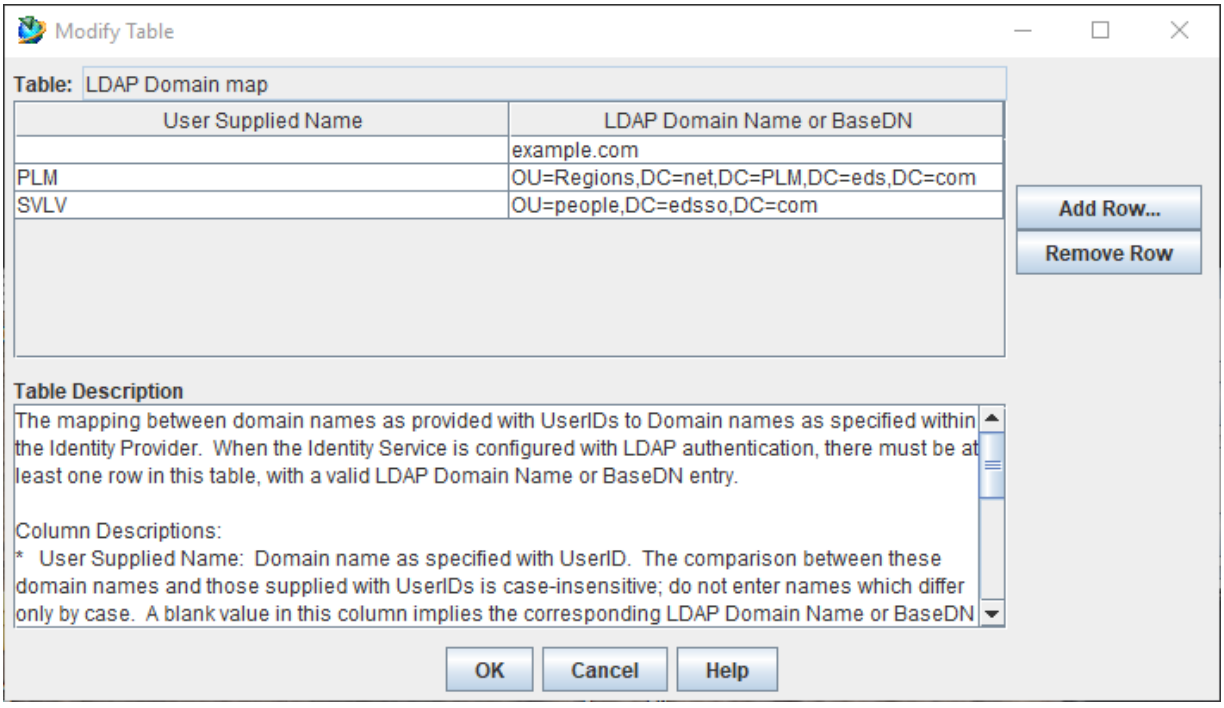
> Note:
>
> Use the **LDAP Domain map** table only if you are using LDAP referrals. Otherwise, you only need to use the **LDAP Configuration** table.

The LDAP Domain map holds the base DN values for each of the configured LDAP repositories. Entries in the **LDAP Domain Name** column can take two forms:

- If entered as a domain name (for example, **corp.plm.net**), it is converted to an equivalent base DN (for example, **DC=corp, DC=plm, DC=net**) and presented to the LDAP.
- If entered as a base DN, it is presented to the LDAP without conversion.

The LDAP Domain map also defines the mapping between domain values entered as the prefix of the user ID and the base DN values. Referring to the following example, a user ID entered as **plm\johnj** is presented to the LDAP identity provider exactly as if the user had entered **corp.plm.net\johnj**. This mapping applies only to LDAP lookups and does not affect the user IDs subsequently included in Teamcenter Security Services tokens. Set the configuration values using the **Modify Table** dialog box.

There must be one row in this table for each row in the LDAP Configuration table. Typically, one row has a blank **User Supplied Name**, making the corresponding **LDAP Domain Name or BaseDN** the default for user IDs entered without a domain prefix.



The following table describes the LDAP Domain map configuration values.

Field	Description
User Supplied Name	Specifies the domain name, as optionally specified with user ID. The comparison between these domain names and those supplied with user IDs is case-insensitive; do not enter names that differ only by case. A blank value in this column implies the corresponding **LDAP Domain Name** or **BaseDN** is the default value.
LDAP Domain Name or BaseDN	Specifies the corresponding **LDAP Domain Name** or **BaseDN** in the identity provider. • If the string contains an equal sign (=), it is interpreted as a base DN and forwarded to the LDAP unmodified. • If it is a domain name of the form **x,y,z**, it is converted to the base DN **DC=x,DC=y,DC=z** and forwarded to the LDAP.

LDAP Configuration table

Use the LDAP Configuration table to configure your LDAP servers. Each host and port combination identifies an LDAP server to provide a distinct configuration. Set the configuration values using the **Modify Table** dialog box.

The following table describes the LDAP Configuration table values.

Note:

If LDAP password change is allowed, you must use TLS, and the **Query DN** value must be an administrator level for password change permissions.

Field	Description
LDAP Ordinal	Indicates the search order for the LDAP server. The row with the lowest number in this column indicates the *primary LDAP server*. The numbers in the other lines indicate the order in which the secondary LDAP servers are searched, from lowest to highest.
LDAP Host	Specifies the host name of the LDAP server. Populate this field in one of the following ways: • As a single LDAP host name. • As a space-delimited list of host names, where the listed hosts contain identical LDAP repositories. Connect attempts occur in the order listed. • As an Active Directory domain name, in which case Security Services performs a run-time DNS query to resolve the available Active Directory domain controllers (DCs). Priority and weight attributes for the DCs determine the connect order, according to RFC 2782, When specifying specific host names, these host names can each optionally include a trailing colon and port number. If a port is not included, the LDAP port number is used. When configuring an SSO environment with primary and non-primary LDAP servers using referrals, the *refURL* in the LDAP `referral` object must match an LDAP `host` value in the LDAP configuration table for users to be able to log in. The *refURL* and `host` value must both be either a domain name or a fully-qualified host name.
LDAP Port Number	Specifies the port number of the LDAP server. This is ignored for any LDAP host parameter that includes a colon and port number, or if the host parameter is an Active Directory Domain name.

Field	Description
	Note: If **LDAP Port Number Override?** is set to **Y**, the port number is used.
LDAP Port Number Override?	Specifies whether the LDAP host is an Active Directory domain name and uses the LDAP port number in place of the port returned by DNS (typically 389). If this is true, enter **Y**. Otherwise, leave this field empty or enter **N**.
LDAP Connect Type	Specifies the type of LDAP server connection (**tls**,**ldap**, or **ldaps**). • **tls** creates non-SSL LDAP connections, but then uses the startTLS protocol to promote the connection to TLS. Siemens Digital Industries Software recommends this connection type if it is supported by your LDAP. • **ldap** creates non-SSL LDAP connections. • **ldaps** creates SSL LDAP connections. • **auto** determines the connection type dynamically for each LDAP server. Note: If certificates are required, generate the keystore file, and for JBoss, add the following option to the **run.bat** file: `-Djavax.net.ssl.trustStore=`*path-to-keystore-file*
Max LDAP Connections	Specifies the maximum number of connections that can be created per Identity Service instance for each LDAP server. The value must be at least **2**. Larger values indicate lower resource contention at the expense of higher resource consumption. Smaller values conserve resources, but these can cause some blocking during logon due to resource contention. In a clustered environment with many Identity Service instances, Siemens Digital Industries Software recommends a value between **2** and **20**.
Query DN	Specifies the DN used to authenticate to the LDAP servers.

Field	Description
	• If this field is blank, anonymous authentication is used. • If **PasswordResetEnabled** is **true**, this field must not be blank and the DN must have administrator privileges.
Query DN Password	Specifies the password for the query DN.
LDAP BaseDN	Specifies the value used with the **LDAP Host** and **LDAP Port Number** values to connect to secondary LDAPs without the need for defining an LDAP referral.
UserObjectClass	Specifies the user object class to use as part of the search filter.
Fall Back to User Attribute	Specifies how to handle the case where the Application Registry table **LDAP UserName Attribute** value is not found in the associated LDAP system for this row. A value of **Y** indicates that the **User Attribute** value from the LDAP Configuration table is used instead of the **LDAP UserName Attribute** value, if the **LDAP UserName Attribute** is not found. A value of **N** indicates that the **LDAP UserName Attribute** must exist.
User Attribute	Specifies the user attribute to search.
LDAP Connection Setup Delay	Specifies the interval in seconds to wait between initiating a parallel connection to each successive server in the list when multiple LDAP servers are specified in LDAP hosts or when multiple domain controllers are discovered using DNS lookup. A value of **-1** indicates a serial connection to the servers. A value of **0** indicates parallel connections are initiated to all servers at once.
LDAP Connection Timeout	Specifies the interval in seconds to wait before abandoning an LDAP request, which can include connection, search, and bind attempts. If **LDAP Connection Setup Delay** is greater than **0**, set this value to be greater to allow multiple connection attempts. A value of **0** indicates unlimited connection attempts.

LDAP Domain map considerations

Using the LDAP Configuration table and the LDAP Domain map

You can use the LDAP Configuration table and LDAP Domain map table together to solve some common administrative problems related to multiple user ID populations:

- Case 1: Distinct Teamcenter user IDs and Teamcenter administrative IDs
- Case 2: Mixed gateway and LDAP authentication
- Case 3: Multiple corporate domains

Case 1: Distinct Teamcenter user IDs and administrative IDs

Common Security Services deployments use the corporate LDAP repository for authenticating ordinary Teamcenter users. However, many customers also maintain a set of Teamcenter administrative users, who have no corresponding entry in the corporate LDAP. Those users can be authenticated using a separate local LDAP repository.

For example, consider two LDAP repositories:

- **Admin LDAP**

  This local repository, which is maintained and visible locally, only contains administrative User IDs.

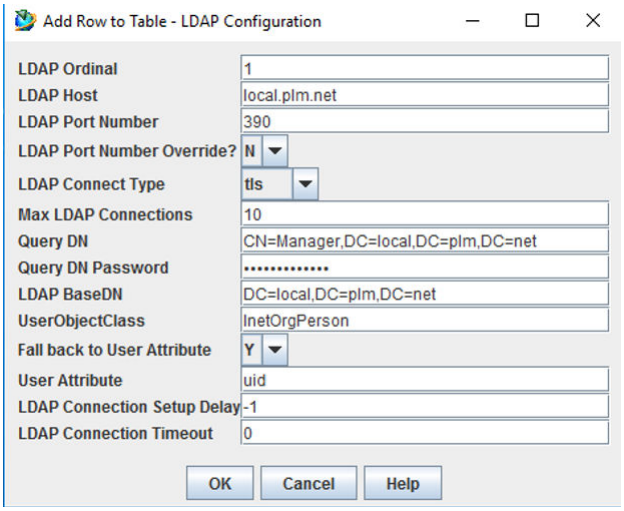
- **Corporate LDAP**

  This global repository is visible company wide.

Each repository has an LDAP configuration.

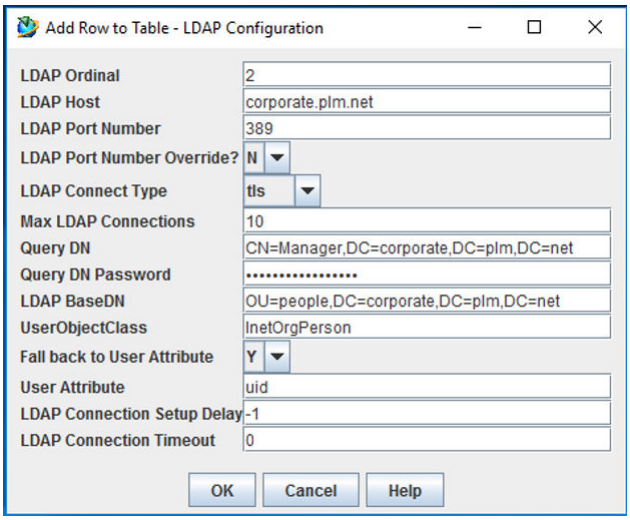
Admin LDAP	Corporate LDAP
local.plm.net	**corporate.plm.net**
390	**389**
CN=Manager,DC=local,DC=plm,DC=net	**CN=Manager,DC=corporate,DC=plm,DC=net**
passwordLocal	**passwordCorporate**
DC=local,DC=plm,DC=net	**OU=people,DC=corporate,DC=plm,DC=net**

In the Identity Service, configure each of the LDAP servers.

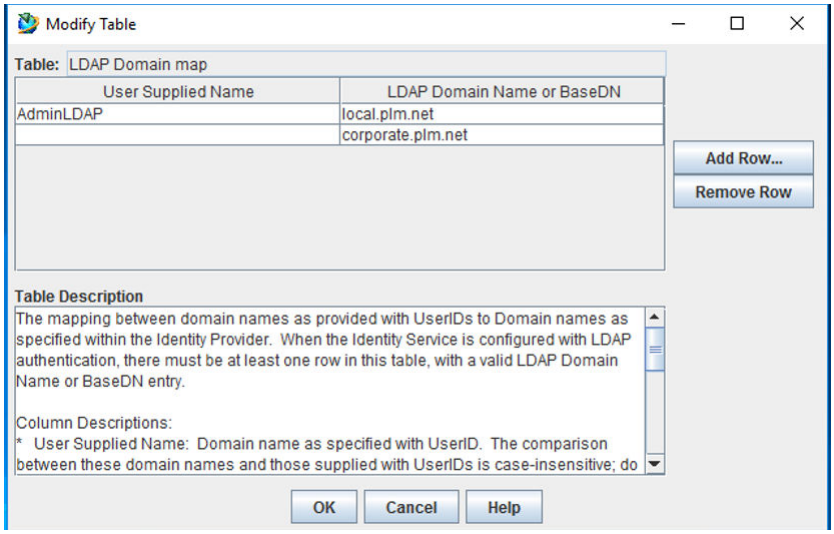


These are the **Admin LDAP** parameters.

Define credentials to use when connecting to other LDAP servers during referrals.



The LDAP Domain map table defines the base DN for each LDAP server. It also defines the mapping between the domain values entered as part of the user ID and the domain values used for authentication in the underlying LDAP repository. The entry with a blank **User Supplied Name** box is associated with a base DN to be used for users who specify no domain name prefix on their user IDs.



The LDAP Domain lookup table substitutes the **BaseDN** value when a user prepends a domain name of the form **ADMINLDAP\UserID** to their user ID.

The **Admin LDAP** is configured to refer to the **Corporate LDAP** when given the **Corporate LDAP** base DN, but the **Corporate LDAP** knows nothing of the local one.

Use cases

- An ordinary user logging on without a prepended domain is looked up first in the **Admin LDAP** because that LDAP is designated **Primary** in the LDAP configuration table. The **Admin LDAP** (local) immediately returns a referral to the **Corporate LDAP**, triggered by the corporate **BaseDN** value. The search retries, this time with the corporate LDAP configuration. The search completes at the **Corporate LDAP**.

  There is the slight overhead of an LDAP referral for this use case.

- An administrative user prepends a domain name to the user ID, for example, **LOCALLDAP\Tc-admin-user**. This replaces the base DN with **DC=local,DC=plm,DC=net** in the initial LDAP lookup, and that initial search completes.

Case 2: Mixed gateway and LDAP authentication

When the Security Services Login Service is configured behind a gateway (that encompasses authenticating reverse proxies, PKI authentication, SAML, and Kerberos), you can configure the Identity Service with one or more LDAP repositories. So configured, the Identity Service can, for instance, authenticate administrative Teamcenter users, where the authentication requests come directly from a TcServer instance when those users invoke ITK commands.

This is useful when multiple Login Service instances are configured to share a single Identity Service deployment. An example would be where one Login Service instance is deployed behind a gateway, and is intended for external Teamcenter users. A second Login Service instance is configured for

LDAP lookup, and is intended for internal Teamcenter users. In this case, the shared Identity Service deployment performs LDAP authentication only for the users of the latter Login Service.

Case 3: Multiple corporate domains

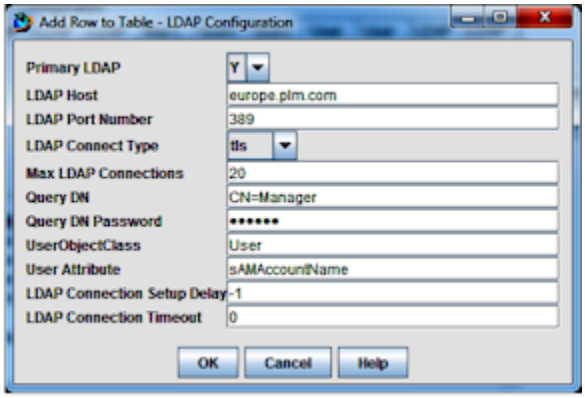
Consider a corporate environment with two domains and corresponding base DNs:

- Domain **usa.plm.com** with **DC=usa,DC=plm,DC=com** base DN
- Domain **europe.plm.com** with **DC=europe,DC=plm,DC=com** base DN

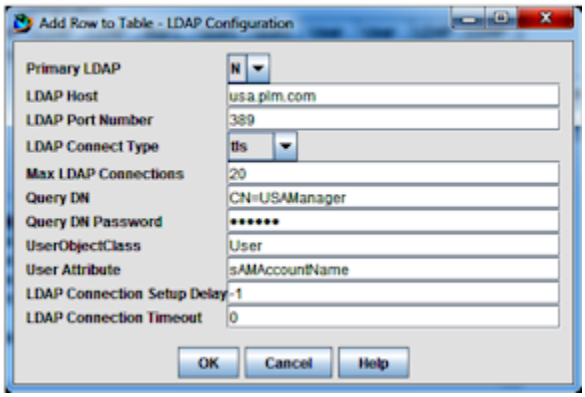
The domains have separate user communities, and it is possible that the same user ID could exist in both domains but refer to different users. However, Teamcenter users have globally unique user IDs.

This is readily handled in Security Services, providing the following conditions are met:

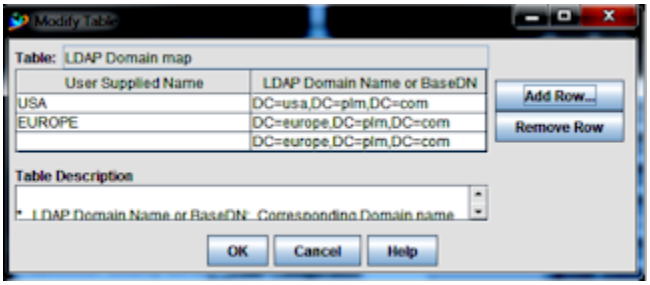
- One of the domains is considered primary and default. That domain is the one designated Primary in the **LDAP Configuration**, where **europe** is the default domain:



- A secondary domain is also defined.



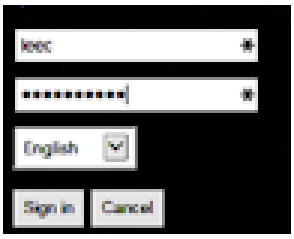
- Entries are provided in the LDAP Domain map for each potential domain name.



The primary LDAP repository is configured to recognize the base DN of the secondary domain. That is, if a request arrives at the primary LDAP (**EUROPE**) with a base DN of **DC=usa,DC=plm,DC=com**, that primary LDAP responds with a referral exception directing the caller to the **USA** domain. Also, this configuration allows users of the **EUROPE** domain to enter their user IDs either with or without the **EUROPE** prefix.

Use case

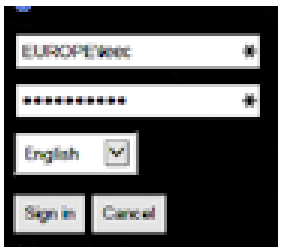
Log on as a user in the default domain.



This scenario uses the default **EUROPE** domain for the LDAP lookup. There is no referral because the **BaseDN** value is correct for that LDAP. The user authenticates providing the user ID and password are valid.

Use case

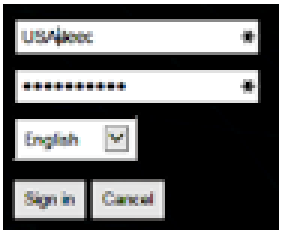
Log on as a user explicitly in the **EUROPE** domain.



In this scenario, a domain is included with the user ID, so the **BaseDN** entry for that domain is used. As before, there is no referral and the user authenticates if the user ID and password are valid.

Use case

Log on as a user in the **USA** domain.



In this scenario, the base DN corresponding to the **USA** domain is **DC=usa,DC=plm,DC=com**, and replaces the default base DN for the query. This time, there is an immediate referral exception, and the caller (the Identity Service) formulates a request to the secondary LDAP. The user authenticates on the secondary LDAP if the user ID and password are valid.

> Note:
>
> These scenarios can be combined. For example, there can be a local administrative LDAP in addition to multiple corporate LDAP domains.

Configuring the secure socket layer (SSL)

Overview of secure socket layer (SSL) configuration

HTTP over SSL (HTTPS) requires the contacted server to supply a certificate that the client subsequently validates. Certificates can be obtained through various means and installed into a web server or local certificate store. Clients receiving a certificate validate it against its certificate store, and on that basis, accept or reject the connection.

There are several Security Services communication channels over which HTTPS traffic can flow. From a security standpoint, the critical connections are those that cross a firewall, for example, from the user's workstation to the Login Service and Identity Service. For performance reasons, you should not configure backend servers to use SSL when invoking the Identity Service because that communication occurs behind established firewalls.

Enabling SSL for a Security Services deployment involves deploying the Security Services components to a web server configured for SSL and configuring the various Teamcenter clients and servers appropriately.

Configuring FIPS with Security Services

Security Services relies on the application server and Java Runtime Environment (JRE) to provide Federal Information Processing Standards (FIPS) compliance. The Login Service and Identity Service web applications, as well as the client-side Session Agent application have been verified on the following FIPS-enabled application servers and JREs:

- Oracle JRE with open source Bouncy Castle FIPS JCE provider
- IBM WebSphere Application Server
- IBM J9 VM with IBMJCEFIPS provider

The steps followed to enable FIPS mode in these application server and JREs can be found at the following vendor URLs:

- **https://docs.oracle.com/middleware/1213/wls/SECMG/fips.htm#SECMG768**
- **https://www.ibm.com/support/knowledgecenter/en/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/fips.html**
- **https://www.bouncycastle.org/fips-java**

Enable SSL for Security Services components

You can deploy the Security Services components on web servers enabled for SSL.

See the application web server's documentation for information on enabling and configuring SSL. Although you can use a demonstration certificate, this requires more configuration effort for the Teamcenter applications.

In this example, the integration machine's application server **SSLHost** is configured with port **7002** for SSL. The Login Service deployment is **LoginService11** and the Identity Service deployment is **IdentityService11**. To connect to the Login Service using SSL, clients use HTTPS requests, for example:

https://SSLHost:7002/LoginService11

It is not necessary to configure the Login Service.

If you want to use SSL between the Login and Identity Services, configure the **tcsso.login_service.sso_service_url** Login Service context parameter using HTTPS, for example:

https://SSLHost:7002/IdentityService11

Enable SSL for Teamcenter clients

Communication from rich clients to the Login Service is always via a browser.

Browsers come preconfigured with various certificates. When a browser receives a certificate from an unknown or untrusted source, it prompts if you want to accept the certificate one time or if you want to install the certificate in the browser. Therefore, when you first use Teamcenter with SSL enabled, the browser prompts you to install a certificate.

Enable SSL for Teamcenter applications

1. Configure the Identity Service URL.

   Set the Identity Service URL to an HTTPS URL, for example:

   https://SSLHost:7002/tcssoservice

   Because each Teamcenter product has a different way to set this URL, see the appropriate Teamcenter product documentation for details. For the Login Service, use the Web Application Manager to set this context parameter.

2. Trust the server's certificate.

   Configuring this is a function of which Teamcenter Security Services libraries have been integrated. Therefore, this varies depending on which Teamcenter product is being configured. If you need to obtain a copy of a certificate that you want to add to your certificate store, point your browser at the Identity Service over HTTPS (for example, **https://SSLHost:7002/tcssoservice11/xmlrpc**) and when the browser prompts if you want to trust the certificate, save the certificate in a file named **thecert.cer**.

   - Community Collaboration

     For NET applications, the Teamcenter Security Services .NET library accepts all certificates, even demo certificates. This is safe because the single Identity Service URL configured in **Community Collaboration** is set by an administrator.

   - Teamcenter Enterprise and Engineering Process Management (C applications):

     The technique is platform dependent:

     - Linux systems:

       You can make the Teamcenter Security Services C library on Linux systems accept all certificates with the following command:

       ```
       setenv EAISSL_DISABLE_CERTIFICATE_CHECK True
       ```

     - Windows systems:

       The Teamcenter Security Services C library requires that the server's certificates be accepted. Some certificate authorities are likely already accepted. If not, add them to your local machine's certificate store using **CertMgr**, a utility in the .NET Framework SDK.

       ```
       certmgr -add -c xxx.cer -s -r localMachine Root
       certmgr -add -c thecert.cer -s -r localMachine CA
       ```

- Systems Engineering and Requirements Management, Teamcenter portfolio, program and project management, Integrator, and Login Service

  If any of these Java applications require the use of SSL when invoking the Identity Service, you may need to import the web server's certificate into the appropriate JVM. Generally, your JVM is configured to accept certificates from the popular CAs like VeriSign, so importing is not necessary. If your server is using a demonstration certificate or one generated using **keytool**, you may need to import a certificate.

  To import a certificate, determine which JRE is associated with the Login Service. This is the JRE associated with the web application server in which the Login Service is executing. Because many machines have more than one JRE, and some web servers have their own JRE, it is critical to locate the correct **JRE** folder.

  After locating the **JRE** folder, go to the **lib/security** folder. You can import using the **keytool** utility in the **jre/bin** folder:

  a. Copy the certificate to the **jre/lib/security** folder.

  b. Import the certificate to your certificate store. Assuming the **cacerts** file is in the **jre/lib/security** folder, change to that folder and run **keytool**:

     ```
     keytool -import -alias somename -keystore cacerts -file
     thecert.cer
     ```

- Teamcenter

  Set the **TEAMCENTER_SSL_CERT_FILE** environment variable in the **tc_profilevars** property file, immediately after the **TC_DATA** variable setting, as follows:

  ```
  TEAMCENTER_SSL_CERT_FILE=${TC_DATA}/pom_transmit/ca-bundle.crt;
      export TEAMCENTER_SSL_CERT_FILE
  ```

  In this example, the CA file (**ca-bundle.crt**) is located under the **pom_transmit** directory.

Debug SSL issues

If you used these configuration steps and the communication fails, the issue may be caused by an incompatible cipher suite. The first phase of the SSL handshake involves negotiation of the cipher suite to use during data transfer. If no common cipher can be negotiated, the communication fails.

Security Services supports both 40-bit and 128-bit ciphers.

To diagnose SSL issues, add the following options to the JVM command line:

```
java -Djavax.net.debug=all ...
```

Because each application web server has its own means for modifying the JVM command line arguments, see the appropriate server documentation.

This setting generates logging output about the handshake and supported cipher suites.

Deploying Security Services on web application servers

You can deploy Security Services on the following web application servers:

- Sun Java System application server
- Oracle application server
- IBM WebSphere application server

For more supported web application servers, see the Hardware and Software Certifications knowledge base article on Support Center.

When using these servers, there are vital **configuration considerations** for each of these servers. Siemens Digital Industries Software recommends you review these configuration considerations.

Setting environment variables

There are four user-configurable environment variables that impact Security Services:

- **TCSSO_LOGIN_SERVICE_URL**

  Enables the WebSEAL feature within clients. Set this variable to the URL address of the Security Services Login Service as configured for the **proxy server**.

- **TCSSO_HARD_LOGIN_TIMEOUT_SECONDS**

  Specifies the amount of time (in seconds) to display the consent login banner on which the user must click the **I Accept** button. The default setting is **150**.

- **TCSS_DEBUG_LEVEL**

  Enables debug output to go to the **%APPDATA%\teamcenter\sso\debug.log** file for the .NET library. It accepts the following case-insensitive values:

  - **true**
  - **false**
  - **debug**

- **info**
- **warn**
- **error**
- **fatal**

If this variable is not defined, Teamcenter sets the level to **warn**. If the variable is defined with no value or and invalid value, Teamcenter sets the level to **debug**.

- **TCSSO_SAM_AUTH_ZONE**

  Specifies an IdP zone that is registered with SAM Auth, which is used to authenticate users.

4. Verifying Security Services

Test Identity Service

Ping the Identity Service through a web browser using the following URL:

http://*host***:***port***/***identity-service-war-file***/json**

If the request results in an error, the problem may be one of the following:

- The web server failed to start. If so, check the web server host and port numbers.
- The Identity Service WAR file did not deploy correctly. If so, consult the web server log files and administration console for more information.

Alternatively, you can ping the Identity Service using the following URL:

http://*host***:***port***/***identity-service-war-file***/xmlrpc**

- If the resulting window displays an XML string containing **SSOSystemException** and the **Internal error: null request** message, the Identity Service started successfully.
- If the resulting window displays an **SSOConfigurationException** exception, the Identity Service failed to initialize. Check the following LDAP settings in the Identity Service configuration: **LDAPHosts**, **LDAPPortNo**, **QueryDN**, and **QueryDNPassword**. Also, consult the web server's log files to obtain more information.
- If the request results in an HTTP error, the problem may be one of the following:
  - The web server failed to start. If so, check the web server host and port numbers.
  - The Identity Service WAR file did not deploy correctly. If so, consult the web server log files and administration console for more information.

Test Login Service

1. Log on to the Login Service using the following URL:

   http://*host***:***port***/***tcssoLOGINservicewarfile*

2. If this window does not load, proceed to step c, which indicates the Login Service servlet did not load. Otherwise, click the **Enter Teamcenter** link on that window.

   a. If the logon window appears, enter a valid user name and password.

- If the Welcome to Teamcenter window appears with an **Exit** button, and a small browser appears on the desktop (this may be hidden behind other windows), Security Services is correctly installed and configured.

- If an error window appears, this often means the Identity Service URL configured for the Login Service is incorrect. The URL, including the protocol portion (http://), should be:

  http://*host***:***port***/***tcssoservicewarfile*

- If an **Invalid User** message appears, make sure the user is in the directory using the LDAP administration console. Then, verify that the base DN value configured for the Identity Service matches what is defined in the LDAP. Also, verify that the **UserAttribute** configuration parameter is correct: In the LDAP's administration console, examine the full DN (**entryDN**) for the user and make sure the naming attribute (for example, **cn** or **uid**) matches what you set for **User Attribute** in the Identity Service's configuration settings.

- If an **Invalid Password** message appears, make sure the user's password is set correctly and that you are entering the correct case.

- If an authorization failure occurs, then some attributes are missing for that user in LDAP.

b. If a blank window appears, this may indicate that the Java Plug-in is not configured correctly for your browser. Perform the following steps:

   A. Open the Control Panel.

   B. Open the Java Plug-in Control Panel. If this is not listed, obtain it by installing a JRE (for example, Java 2 Runtime Environment, Standard Edition 1.5.0_07).

   C. On the **Basic** tab, enable the Java Plug-in.

   D. On the **Browser** tab, select the box for your browser.

   E. On the **Advanced** tab, select the appropriate JRE.

   F. Restart your browser.

c. If the logon window does not appear, one of the following events may have occurred:

   - The web server failed to start. Check the web server host and port numbers.

   - The Login Service WAR file did not deploy correctly. Consult the web server log files and administration console for more information.

Test Java API documentation

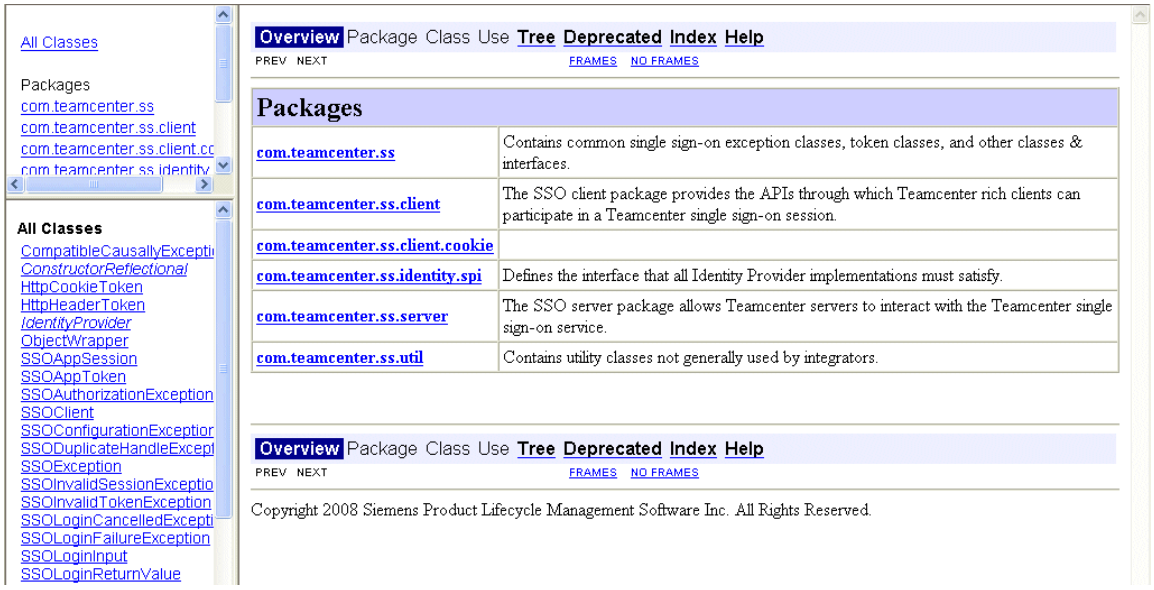
If you added Java API documentation as part of the Login Service solution, test it using the following URL:

http://*host***:***port***/***tcssoLOGINservicewarfile***/teamcenter/javadocs/sso_loginservice/index.html**

For example:

http://mspm011:7001/tcssologinservice/teamcenter/javadocs/sso_loginservice/index.html

You should see a window resembling the following.



Java API documentation

Verify DNS lookup of Active Directory domain controllers

If the **LDAP Host** parameters are configured with an Active Directory domain name, as opposed to specific domain controller names, it is easy to verify whether Security Services is successfully mapping the domain name to a domain controller list. Set the Identity Service **DEBUG** context parameter to the **INFO** level, log on using Security Services, and inspect the resulting **debug.log** file.

- Successful domain name mapping

  In the following example of a successful mapping, the **LDAPHosts** context parameter is set to **example.com**:

```
11/10/28  13:14:31.343 INFO   ConnectionService:getServers()
     Servers: DEFRDPLM001.example.com:389 uscidplm005.example.com:389 …
```

There is no **NamingException** error, and the returned servers differ from the **LDAPHosts** value.

- Unsuccessful domain name mapping

  Unsuccessful domain name mapping can occur typically in Security Services because the Domain Name Server (DNS) does not recognize the **LDAPHosts** value as an Active Directory domain. In such cases, text similar to the following appears in the **debug.log** file.

  In the following file sample, the **LDAPHosts** context parameter was set to **svlv6080.example.com**:

  ```
  11/10/28  13:11:10.705 INFO
       ConnectionService:findLDAPServersInWindowsDomain()
       NamingException. Explanation: DNS name not found
       [response code 3] ResolvedName: RemainingName:
       _ldap._tcp.svlv6080.example.com
  11/10/28  13:11:10.705 INFO
       RefPrincipal:search() Searching for user at or below:
       ou=people,dc=edssso,dc=com with filter:
       (&(objectclass=inetOrgPerson)(uid=EUser5))
  11/10/28  13:11:10.736 INFO
       ConnectionService:getServers() Servers: svlv6080.example.com
  11/10/28  13:11:10.767 INFO
       ConnectionService:getServers() Servers: svlv6080.example.com
  ```

  In this example, there is a **NamingException** error, and the **getServers()** method simply returns the configured **LDAPHosts** value. This is expected and correct behavior if **LDAPHosts** is not an Active Directory domain name.

5. Customizing Security Services

Overview of Security Services customization

You can customize Security Services to meet specific needs in several ways:

Client-certificate authentication customization

You can customize client-certificate authentication using a smart card or a soft certificate, such as a **.p12**, **.pfx**, or **.jks** file.

Logon window customization

You can customize the logon window, the window containing the user challenge, to meet the needs of a specific deployment. For example, you can add another authentication field.

Identity provider customization

Security Services provides pluggable authentication. For example, you can develop a custom driver to use a different authentication service instead of LDAP v3-compliant repositories.

SSO gateway interoperation

Security Services supports interoperation with SSO gateway products using configuration. If the standard configuration options are not sufficient, you can customize Security Services as necessary.

Password management support

Security Services provides the hooks whereby a password management facility can be invoked to support cases where users have forgotten their passwords or want to regenerate their password.

Siemens Digital Industries Software customization support

Siemens Digital Industries Software is committed to maintaining compatibility between Teamcenter product releases. If you customize functions and methods using published APIs and documented extension points, be assured that the next successive release will honor these interfaces. On occasion, it may become necessary to make behaviors more usable or to provide better integrity. Our policy is to notify customers at the time of the release *prior* to the one that contains a published interface behavior change.

Siemens Digital Industries Software does not support code extensions that use unpublished and undocumented APIs or extension points. All APIs and other extension points are unpublished unless documented in the official set of technical manuals and help files issued by Siemens Digital Industries Software.

The Teamcenter license agreements prohibit reverse engineering, including: decompiling Teamcenter object code or bytecode to derive any form of the original source code; the inspection of header files; and the examination of configuration files, database tables, or other artifacts of implementation. Siemens Digital Industries Software does not support code extensions made using source code created from such reverse engineering.

Customize client-certificate authentication

You can customize client-certificate authentication to allow Teamcenter to use a digital certificate in a public key infrastructure (PKI) for authentication to protected Teamcenter applications.

What is client-certificate authentication?

Client-certificate authentication is an authentication method in which clients are required to submit certificates that are issued by a certificate authority that you choose to accept. Client-certificate authentication is an alternative to basic, digest, NTLM, Kerberos, or form-based authentication. It uses HTTP over SSL (HTTPS), in which the server and the client authenticate each other using a public key certificate. Client-certificate authentication provides data confidentiality, data integrity, and client and server authentication for a TCP/IP connection.

To implement client-certificate authentication, you must identify the user either by reading the user name from a certificate, or by extracting the user name from a certificate.

Read Teamcenter user names from certificates

To read a Teamcenter user name (common name, surname, or given name) from certificates, you must do the following from the Login Service:

1. Set the **tcsso.gateway.field.type** context parameter to **client_certificate**.

2. Set the **tcsso.gateway.field.name** context parameter to **CN** (common name), **SN** (surname), or **G** (given name).

Extract Teamcenter user names from client certificates

Use the following procedure to extract Teamcenter user names from the client certificates without using the common name, surname, or given name of the user.

1. Ensure the Login Service **tcsso.gateway.field.type** context parameter is set to **filter_class**.

2. Set the **tcsso.username.filter.class** context parameter to the fully qualified class name implementing the **javax.servlet.Filter** context parameter.

3. Write the custom implementation:

    a. Create a Java EE servlet filter that contains custom code to extract the user name from the certificate.

```
package com.company.security.filter;

import java.io.IOException;
import java.security.cert.X509Certificate;
import java.util.List;
```

```
import javax.naming.InvalidNameException;
import javax.naming.ldap.LdapName;
import javax.naming.ldap.Rdn;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

/**
* This is an example on how to write a custom filter. You can copy this class
* to write your own custom filter expect code logic to read user name from
* certificate field in #readUsername(X509Certificate).
*
 */
public class CustomFilter implements Filter
{
    @Override
    public void init(FilterConfig arg0) throws ServletException
    {
    }

    @Override
    public void destroy()
    {
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse
            response, FilterChain chain) throws IOException,
            ServletException
    {
        X509Certificate certificate = findCertificate(request.getAttribute
            ("javax.servlet.request.X509Certificate"));

        String userID = null;
        if (certificate == null)
        {
            // Should not happen
        }
        else
        {
            userID = readUsername(certificate);
        }

        if (userID != null)
        {
            // Teamcenter SSO uses HttpServletRequest#getRemoteUser
            // method to read user name
            chain.doFilter(new RequestWrapper((HttpServletRequest)
                 request, userID), response);
        }
        else
        {
            chain.doFilter(request, response);
        }
```

```
}

private static X509Certificate findCertificate(Object o)
{
    if (o == null)
    {
        return null;
    }
    Class<? extends Object> clazz = o.getClass();
    if (clazz.isArray())
    {
        Object[] objects = (Object[]) o;
        if (objects == null || objects.length == 0)
        {
            return null;
        }
        else
        {
            return findCertificate(objects[0]);
        }
    }
    else if (o instanceof X509Certificate)
    {
        return (X509Certificate) o;
    }
    else
    {
        return null;
    }
}

private String readUsername(X509Certificate certificate)
{
    String dn = certificate.getSubjectX500Principal().toString();
    List <Rdn>  rdns = null;
    try
    {
        LdapName ldapDN = new LdapName(dn);
        rdns = ldapDN.getRdns();
        if (rdns == null || rdns.size() == 0)
        {
            return null;
        }
        for (Rdn rdn : rdns)
        {
            // For example, We wants to use CN field value as username
            // TODO You can change code logic to read appropriate
            // field name
            if (rdn.getType().equals("CN"))
            {
                return rdn.getValue() == null ? null :
                     rdn.getValue().toString();
            }
        }
    }
    catch (InvalidNameException e)
    {
    }

    return null;
```

```
        }

        private class RequestWrapper extends HttpServletRequestWrapper
        {
            private String userName;

            public RequestWrapper(HttpServletRequest request, String userName)
            {
                super(request);
                this.userName = userName;
            }

            @Override
            public String getRemoteUser()
            {
                return this.userName;
            }
        }
    }
```

b. Package this class into a JAR file.

c. Copy the JAR file into the application *staging_location**webapp_root***WEB-INF\lib** folder.

Customize the display notice and consent logon banner

To customize and display the notice and consent logon banner to users upon logon, perform the following:

1. From the Login Service, set the **tcsso.client.enable.notice.consent.logon.banner** context parameter to **true**.

2. Modify the **login_banner.zip** file that comes with your Security Services installation.

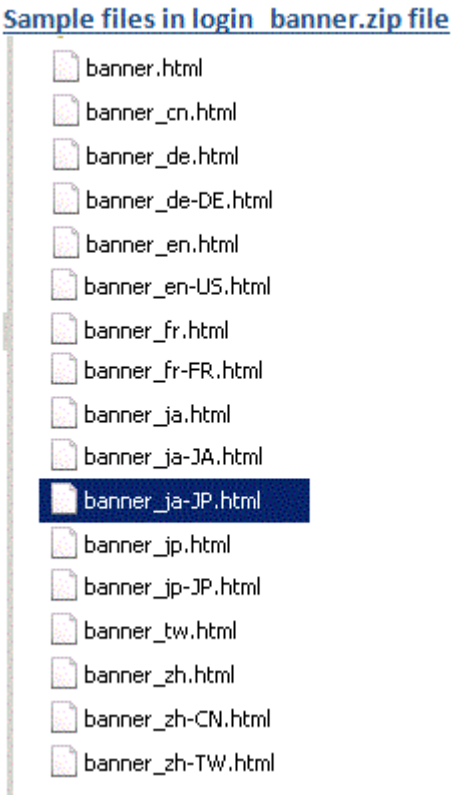
   The **login_banner.zip** file in your Security Services installation contains a default **banner.html** template file and the **banner_en.html** file that specifies the language, as denoted by the **en** extension, which indicates the language on the banner is English.

   The banner file naming convention is as follows:

   `banner[_`*language*`][-`*country*`].html`

   language and *country* are two-character ISO-8859 codes supported by Security Services.

   You can add language files to your **login_banner.zip** file, as shown in the following **login_banner.zip** sample file. In this sample file, there is a country-specific language file for each supported language supported by Security Services. The highlighted file indicates that the contents of this banner file is in Japanese for use in Japan.



Sample files in the login_banner.zip file

Create as many banner files as you need for the languages and countries you support.

3. Add your customized **login_banner.zip** file into the staging area using the Web Application Manager.

If you later customize additional files in the **login_banner.zip** file or add files to the **login_banner.zip** file, you must reinstall the **Teamcenter Security Services Login Service Web Application** web tier solution to update the display notice and consent logon banner.

1. In the Web Application Manager, select the Security Services web application and click **Modify**.

2. In the **Modify Web Application** dialog box, click **Reinstall Solutions**.

3. Select the **Teamcenter Security Services Login Service Web Application** web tier solution.

4. Redeploy the Security Services web application.

Customize the logon window

You can customize the Security Services logon window by modifying the **LoginPage.jsp** file located in the **sso/loginservice** folder in your web deployment.

For example, if your site requires the user to enter an account number, in addition to their user name and password, you can add an additional field to the logon window by editing the **LoginPage.jsp** file and adding an entry to the Login Input Definitions table when you configure Security Services using the following steps:

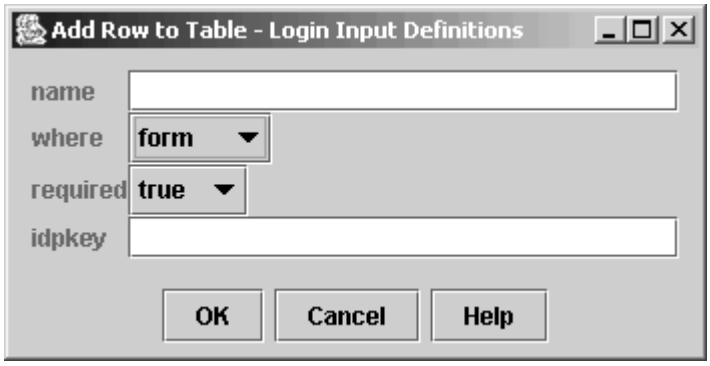
1. From the **Teamcenter Web Application Manager** dialog box, select the Login Service and click **Modify**.

2. From the **Modify Web Application** dialog box, click **Modify Tables**.

3. From the **Modify Tables** dialog box, select **Login Input Definitions** and click **Modify** to display the Login Input Definitions table.

4. From the **Modify Table** dialog box, click **Add Row**.

   This displays the **Add Row to Table – Login Input Definitions** dialog box.



   You must enter values for the fields described in the following table.

Field	Description
Name	Specifies the attribute of the user credentials.
Where	Specifies where the **Name** field is added: **form**, **cookie**, or **header**.
Required	Specifies whether or not the field is required. Select either **true** or **false** from the dropdown list depending on whether the field is required.
idpkey	Specifies the corresponding name used within the underlying identity provider.

The **idpkey** box defines the name of the box used within the underlying identity provider, for example, **Acctname**.

In this example, you must modify the underlying identity provider to look for the account name in the logon credentials (using the idpkey **Acctname**) and verify the value (for example, verify that it is a correct account number).

Customize the display for the Session Agent logout window

To customize and display the browser logout window for Session Agent:

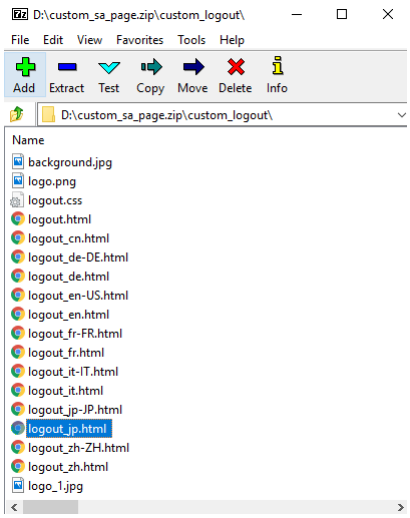
1. Modify the **custom_sa_page.zip** file that comes with your Security Services installation.

   After unzipping this file, place your logout-related HTML files inside the empty folder, **custom_logout**. Make certain your HTML files have the following naming convention:

   ```
   logout[_language][-country].html
   ```

   The default file name is **logout.html**.

   You can add language files to your **custom_sa_page.zip** file inside the **custom_logout** folder, as shown in the following **custom_sa_page.zip** sample file. Inside this sample file, there is a country-specific language file for each supported language supported by Security Services. The highlighted file, **logout_jp.html**, indicates that the contents of this logout file is in Japanese for use in Japan.



   You can create as many HTML logout files as you need for the languages and countries you support in this deployment.

   For example, the following is a sample logout file, **logout_en-US.html**:

```
<html>

<head>
<link rel="stylesheet" type="text/css" href="<host>:<port>/<app>/Logout.css">
</head>

<body background="<host>:<port>/<app>/background.jpg">

<img src="<host>:<port>/<app>/logo.png" alt="XYZ-company" width="300" height="200">
<h1>XYZ-Company Single Sign On</h1>

<div class="a">
<p>XYZ-Company is a conglomerate company headquartered in ....</p>
</div>

</body>

</html>
```

2. Replace the **custom_sa_page.zip** file in the staging directory with your customized **custom_sa_page.zip** file and reinstall the solution for the Security Services Login Service web application.

Any updates to your **custom_sa_page.zip** file require you to reinstall the Security Services Login Service web application web tier solution.

1. In the Web Application Manager, select the Security Services web application and click **Modify**.

2. In the **Modify Web Application** dialog box, click **Reinstall Solutions**.

3. Select the Security Services Login Service web application web tier solution.

4. Redeploy the Security Services web application.

Interoperating with SSO gateway products

Integration overview

You can configure and/or customize Security Services to interoperate with commercial or proprietary web-based SSO products. These products provide single sign-on for web applications accessed using a browser. Once the system has authenticated a user, subsequent launches of protected web-based applications do not involve a challenge. When forwarding requests to the target web applications, the product includes some configurable form of credentials (typically the user's ID) so the target application can determine who the user is without challenging the user independently.

Note:

If you deploy Security Services on JBoss/Wildfly, you must enable Apache JServ Protocol (AJP) before you configure the SSO gateway. Enable AJP as described in one of the following topics in the *Web Application Deployment* guide:

- *Deploy the Teamcenter web application on JBoss (H-S)*
- *Deploy on a JBoss application server (H-S)*

When integrated with an SSO gateway product, Security Services defers authentication to the commercial product and refrains from displaying the Teamcenter logon window to the user. Security Services effectively integrates all Teamcenter applications, even those with non-web-based clients behind the gateway's authentication layer. Teamcenter products do not require extra customization or configuration to work with the commercial product, because the launch of any Teamcenter client involves an interaction with the Security Services (web-based) Login Service and therefore involves the SSO product.

Teamcenter products with web clients, including the Login Service, must be registered as protected applications within the SSO product. Security Services registration includes the specification of what credential information it receives in the HTTP request forwarded by the SSO product. The Login Service must look for this information (for example, the user's ID) in the request and create an Teamcenter Security Services session on behalf of that user.

The credentials forwarded by the SSO product normally consist of only the user's Teamcenter ID. Most support a number of different mechanisms for transmitting this value, such as using a header, cookie, or parameter field. Security Services can be configured to accept several types of request fields, including header, cookie, principal, or parameter, and can be customized to work with others.

Many SSO gateway integrations with Security Services can be accomplished simply through configuration. Customization may be required for more complex deployments. In either case, integration begins with registering the Security Services Login Service with the SSO product.

Siemens Digital Industries Software does not support integration with SSO gateway software and does not guarantee it will work in all environments.

Configure Security Services for interoperation

Configuring Security Services to interoperate with commercial single sign-on products involves setting Security Services configuration variables.

1. To configure the Login Service, set the following configuration variables:

   - **tcsso.behind_sso_gateway = true**

     Specifies that a third-party SSO solution is being used.

- **tcsso.gateway.request.field = header/parameter/cookie/principal/remoteuser**

  Specifies the type of field in the HTTP request that contains the user's Teamcenter ID.

- **tcsso.gateway.field.name =** *field_name*

  Specifies the name of the field in the request, for example, **COMMSSOCREDENTIAL** or **TEAMCENTERUSERNAME**.

- **tcsso.login_service.proxyURL =** *proxy_URL*

  Specifies the *protocol*://*host*:*port* URL for the Teamcenter Security Services Login Service when used with load balancing or SSO gateway proxies.

2. To configure the Identity Service, set the **GatewayAliasingEnabled** configuration variable to **false**.

   If you set the value to **true**, an LDAP connection is made, and the Teamcenter application ID is searched in your LDAP entry to see if you are authorized for that Teamcenter application ID. If you are authorized, the system returns the value held for that entry as your ID for that Teamcenter application ID.

The integration is complete when these configuration variables are set and Security Services and other Teamcenter applications are registered with the SSO gateway product.

> Note:
>
> When configuring the Identity Service URL in server-side Teamcenter applications (such as in **tc_profilevars.bat** for **tcserver**), do not attempt to protect the Identity Service behind the gateway. Those Teamcenter applications are not equipped to respond to any authentication challenges issued by the gateway. Instead, configure the direct URL to the Identity Service.

Customizing Security Services for interoperability

About customizing Security Services for commercial products

In some cases, integration with a commercial product may not be feasible via configuration alone. For example, an SSO service may involve additional credentials or mechanisms, or a proprietary solution may involve a completely different protocol than described earlier. In these cases, Security Services can be customized.

You can customize Security Services in two locations:

- Login Service **PreLoginPage.jsp** file
- Login Service logon input definitions

In addition, you must configure the Login Inputs Definition table in conjunction with the customizations, as the table maps the fields used by the **PreLoginPage.jsp** file to credentials supplied to the identity provider.

Customizing the Login Service PreLoginPage.jsp file

The **PreLoginPage.jsp** file, an invisible window loaded prior to the logon window, determines if specific credentials are present in the request and if a Teamcenter Security Services session already exists on the user's workstation. This file is located in the web application root directory under **sso/loginservice**.

You can modify the **PreLoginPage.jsp** file as needed to work with the Teamcenter Security Services Identity Service. For example, you can modify it to look for the credentials provided by the commercial product and to forward them to the Login Manager in form fields.

As described previously, an alternate identity provider can be developed and used in place of the default identity provider. In the context of an SSO gateway product, customization may be needed to interface to an alternate credential verification service.

In most cases, such customizations are not required, since generally it is sufficient to turn off authentication as described earlier in this section.

Configuring the Login Service logon input definitions

The credentials forwarded by the modified **PreLoginPage.jsp** file and consumed by the identity provider must correspond to the entries in the Login Service's Login Input Definitions table. For example, if the **PreLoginPage.jsp** file submits a user name in a field called **myusername**, and if the key name expected by the identity provider is **User**, the user name form field should be set to **myusername** and the idpkey to **User**. Furthermore, the password row from the table is either removed or set to **not required**.

Interoperating with a password management facility

If the user can access a URL that provides support for password management, this URL can appear as a hypertext link on the logon window underneath the logon button. The hypertext, by default, is **Forgot password?**, but this can be changed if necessary. The URL must be a site where the user can, for example, recover a forgotten password or regenerate a password.

1. Build a password management servlet or use an existing web-based facility.

2. Set the Login Service **tcsso.forgotten.password.URL** context parameter to the URL of the password management servlet.

3. If the default hypertext is not acceptable, modify the Login Service text bundles for the default local and any desired localizations. These text bundles can be found in the Login Service's **WEB-INF/classes/com/teamcenter/_ss/sso_loginservice/text** folder.

6. Using smart card authentication

How does smart card authentication work?

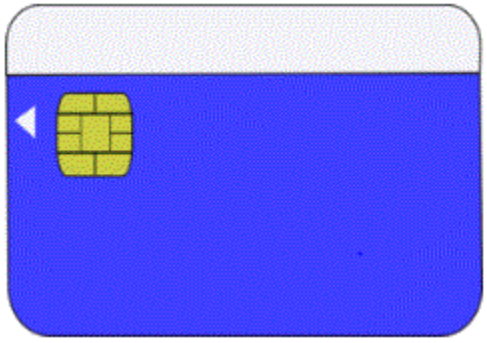
Smart card authentication allows Teamcenter to use a digital certificate in a public key infrastructure (PKI) for authentication to protected Teamcenter applications. Smart card authentication is an alternative to basic, digest, NTLM, Kerberos, or form-based authentication. It uses HTTP over SSL (HTTPS), in which the server and the client authenticate each other using a public key certificate. Smart card authentication provides data confidentiality, data integrity, and client and server authentication for a TCP/IP connection.

In client-certificate authentication, clients are required to submit certificates that are issued by a certificate authority that you choose to accept.

A client certificate can be in one of the following forms:

- A *smart card* containing a client certificate and adhering to the PKCS #11 standard.

  Smart card authentication is an example of two-factor authentication (2FA) that requires the presentation of something the user knows, such as a PIN number, and something the user has, such as a smart card.



Example of a smart card

- A file containing a client certificate and adhering to the PKCS #12 standard.

  Commonly used file extensions are **.p12**, **.pfx**, or **.jks**. These types of certificates files are also known as soft certificates. This is supported with both 32-bit and 64-bit Teamcenter client applications on Microsoft Windows, Red Hat Linux, and SUSE Linux systems.

Support deployment models

The following table shows the deployment models supported with smart card authentication.

Client type	Client transport mode	Protected applications	Number of PIN prompt
Rich client	SSO with TCCS Browserless	Login Service	1
Rich client	SSO with TCCS Browserless	Login Service and Teamcenter web tier	1
Rich client	SSO/SSO with TCCS applet	Login Service	2
Rich client	SSO/SSO with TCCS applet	Login Service and Teamcenter web tier	2

Note:

When protecting the Teamcenter web tier, you may experience a performance impact caused by a two-way SSL between the Teamcenter client and server.

Configuring smart card authentication

Both client and server sides must be configured for authentication via smart card.

Client-side configuration

To configure your client for smart card authentication, see the appropriate installation guide for your product, for example, the *Teamcenter Rich Client Installation on Windows*.

Note:

For smart card authentication to work, the client must use the HTTPS URL of the server end points.

Server-side configuration

To protect your Teamcenter application with smart card authentication, you can use either one of the following servers to enable smart card authentication:

- Reverse proxy server
  - Internet Information Service (IIS)

    To enable two-way SSL authentication using IIS, see **http://technet.microsoft.com/en-us/library/cc733010(v=ws.10).aspx**.

    After you enable smart card authentication for Teamcenter, **configure** the Login Service by setting the following context parameter values:

    - **tcsso.behind_sso_gateway = true**
    - **tcsso.gateway.field.type = header**

- **tcsso.gateway.field.name = Proxy-Remote-User**

- Java EE application server
  - Using WebSphere to host your Teamcenter application

    To enable two-way SSL authentication using WebSphere, see **https://www.ibm.com/support/knowledgecenter**.

  - Using JBoss to host your Teamcenter application

    To enable two-way SSL authentication using JBoss, see **http://docs.jboss.org/jbossweb/7.0.x/ssl-howto.html**.

After configuring your application server for two-way SSL, configure the Login Service by setting the following context parameter values:

- **tcsso.behind_sso_gateway**

  Set this value to **true**.

- **tcsso.gateway.field.type**

  Set this value to either **client_certificate** or **filter_class**.

- **tcsso.gateway.field.name**
  - If you set the **tcsso.gateway.field.type** context parameter to **client_certificate**, enter one of the following values: **CN** (Common Name), **SN** (Sur Name), or **G** (Given Name). The entered certificate attribute value (**CN**, **SN**, or **G**) is used as the Teamcenter user name.
  - If you set the **tcsso.gateway.field.type** context parameter to **filter_class**, **extract the Teamcenter user name from the client certificate**.

Server-side configuration

To protect your Teamcenter application with smart card authentication, you can use either one of the following servers to enable smart card authentication:

- Reverse proxy server
  - Internet Information Service (IIS)

    To enable two-way SSL authentication using IIS, see **http://technet.microsoft.com/en-us/library/cc733010(v=ws.10).aspx**.

After you enable smart card authentication for Teamcenter, **configure** the Login Service by setting the following context parameter values:

- **tcsso.behind_sso_gateway = true**
- **tcsso.gateway.field.type = header**
- **tcsso.gateway.field.name = Proxy-Remote-User**

- Java EE application server
  - Using WebSphere to host your Teamcenter application

    To enable two-way SSL authentication using WebSphere, see **https://www.ibm.com/support/knowledgecenter**.
  - Using JBoss to host your Teamcenter application

    To enable two-way SSL authentication using JBoss, see **http://docs.jboss.org/jbossweb/7.0.x/ssl-howto.html**.

After configuring your application server for two-way SSL, configure the Login Service by setting the following context parameter values:

- **tcsso.behind_sso_gateway**

  Set this value to **true**.
- **tcsso.gateway.field.type**

  Set this value to either **client_certificate** or **filter_class**.
- **tcsso.gateway.field.name**
  - If you set the **tcsso.gateway.field.type** context parameter to **client_certificate**, enter one of the following values: **CN** (Common Name), **SN** (Sur Name), or **G** (Given Name). The entered certificate attribute value (**CN**, **SN**, or **G**) is used as the Teamcenter user name.
  - If you set the **tcsso.gateway.field.type** context parameter to **filter_class**, **extract the Teamcenter user name from the client certificate**.

> Note:
>
> For smart card authentication to work, the client must use the HTTPS URL of the server end points.

7. Using Kerberos authentication

Kerberos authentication in Teamcenter

> Note:
>
> The following Kerberos information is specific only to Teamcenter.

Kerberos, a network authentication protocol, provides authentication for client/server applications by using secret-key cryptography. With the Kerberos protocol, a client proves its identity to a server (and vice versa) across an insecure network connection.

Kerberos allows a user to be authenticated without sending the user's password over the network. Instead, encrypted tickets derived from the password and secret key information are sent over the network.

Sign-on types

There are two ways to sign on to Teamcenter:

- Zero sign-on in browserless mode

  Teamcenter supports zero sign-on (where the user does not enter credentials to log on to applications) using Kerberos authentication in browserless mode. To enable Kerberos browserless mode, perform the following configurations:

  - Configure the client to use Security Services in Kerberos browserless mode.
  - Deploy the Security Services Login Service behind an authenticating server configured with HTTP 401-based authentication with Kerberos. This configuration requires a reverse proxy server, for example, WebSEAL, or a third-party authentication server such as IIS 7 configured to authenticate the user accessing the web service.

- Browserless sign-on

  Rich clients using Teamcenter client communication system (TCCS) can participate in a single signon session without using a web browser. In this mode, the TCCS process challenges the user for credentials to create a Security Services session. A web browser is not launched at authentication time. For this mode, the access to the Security Services Login Service requires HTTP 401-based authentication. Form-based authentication is not supported with this mode.

  A browser-based client cannot share a browserless, single sign-on session with rich clients. If a browser-based client is launched, a second sign-on challenge appears.

> Note:
>
> In browserless sign-on, the Security Services session on the client is maintained in TCCS. Therefore, you must shut down TCCS to log off your Security Services session.

Advantages of using Kerberos

Kerberos offers users the advantage of zero sign-on. Because you are already logged on to the operating system, you use your operating system identity to get into Teamcenter.

Another advantage of using Kerberos is there is no password sent across the network. Thus, Kerberos provides a security advantage.

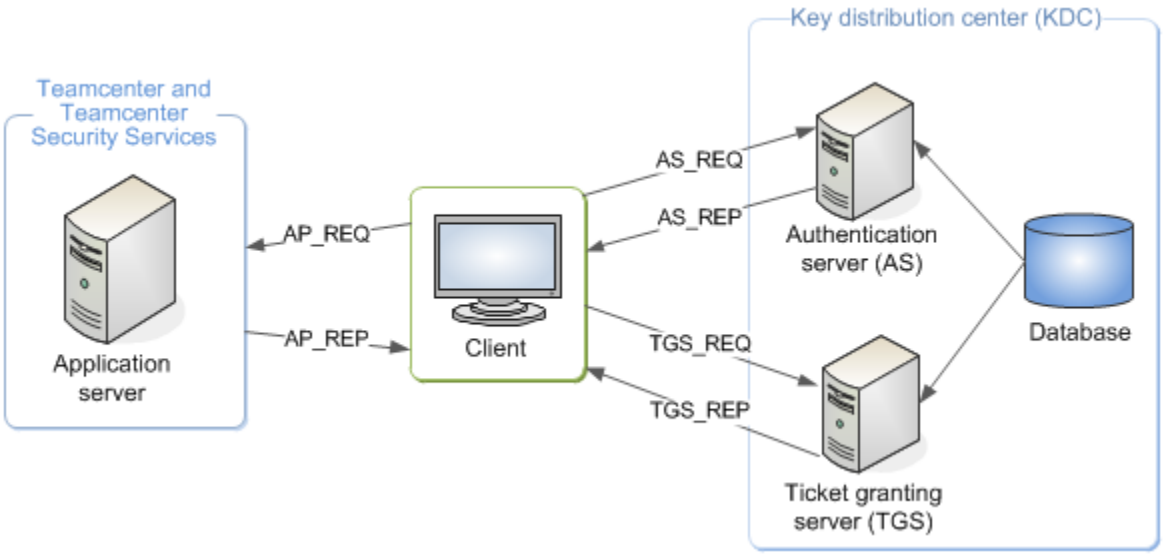
Kerberos is only supported on Windows and its use with Teamcenter is optional.

Kerberos authentication sequence

The following example illustrates the three key elements used by Kerberos:

- Client, which needs a service.
- Application server, which provides a service.
- Key distribution center (KDC), which communicates with the client and server. It contains:
  - Authentication server (AS) that contains user names and passwords.
  - Ticket granting server (TGS) that contains tickets for all the services.

  Both of these components use a database containing information about users and protected services. In Windows Integrated Authentication, the Active Directory serves as the KDC.



1. A user logs on to the client workstation using their operating system user name and password.

2. Authentication server request **(AS_REQ)**

   At logon, Windows sends a message to the authentication server containing the operating system user name. The authentication server uses the client secret key and checks to make certain the user is in its database. The client has a lifetime for validity.

3. Authentication server response **(AS_REP)**

   At logon, if the authentication server finds the user in its database, it sends a response and returns a ticket granting ticket.

4. Ticket granting server request **(TGS_REQ)**

   The client sends a request to the TGS containing the ticket granting ticket and the requested service.

5. Ticket granting server reply **(TGS_REP)**

   The TGS validates the ticket and, if valid, responds with a service ticket.

6. Application server request **(AP_REQ)**

   The client sends the service ticket to the desired service.

7. Application server reply **(AP_REP)**

   The application server validates the service ticket and responds to the request.

Setting up Teamcenter to use Kerberos

Prerequisites for Kerberos authentication

The following are prerequisites for setting up your Teamcenter installation to use Kerberos authentication:

- You must install and configure the Teamcenter rich client with TCCS on each Windows client.
- You must install Security Services behind an authenticating gateway.
- You must install Microsoft Internet Information Services (IIS) on an appropriate Windows server.

Install the unlimited strength Java cryptography extension

Because of import control restrictions, the version of Java Cryptography Extension (JCE) policy files that are bundled in the JDK 7 environment allow strong but limited cryptography to be used. Oracle provides an alternative bundle containing unlimited strength policy files.

> Note:
>
> This installation applies only to Java Runtime Environment (JRE) 7 and assumes that JRE 7 is already installed.

1. Browse to the Oracle web site regarding JRE 7:

   http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

2. Accept the license agreement and download the **UnlimitedJCEPolicyJDK7.zip** file.

3. Extract the downloaded file to a temporary directory, such as **c:\temp**.

   > Note:
   >
   > - Before proceeding, read and understand terms and condition mentioned in the JCE **README.txt** file located in the temporary directory.
   > - Siemens Digital Industries Software recommends you back up your *TC_ROOT***install****install****jre****lib****security** directory before continuing the installation.

4. Copy all files from the temporary JCE folder (for example, **c:\temp\jce**) folder to your *TC_ROOT***install****install****jre****lib****security** folder by replacing all previous files.

Installing and configuring Kerberos

To use Kerberos on your Windows client, you must perform the following:

1. Install your Teamcenter corporate server using Deployment Center or TEM and the Teamcenter web tier.

2. Install Security Services using Deployment Center or TEM.

3. Configure Microsoft Internet Information Services (IIS) and reverse proxy servers as required to support Kerberos authentication.

   For information about versions of operating systems, web application servers, and other third-party software supported for use with Teamcenter, see the Hardware and Software Certifications knowledge base article on Support Center.

4. Install Teamcenter clients and configure TCCS and Kerberos authentication.

   > Note:
   >
   > For Teamcenter versions 11.5 and later that are installed on Windows, when configuring TCCS to use Kerberos authentication, do not specify a **krb5.ini** file in the TCCS configuration, as it is no longer used for Kerberos configuration.

5. To support browserless signon, append **/tccs** to the Security Services Login Service URL. For example:

   `http://host:port/`*SSO-login-service-name*`/tccs`

   The Teamcenter Security Services login URL is defined in one of several locations, depending on how clients are configured to access Teamcenter Security Services at your site.

Client configuration	Description
Rich client is configured to use TCCS transport	Defined in the **ssologin.xml** file in the TCCS configuration directory. By default, this directory is in: *user-profile* **\Siemens\cfg\tccs***tccs-config* or *all-users-profile* **\Siemens\cfg\tccs***tccs-config*

Client configuration	Description

	If you defined a custom location using the **TCCS_CONFIG_HOME** and **TCCS_CONFIG** environment variables, the custom location is formed as:
	• **Windows systems:**
	`%TCCS_CONFIG_HOME%\%TCCS_CONFIG%`
	• **Linux systems:**
	`$TCCS_CONFIG_HOME/$TCCS_CONFIG`

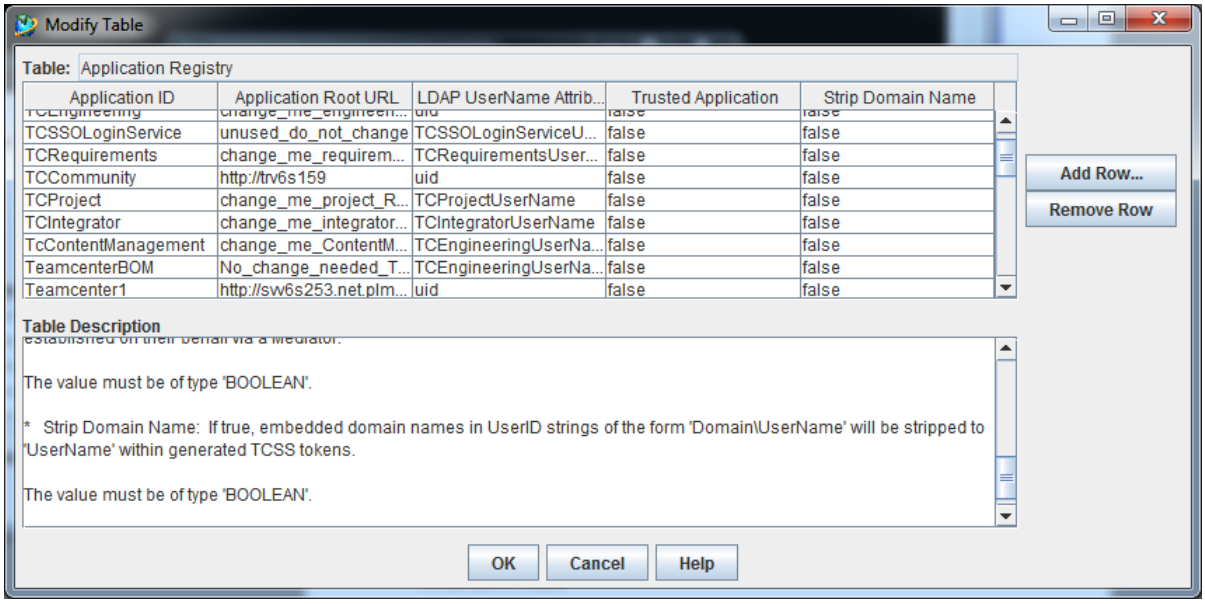
Rich client using HTTP rather than TCCS for Teamcenter Security Services	Defined in the rich client **client_specific.properties** file located in the **rac\plugins\configuration_9000.1.0** directory.
FSC is protected by reverse proxy	Defined in the location specified by the **TCSSO_LOGIN_SERVICE_URL** environment variable.

6. Configure the Teamcenter Security Services Login Service by setting the following context parameters using the **Modify Context Parameters** panel in the Teamcenter Web Application Manager:

   - Set the **tcsso.behind_sso_gateway** parameter to **true**. This parameter indicates the presence of a third-party authentication solution, which is IIS.
   - Set the **tcsso.gateway.field.type** parameter. This parameter indicates how the gateway transmits credential information, for example, how the Teamcenter user ID in the HTTP request is transmitted to the Security Services Login Service.
     - For the JBoss application server, set the value to **remote_user**.

     This parameter indicates how the gateway transmits the credential information, for example, the Teamcenter user ID in the HTTP request to the Security Services Login Service.
   - Set the **tcsso.gateway.field.name** parameter to **Proxy-Remote-User**. This parameter indicates the name of the chosen field specified by the **tcsso.gateway.field.type**.

7. Configure the Security Services Identity Service **Application Registry Table** by stripping the domain part from the authenticated user because the user in Teamcenter does not have the domain name.



8. Log on to Teamcenter.

8. Using federated authentication

What is federated authentication?

Federated authentication is an extension of identity management. It uses one of the standard identity protocols, for example, SAML, to convey trusted identities. *Federation* allows single sign-on authentication through a third party identity provider using trusted tokens. For comparison, single sign-on (SSO) is a term used for any time a user can log on to multiple applications while only authenticating once and it requires authentication that is specific to an individual organization.

Security Services supports four types of federated authentication:

- **Security Assertion Markup Language** (SAML), an industry standard protocol.
- **Security and Access Management Authentication** (SAM Auth), a Teamcenter service offering.
- **User Management Control** (UMC), a Siemens product.
- **OpenID Connect** (OIDC), a service offering.

Some base configuration parameters common to all federation types are set in the **Login Service Context Parameters** table. In addition, there are parameters specific to the SAML and SAM Auth federation types, which are set in the **federation.properties file**.

Configure your Login Service for federation mode

Before using a federated logon service for authentication, you must configure your Login Service for federation mode.

1. Launch the Web Application Manager.

   Windows: Browse to the *WEB_ROOT* directory and double-click the **insweb.bat** program icon.

   Linux: Change to the *WEB_ROOT* directory and type the **insweb** command.

2. Select the Login Service and click **Modify**.
3. Click **Modify Context Parameters**.
4. Set the following parameter values:

Context parameter name and description	Value
tcsso.federation_type Enables the federation identity provider to perform user authentication and Security Services to perform as a service provider authorizing users for Teamcenter applications (**none**).	Enter the corresponding federation type: • **UMC** Enables User Management Control. • **SAML** Enables SAMLv2 protocol. • **SAMAUTH** Enables SAM Auth. • **OIDC** Enables OpenID Connect.
tcss.federation_url Redirects the user to the URL for the Federation Identity Provider to perform user authentication and return the user to the Teamcenter application after successful authentication (*blank*). The URL needs to be a single sign-on (SSO) URL that conforms to the Security Assertion Markup Language (SAML) standard, that is, a service provider-initiated login URL provided by the SAML identity provider (IdP).	• **UMC**: **https://***UMC-host***:***port* **/umc-sso/action=loginrequest** • **SAML**: **https://***SAML-host***:***port* • **SAMAUTH**: **https://***SAMAUTH-host***:***port* • **OIDC**: **https://***OpenIDProvider-host***:***port*
tcsso.federation_reply_url Provides the URL to the federation identity provider to redirect the user following authentication (*blank)*.	• **UMC**: **https://***SSO-host***:***port***/***path-to-login-service* **/weblogin/home** • **SAML**: **https://***SSO-host***:***port***/***path-to-login-service* **/weblogin/saml_acs**

Context parameter name and description	Value
	• **SAMAUTH**: **https://***SSO-host***:***port***/***path-to-login-service***/weblogin/sam_auth_callback** • **OIDC**: **https://***SSO-host***:***port***/***path-to-login-service***/weblogin/oidc_callback**
tcsso.federation_logout_url Provides the URL to the federation identity provider to log the user out from both the identity and service providers (*blank)*.	• **UMC**: **https://***UMC-host***:***port***/***path-to-UMC-logout* • **SAML**: **https://***SAML-host***:***port***/***path-to-SAML-logout* • **SAMAUTH**: **https://***SAMAUTH-host***:***port***/session/end** • **OIDC**: **https://***OpenIDProvider-host***:***port***/***path-to-end-session-endpoint*

5. Generate the new WAR file and deploy to your application servers.

Configure Security Services federation properties

When configuring Security Services for SAML or SAM Auth federation authentication (currently not used for UMC authentication), you must manually update the property values in the **federation.properties** file. This file is located in the **WEB-INF\classes** directory where the Login Service is installed, for example:

```
C:\INSWeb_install\staging_folder\webapp_root\WEB-
INF\classes\federation.properties
```

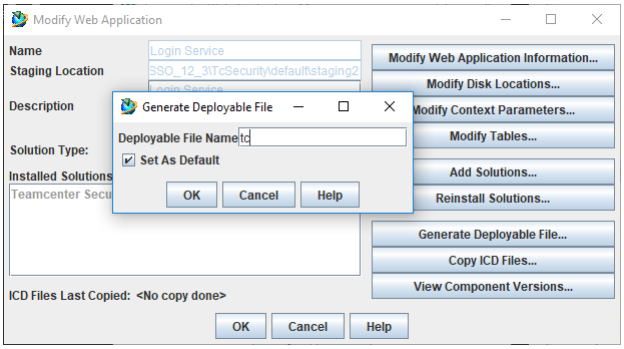
To edit these properties, enter your values and save the file. For example, to set the SAM Auth client ID, change:

```
tcsso.samauth_client_id=clientId
```

to

```
tcsso.samauth_client_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

From within the Web Application Manager, generate a new Login Service WAR file by clicking the **Generate Deployable File** button on the **Modify Web Application** window. Then, deploy this to your application server.



Because the property values shown inside brackets (**< >**) have no default values, you must change these if you enable the corresponding federation type. Values not inside brackets are set to default values; you should verify these values for each deployment.

> Note:
>
> Only the values in the **federation.properties** file that correspond to the enabled federation type are used. For example, SAML properties are ignored if the SAMAUTH federation type is enabled.

Password encoding

To encode passwords for use in the **federation.properties** file, run the following command:

```
INSWeb_install\staging_folder\webapp_root\WEB-INF\lib>
java -cp ./* com.teamcenter._ss.util.PasswordEncoder passwordToEncode
```

Example command output:

```
INFO  - 2019/09/17-03:10:35.849 UTC - [1] SSOObjectEncoder:<init>(92):
     Compression is in use. ClearpasswordToEncode,
     Encoded[OBF-zHRt1bpxnTfhDf6e89ll1g]
```

Replace the clear-text password in the **federation.properties** file with the generated encoded value beginning with **OBF-**.

Using Security Assertion Markup Language (SAML) authentication

Authenticating Teamcenter users using SAML

Security Services supports authentication using the SAML protocol. As the service provider, Security Services can connect to the Identity Provider (IdP) and consume the SAML assertion directly, without the assistance of a third-party SAML Service Provider (SP) proxy product. In this integration, Security Services is limited to the role of the SP and must be configured with an external SAMLv2 IdP.

External user IDs

With SAML SP support, you can configure Security Services with a SAML IdP that supplies users provisioned by external systems for access to Teamcenter. This enables users created in an existing corporate user database or LDAP to seamlessly authenticate with Teamcenter. Further, it can leverage users created from trusted partners where user IDs exist outside of the corporate solution.

User mapping

Federated user IDs are expected to match user IDs in Teamcenter. User IDs that do not match can be mapped to Teamcenter user IDs using the supplied ApacheDS LDAP. (See *Example 1 - Defining multiple attributes* and *Example 2 - Defining a single shared Teamcenter attribute*.)

Certificate exchange

SAML requires that trust be established between the SAML IdP and the SP during the configuration and administration phase. Being a SAML SP, Security Services accomplishes this by exchanging public key certificates with the SAML IdP. These certificates, used during authentication to sign and encrypt the SAML assertion, ensure the SAML assertion was generated by the trusted source and has not been tampered with. You can configure the certificates in the Security Services **federation.properties** file.

If you are unable to acquire corporate certificates for use in the SAML environment, you can generate a Java keystore and self-signed certificates using tools from the Java Runtime Environment or by using free utilities available online, for example, KeyStore Explorer. You can find the steps for generating certificates and keystores online. Because using certificates is central to maintaining trust and security in a SAML environment, you must securely manage the certificates, keystores, and their passwords.

Enable signed SAML authentication requests

To enable the signing of SAML authentication requests made from Security Services to the configured SAML Identity Provider:

1. Open the Security Services **federation.properties** file.

2. Set the **tcsso.saml.sign_authn_request** property to **true**:

```
tcsso.saml.sign_authn_request=true
```

3. Configure the signing keystore and signing key properties, along with their password properties, to point to a valid keystore and key.

```
tcsso.saml.signing_private_jks_file=</secure/path/sp_signing_key.jks>
tcsso.saml.signing_private_jks_file_pwd=<secret>
tcsso.saml.signing_private_key_name=<key name>
tcsso.saml.signing_private_key_pwd=<secret>
```

SAML service provider metadata file generation

SAML uses XML metadata files to exchange information that establish trust between SAML-relying parties and includes configuration settings that are specific to each party.

Although not necessary, the metadata can simplify configuration between relying parties. By consuming the metadata of the relying party, a SAML provider can automatically configure a trusted relationship based on its contents. Additionally, you can update and maintain the configuration through the automatic reloading of metadata changes done at the corresponding relying party.

Security Services provides the ability to generate a metadata file for its native SAML service provider feature. The generated metadata file includes the Security Services entity ID, the assertion consumer service URL, the public certificate used for verifying authentication request signatures, and the public certificate used for encryption, along with other default SAML service provider settings.

1. Before generating the XML metadata file, configure the Security Services **federation.properties** file, as described in *Configure Security Services SAML federation properties*.

2. Generate the Login Service WAR file and deploy it.

3. Generate the Security Services service provider metadata file by accessing the **saml_sp_metadata** endpoint of the Login Service URL with a browser. For example:

   ```
   https://host.domain:port/loginservice_war_context/saml_sp_metadata
   ```

   Accessing this endpoint dynamically generates a new metadata XML file with each request. To make changes to the metadata, deploy the Login Service WAR file containing the changed **federation.properties** file and reload this endpoint in a browser to get the updated SAML metadata XML file.

SAML authentication process flow example

The following example illustrates the user authentication process flow between Security Services and the SAML IdP:

1. A user enters the credentials to access a Teamcenter application.

2. Teamcenter directs the user to the Security Services Login Service. This service checks the federation configuration and redirects the user to the configured SAML IdP URL for authentication.

3. The user authenticates with the SAML IdP and returns the response with a SAML assertion to the Login Service.

4. The Login Service validates the SAML assertion from the IdP, retrieves the user ID from the assertion, establishes the user session, and issues a Teamcenter SSO token. The user is then returned to the Teamcenter application and can log on to Teamcenter using the SSO token.

Configure Security Services SAML federation properties

Settings specific to the SAML configuration are included in the **federation.properties** file. Sets include:

Property	Definition
tcsso.saml.issuer_id	Specifies a unique string that identifies Security Services as a SAML SP and must be provided to the SAML IdP.
tcsso.saml.idp_public_key_file	Specifies the file location of the certificate containing the public key of the SAML IdP.
tcsso.saml.decryption_private_jks_file	Specifies the file location of a JKS-formatted keystore containing the private key used to decrypt a SAML Assertion coming from the SAML IdP.
tcsso.saml.decryption_private_jks_file_pwd	Specifies the password for the JKS keystore file. Although it can be entered as text, encoding of the password is recommended. Refer to *Password Encoding* for instructions on how to encode this password.
tcsso.saml.decryption_private_key_name	Specifies the name of the private key in the keystore used to decrypt the SAML Assertion.
tcsso.saml.decryption_private_key_pwd	Specifies the password for the private key. Although it can be entered as text, encoding of the password is recommended. Refer to *Password Encoding* for instructions on how to encode this password.
tcsso.saml.sign_authn_request	Specifies whether the authentication request made to the SAML IdP must be encrypted. When set to **true**, the signing private keystore, private key, and accompanying passwords must be configured correctly.

Property	Definition
tcsso.saml.signing_private_jks_file	Specifies the file location of a JKS-formatted keystore containing the private key used to sign SAML2 authentication requests sent to the SAML IdP.
tcsso.saml.signing_private_jks_file_pwd	Specifies the password for the signing JKS keystore file. Although it can be entered as text, encoding of the password is recommended. Refer to *Password Encoding* for instructions on how to encode this password.
tcsso.saml.signing_private_key_name	Specifies the name of the private key in the keystore used to sign the SAML authentication requests sent to the SAML IdP.
tcsso.saml.signing_private_key_pwd	Specifies the password for the signing private key. Although it can be entered as text, encoding of the password is recommended. Refer to *Password Encoding* for instructions on how to encode this password.
tcsso.saml.userid_attribute_name	Specifies the SAML name attribute containing the trusted Teamcenter user ID.
tcsso.saml.authn_context_class	Specifies the desired authentication method to be used at the SAML IdP. This enables the SAML SP to select a preferred authentication method at the IdP when it is available. Although requested, it does not guarantee the SAML IdP honors the request. The default value is a form-based logon method: `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`

Property values shown inside brackets (< >) have no default values and must be changed if the corresponding federation type will be enabled. Values not inside brackets are set to default values that can remain unchanged; however, they must still be verified for each deployment.

> Note:
>
> You must select the **SAML** federation type even if the SAML federation properties are set correctly. If you have not yet configured the Login Service to use SAML federation, see the details in *Configure your Login Service for federation mode*.

Adding SameSite=None attribute to session cookie

Starting in version Chrome 80 and eventually in browsers leveraging Chromium, cookies that do not specify the **SameSite** attribute will be treated as if they were set to **SameSite=Lax**. The **SameSite** attribute declares how cookies should be restricted to a same-site context. When set to **Lax**, the cookie is only sent to same-site requests or top-level navigation. However, certain Security Services environments require these cookies to be preserved in the third-party context to keep users properly signed in during their session. The main situation where this comes into play for Security Services is when authentication is federated to some third party as in the following case:

- Login is initiated at the Security Services server by browser and the Security Services server sets session cookies in the browser.
- Login sequence is redirected to a third party (Identity Provider) to authenticate the user.
- The third party sends authentication data back to the Security Services server via the browser.

  This browser request needs to contain the original session cookies set by the Security Services server.

Currently, this restriction affects Security Services configurations with **federation_type=SAML**.

SAM Auth and OIDC configurations (both OAuth-based) have login sequence considerations that are similar to SAML; however, there is one main difference. When the identity provider sends authentication data back to the Security Services server through the browser in the SAML flow, it sends it via a POST-type request. In the OAuth-based flows, this data is sent via a GET-type request. Browsers do not currently have the SameSite restrictions for GET-type requests.

For the configurations that are affected, the Security Services server session cookies require the following:

- Set the **SameSite** attribute to **None**.
- Set the **Secure** flag

> Note:
>
> Because the **Secure** flag is required, affected Security Services environments will only work if the Security Services Login Service is accessed through an HTTPS connection.

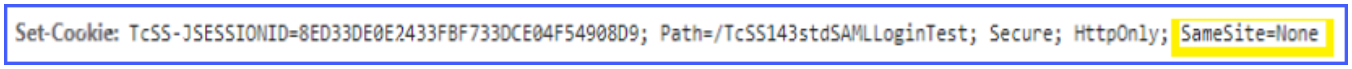
For details on how to set this cookie attribute on supported Teamcenter application servers, refer to the following table:

For this Application Server	Do this
Wildfly (19+)	In the Login Service staging location, create the **WEB-INF\undertow-handlers.conf** file with the following content:

For this Application Server	Do this
	`samesite-cookie(mode=None)`

Tomcat (9.0.29+)	In the Login Service staging location, create the **META-INF\context.xml** file with the following content:
	`<Context>`
	`<CookieProcessor sameSiteCookies="none" />`
	`</Context>`

Check if **SameSite=None** is set on session cookie:



Using Security and Account Manager Authentication (SAM Auth)

Authenticating Teamcenter users using SAM Auth

Security Services supports federated authentication using the Security and Access Management Authentication (SAM Auth) service through the OpenID Connect (OIDC) protocol. In this integration, SAM Auth takes the role of the OIDC IdP and Security Services is the relying party (RP).

Use the **TCSSO_SAM_AUTH_ZONE** environment variable to specify a **zone** value. When the **zone** parameter value specifies a legitimate IdP zone registered with SAM Auth, SAM Auth utilizes the specified IdP to authenticate users.

For more information, see the *Configuring custom IdPs* topic in the *Security and Access Management Authentication* deliverable in the Digital Engineering Services documentation site.

Federated users

With SAM Auth support, Security Services is configured with a SAM Auth IdP that supplies users provisioned by external systems for access to Teamcenter. This enables users that originate in Webkey/ Security and Access Management (SAM) to seamlessly authenticate with Teamcenter.

User mapping

Federated user IDs returned by SAM Auth are expected to match user IDs in Teamcenter. User IDs that do not match can be mapped to Teamcenter user IDs using the supplied ApacheDS LDAP. (See *Example 1 - Defining multiple attributes* and *Example 2 - Defining a single shared Teamcenter attribute*.)

SAM Auth login flow example

The following example illustrates the user authentication process flow between Security Services and SAM Auth:

1. A user attempts to access a Teamcenter application.

2. Because the user is not yet authenticated, Teamcenter directs the user to the Security Services Login Service, which checks the federation configuration and redirects the user to the configured SAM Auth service, which redirects the user to the Webkey logon page.

3. The user authenticates using Webkey, which then returns the response to SAM Auth. At this point, SAM Auth returns the response to the Login Service with a one-time authorization code.

4. The Login Service sends a request directly to the SAM Auth server with the one-time authorization code and receives an ID Token.

5. The Login Service gets the authenticated user's user ID from the ID Token and establishes the user session and issues a token. The user is then returned to the Teamcenter application, logged in, and ready to use the application.

Register the Login Service with SAM Auth

Because SAM Auth uses the OIDC/OAuth2 protocols, it requires that you register RP applications to authenticate with it. The Security Services Login Service is registered with SAM Auth. There are two URI types that must be included in this application registration:

- Redirect URIs

  A Login Service configured for SAM Auth authentication exposes an endpoint to which SAM Auth sends the authorization code response. As part of the OIDC/OAuth2 protocol, you must add this endpoint URI to the application registration's list of redirect URIs. The endpoint has the following format:

  https://<LoginServiceHost>:<Port>/<LoginServiceName>/weblogin/sam_auth_callback

- Post-logout redirect URIs

  When you log out of a client application configured to authenticate using Security Services with SAM Auth, SAM Auth attempts to redirect the browser back to the client application once the logout completes. As part of the OIDC/OAuth2 protocol, you must add these endpoint URIs to the application registration's list of Post-Logout Redirect (otherwise known as Logout) URIs. The endpoints are the URI of the client application. For example, if you configure for Active Workspace client, add the main AWC URI:

  https://<AWC Host>:<Port>/<AWC App Name>/

> Note:
>
> These post-logout redirect URIs are only registered for Teamcenter thin clients, such as Active Workspace.

Configure Security Services SAM Auth federation properties

Settings specific to the SAM Auth configuration are included in the **federation.properties** file. Settings include:

Property	Definition
tcsso.samauth_client_id	Specifies the SAM Auth client ID, which is obtained when the Login Service is registered as a client application.
tcsso.samauth_client_secret	Specifies the SAM Auth client secret, which is obtained when the Login Service is registered as a client application. Although it can be entered here in clear text, encoding this password is recommended. Refer to *Password Encoding* for instructions on how to encode this password.
tcsso.samauth_auth_endpoint	Specifies the SAM Auth server authorization_endpoint.
tcsso.samauth_token_endpoint	Specifies the SAM Auth server token_endpoint.
tcsso.samauth_jwks_endpoint	Specifies the SAM Auth server **jwks_uri**.
tcsso.samauth_userid_claim	Specifies the SAM Auth Token claim that is used as the SSO user ID.

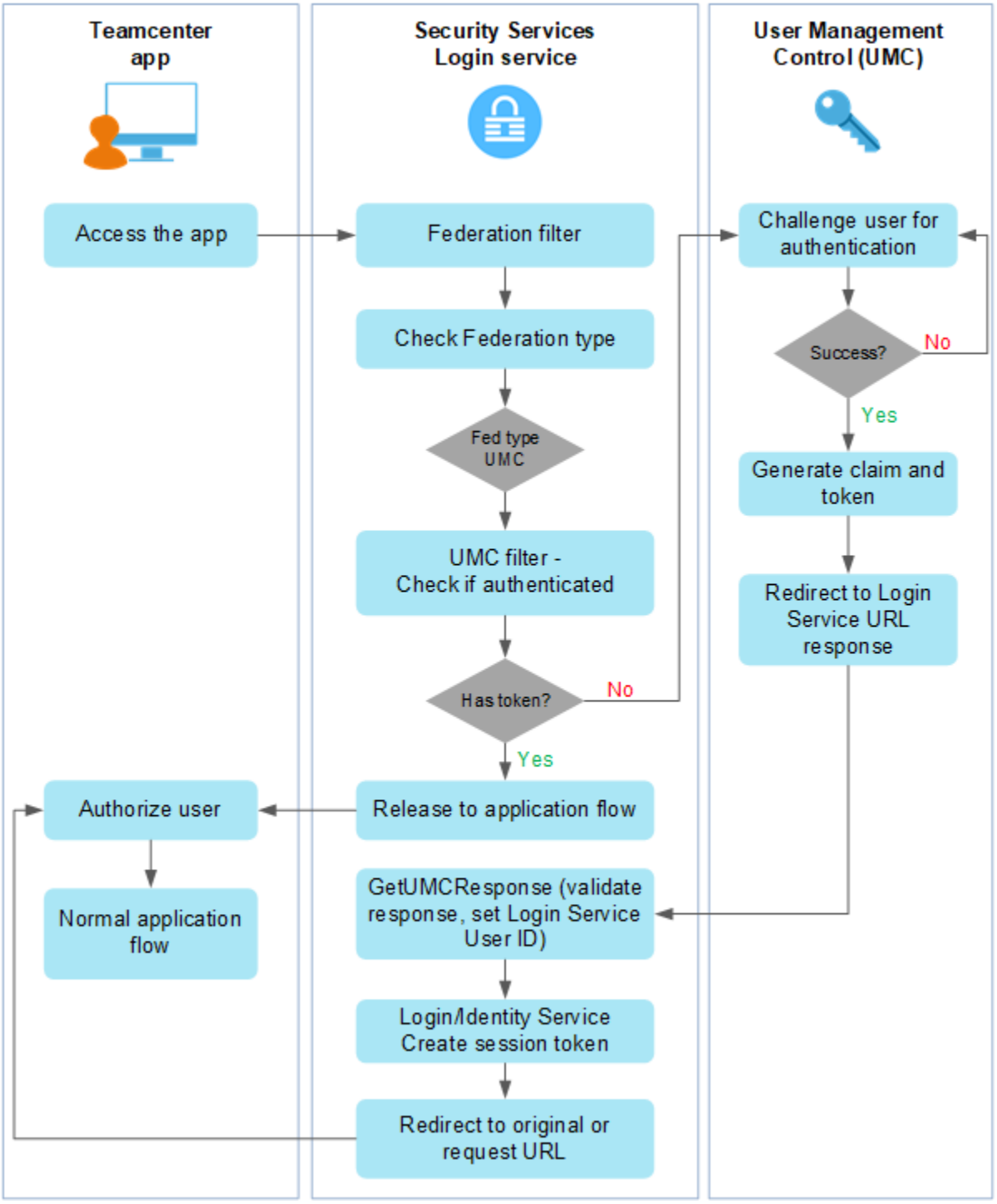
Property values shown inside brackets (< >) have no default values and must be changed if the corresponding federation type will be enabled. Values not inside brackets are set to default values that can remain unchanged; however, they must still be verified for each deployment.

> Note:
>
> You must select the **SAMAUTH** federation type even if the SAM Auth federation properties are set correctly. If you have not yet configured the Login Service to use SAMAUTH federation, see the details in *Configure your Login Service for federation mode*.

Using User Management Control authentication

Implementing IT security federation protocol using Siemens User Management Control (UMC) software

Security Services supports Federation, a technology implementation of IT security federation protocol, using Siemens User Management Control (UMC) software. This protocol treats UMC as the federation identity provider (user authentication) and Security Services as the service provider (user authentication). UMC allows users to authenticate with full access to both UMC and Security Services systems without needing to reauthenticate.



The following example illustrates the user authentication process flow between Security Services and UMC:

1. A user attempts to access a Teamcenter application.

2. Because the user is not yet authenticated, Teamcenter directs the user to the Security Services Login Service, which checks the federation configuration and redirects the user to the UMC logon page.

3. The user authenticates using UMC, which then returns the response to Login Service.

4. The Login Service establishes the user session and issues a token. The user is then returned to the Teamcenter application and is logged in and ready to use the application.

Configure Security Services for federation using UMC

The Security Services Login Service supports cross-origin resource sharing (CORS), which provides the ability to specify a set list of hosts and servers to accept JavaScript and HTML traffic. CORS support protects from cross-site scripting attacks and only allows traffic from defined hosts. Therefore, if a list is not defined, traffic from within the same domain is allowed.

The list of hosts is set using the Teamcenter Web Application Manager:

1. Launch the Web Application Manager application.

2. Select the Login Service application and click **Modify**.

3. Click **Modify Context Parameters**.

4. Select the **tcsso.cors_whitelist** parameter. When set, the list of domains to be whitelisted is honored for cross-origin resource sharing (CORS) support (*blank*).

5. Enter one or more hosts or servers using the fully qualified domain name and protocol.

   Example:

   For a single host and server, enter:

   `https://myserver.mydomain.com`

   If you have multiple hosts and servers, enter them as a comma-separated list:

   `https://myserver.mydomain.com, https://myserver2.mydomain.com`

Use Security Services/UMC federation public key configuration

The Security Services Login Service requires the UMC public key before it can process UMC authentication responses. UMC responses to user authentication requests are encrypted and require the key to verify the signature and validity of the response.

1. Navigate to the machine and UMC directory where UMC is installed. The UMC public key is located under the following directory:

   ```
   C:\ProgramData\Siemens\UserManagement\Cert\CLAIM
   ```

2. Copy the **key.pub** file to the machine's **WEB-INF** directory where the Login Service is installed, under the Web Application Manager. For example:

   ```
   C:\Users\username\Web_Root\LoginService\webapp_root\WEB-INF
   ```

3. Rename the file to **umc_fed_key.pub**.

4. If you have not yet configured the Login Service to use UMC federation, see the details in *Configure your Login Service for federation mode*.

5. Generate a new Login Service WAR file using the Web Application Manager and deploy this to your application servers.

Configure your Login Service for UMC federation mode

Using UMC authentication requires that you must configure your Login Service for federation mode as described in *Configure your Login Service for federation mode*.

Additionally, UMC integration requires that you configure your Login Service in gateway mode using the following context parameters:

Context parameter name	Value
tcsso.behind_sso_gateway	**true**
tcsso.gateway.field.type	**parameter**
tcsso.gateway.field.name	**umcUserName**

> Note:
>
> UMC configuration does not require any changes to the **federation.properties** file.

Using OpenID Connect authentication

Authenticating Teamcenter users using OpenID Connect authentication

Security Services supports federated authentication using OpenID Connect (OIDC) protocol. In this integration, the OpenID Provider (OP) takes the role of the Identity Provider and Security Services is the relying party.

Federated users

With OIDC support, Security Services is configured with an OP that supplies users provisioned by external systems for access to Teamcenter. This enables users originating in the OP to seamlessly authenticate with Teamcenter.

User mapping

Federated user IDs returned by the OP are expected to match user IDs in Teamcenter. For user IDs that do not match, use the supplied ApacheDS LDAP to map them to Teamcenter user IDs. (See *Example 1 - Defining multiple attributes* and *Example 2 - Defining a single shared Teamcenter attribute*.)

> Note:
>
> To enable this user ID mapping in OIDC mode, set the **GatewayAliasingEnabled** content parameter in the Identity Service to **true**.

OIDC logon flow example

The following example illustrates the user authentication process flow between Security Services and the OP:

1. A user attempts to access a Teamcenter application.

2. Because the user is not yet authenticated, Teamcenter directs the user to the Security Services Login Service, which checks the federation configuration and redirects the user (browser) to the configured OP.

3. The user authenticates using whatever method is configured at the OP, which then returns the response to the Login Service with a one-time authorization code.

4. The Login Service sends a back-channel request directly to the OP with the one-time authorization code and receives an ID Token.

5. The Login Service gets the authenticated user's user ID from the ID Token, establishes the user session, and issues a token.

   The user is then returned to the Teamcenter application, logged in, and ready to use the application.

Register the Login Service with the OpenID provider

According to the OIDC/OAuth2 protocols, it is required that you register the relying party applications with the OpenID Provider. In this case, you must register the Security Services Login Service with the OP. You must include the following two Uniform Resource Identifier (URI) types in this application registration:

- Redirect URIs

  A Login Service configured for OIDC authentication exposes an endpoint to which the OP sends the authorization code response. As part of the OIDC/OAuth2 protocol, you must add this endpoint URI to the application registration's list of redirect URIs. The endpoint has the following format:

  `https://`*LoginServiceHost*`:`*Port*`/`*LoginServiceName*`/weblogin/oidc_callback`

- Post-logout redirect URIs

  When you log out of a client application configured to authenticate using Security Services with OIDC, the OP attempts to redirect the browser back to the client application once the logout completes. As part of the OIDC/OAuth2 protocol, you must add these endpoint URIs to the application registration's list of post-logout redirect URIs. The endpoints are the URI of the client application. For example, if you configure for Active Workspace client, add the main AWC URI:

  `https://`*AWC-Host*`:`*Port*`/`*AWC-App-Name*`/`

> Note:
>
> These post-logout redirect URIs are only registered for Teamcenter thin clients, such as Active Workspace.

Configure Security Services OIDC federation properties

The following OIDC configuration settings are included in the **federation.properties** file.

Property	Definition
tcsso.oidc.client_id	Specifies the OIDC client ID, which is obtained when the Login Service is registered as a client application.
tcsso.oidc.client_secret	Specifies the OIDC client secret, which is obtained when the Login Service is registered as a client application. If it is registered with **client_auth_method=client_secret_basic**, see the **tcsso.oidc.client_auth_method** property definition in this table. Note: Although you can enter the password here in clear text, Siemens Digital Industries Software recommends that you encode this password. Refer to *Password Encoding* for instructions on how to encode this password.
tcsso.oidc.auth_endpoint	Specifies the OP authorization endpoint.
tcsso.oidc.token_endpoint	Specifies the OP token endpoint.

Property	Definition
tcsso.oidc.jwks_endpoint	Specifies the OP JSON Web Key Set (JWKS) URI.
tcsso.oidc.userid_claim	Specifies the OIDC token claim, which is used as the Security Services user ID.
tcsso.oidc.scope	Specifies the value of the scope parameter, which is included in the OIDC authentication request.
tcsso.oidc.token_sig_alg	Specifies the signature algorithm used to validate the signature of the signed OIDC ID token.
tcsso.oidc.display	Specifies the value of the display parameter, which is included in the OIDC authentication request. The OP determines which values are supported.
tcsso.oidc.client_auth_method	Specifies the authentication method, which is used when sending a token request to the OP.

The following **tcsso.oidc.signing** and **tcsso.oidc.jwt** properties are only required if the **private_key_jwt** Token Endpoint authentication method (**tsso.oidc.client_auth_method**) is selected above.

Property	Definition
tcsso.oidc.signing_jks_file	Specifies the path to a Java KeyStore (JKS) formatted keystore. This contains the public key that should be passed to the OP to validate the signed JSON Web Token (JWT) sent with a token request, as well as the corresponding private key used to sign the JWT.
tcsso.oidc.signing_jks_file_pwd	Specifies the password for the JKS file.
tcsso.oidc.signing_private_key_name	Specifies the name (alias) of the desired signing private key stored in the JKS file.
tcsso.oidc.signing_private_key_pwd	Specifies the password for the desired signing private key stored in the JKS file.
tcsso.oidc.jwt.sig_alg	Specifies the signature algorithm used to sign the JWT sent to the OP.
tcsso.oidc.jwt.expiration	Specifies the expiration time (in seconds) of the JWT sent to the OP.

The following **tcsso.oidc.encryption** properties are only required if ID Token encryption is supported and enabled at the OP.

Property	Description
tcsso.oidc.encryption_jks_file	Specifies the path to a JKS-formatted keystore containing the public key that should be passed to the OP to encrypt

Property	Description
	the OIDC ID Token, as well as the corresponding private key that is used to decrypt the ID Token.
tcsso.oidc.encryption_jks_file_pwd	Specifies the password for the JKS file.
tcsso.oidc.encryption_private_key_name	Specifies the name (alias) of the desired decryption private key stored in the JKS file.
tcsso.oidc.encryption_private_key_pwd	Specifies the password for the desired decryption private key stored in the JKS file.
tcsso.oidc.enable_jwks_uri	Specifies, when set to **true**, whether a **jwks_uri** is exposed at the *LoginService***/certs** endpoint. Public keys that are configured above (either with **tcsso.oidc**.signing or with **tcsso.oidc**.encryption) are converted to JWK format and returned from that endpoint for retrieval by the OP.

Note:

- Property values shown in italic have no default values. You must change these values if you want to enable the corresponding federation type. Values not shown in italic are set to default values that can remain unchanged; however, you must still verify the values for each deployment.

- You must select the **OIDC** federation type even if the OIDC federation properties are set correctly. If you have not yet configured the Login Service to use OIDC federation, see the details in *Configure your Login Service for federation mode*.

9. Troubleshooting

General considerations

Unauthorized user error

Behavior	The system displays the following error message after you enter a valid user name and password: `User is not authorized`
Platforms and products	All.
Problem	This indicates that the user's LDAP entry in the LDAP server does not contain the attribute corresponding to the application being launched. The name of this attribute is specified in the Application Attribute Mappings table associated with the Login Service.
Solution	Check the user's LDAP configuration information. Ensure the user has the Login Service's attribute set, since all Teamcenter users must have this value set.
Versions	All.

Unable to log on with valid credentials

Behavior	You receive a **Password is incorrect** or other error message after you submit valid credentials using the logon window.
Platforms and products	All platforms.
Problem	The LDAP settings are incorrect or the Login Input Definitions table is incorrectly specified.
Solution	Make certain the Login Input Definitions table has not been modified. Unless you customized the identify provider, the table should not be modified. The **idpkey** values should be **User**, **Password**, and **UserLocale**. These values do not correspond to LDAP server attribute names. Next, check the LDAP settings, in particular the value for **QueryDN**, **QueryDNPassword**, and **BaseDN**.
Versions	All.

SSOException from client library

Behavior	The rich client returns an **SSOException** exception without any additional message.

Platforms and products	Rich client, all platforms.
Problem	The **USER** environment variable is not set in the shell from which the rich client is being launched.
Solution	Set the **USER** environment variable.
Versions	All.

SSOException from client library indicating missing argument

Behavior	The rich client returns an **SSOException** exception with a missing argument message.
Platforms and products	Rich client, all platforms.
Problem	The application ID is not configured on the client.
Solution	Set the application ID according to the Teamcenter product documentation.
Versions	All.

Logons disabled by system administrator

Behavior	When logging on, the following error message appears: `Logins have been disabled by the system administrator.`
Platforms and products	All.
Problem	This either means the directory server is simply not running or it indicates the Identity Service's LDAP entries (**LDAPHosts**, **LDAPPortNo**, **QueryDN**, and **QueryDNPassword**) are not set correctly.
Solution	Make certain that the directory server is running, that logons have not been disabled, and that the LDAP configuration settings for the Identity Service are correct.
Versions	All.

Simpgen translator does not support Security Services

Behavior	When Teamcenter is installed in the Security Services enabled mode, the Simpgen translator may fail because access is denied.
Platforms and products	All.

Problem When Teamcenter is installed in the Security Services enabled mode, the Simpgen translator may fail because access is denied.

Solution Disable Security Services for the Simpgen translator. You can enable Security Services for all other components; however, you must install the Simpgen translator on a separate machine that is not Security Services enabled.

If you want all components on the same machine, associate a new *TC_DATA* directory with the Simpgen translator that is not Security Services enabled. To make the Simpgen translator work in Security Services enabled mode:

1. Duplicate **$TC_DATA** or **%TC_DATA%** by creating the **${TC_DATA}_nonsso** or **%TC_DATA%_nonsso** (depending on the platform) directory.

2. Unset all Security Services related variables in the **${TC_DATA}_nonsso** or **%TC_DATA%_nonsso** directory. The three variables that must be removed or reset are **TC_SSO_APP_ID**, **TC_SSO_SERVICE**, and **TC_SSO_LOGIN_URL**.

   - Windows example:

     ```
     rem set TC_SSO_APP_ID=Tc8S17
     rem set TC_SSO_SERVICE=http://
     svli6011v03.net.plm.eds.com:7105/ssoService8
     rem set TC_SSO_LOGIN_URL=http://
     svli6011v03.net.plm.eds.com:7105/ssoLogin8
     ```

   - Linux example:

     ```
     #TC_SSO_APP_ID=Tc8S19; export TC_SSO_APP_ID
     #TC_SSO_SERVICE=http://
     svli6011v03.net.plm.eds.com:7105/ssoService8;
     export TC_SSO_SERVICE
     #TC_SSO_LOGIN_URL=http://
     svli6011v03.net.plm.eds.com:7105/ssoLogin8;
     #export TC_SSO_LOGIN_URL
     ```

3. Set the *TC_DATA* variable in the **Module\Translators\simpgen\simpgen.bat** file or the **Module\Translators\simpgen\simpgen.sh** file and point it to the newly created **${TC_DATA}_nonsso** (Linux) or **%TC_DATA%_nonsso** (Windows) directory.

Versions All.

Third-party configuration considerations

Application server considerations

Web application servers

The Web Application Manager produces two WAR files, one for the Login Service and one for the Identity Service. With most web servers, you can deploy these WAR files by using an administration console or by dropping the WAR files into a **webapps** folder.

Sun Java System Application Server

After installing the Sun Java System Application Server, edit the **server.policy** file (**... AppServer\domains\domain1\config\server.policy**):

1. Replace line permission **java.net.SocketPermission** with:

   ```
   permission java.net.SocketPermission    "*", "connect,accept,listen";
   ```

2. Replace line permission **java.util.PropertyPermission** with:

   ```
   permission java.util.PropertyPermission "*", "read,write"
   ```

3. Add the following lines:

   ```
   permission javax.security.auth.AuthPermission "modifyPrincipals";
   permission javax.security.auth.AuthPermission "getSubject";
   ```

4. Stop and restart the server.

IBM WebSphere Application Server

Cookie path redefinition

If there are multiple applications deployed in the same instance of an application server, the HTTP session cookie may be overwritten as the browser connects to different applications. This may cause unexpected and inconsistent behavior during authentication and startup of Teamcenter applications.

How to Work Around or Avoid

Implement the following procedure to ensure that multiple applications have their cookie path set so they do not overwrite each other's session.

1. In the WebSphere administration console, choose **Applications→Application Types→WebSphere enterprise applications**.

2. Click your application name, for example, select **Login Service**.

3. Choose **Web Module Properties→Session Management**.

4. In the **General Properties** section, select the **Override session management** check box.

5. Click the **Enable cookies** link and set **Cookie Path** to the context root, for example, **/tcLoginService**.

6. Repeat step 2 through step 5 for the Identity Service.

For more information, see your IBM WebSphere documentation.

IBM WebSEAL

Redirect to application web client can fail when using WebSEAL

WebSEAL removes base **href** information when converting relative URLs into absolute URLs.

How to Work Around or Avoid

You must configure WebSEAL to preserve the base tag information by setting the **preserve-base-href** preference to **yes**.

```
#----------------------
# Filtering
#----------------------
# If preserve-base-href is no, then WebSEAL will remove all BASE HREF
tags
# from filtered HTML documents and prepend the base tag to filtered
links.
# Otherwise, the BASE HREF tag will be filtered.
preserve-base-href = yes
```

For more information, see your IBM WebSEAL documentation.

Shibboleth considerations

When you configure the Shibboleth SAML service provider as an authenticating proxy for Security Services, it maintains the authenticated session for Security Services client applications. To prevent session hijacking, it is responsible for monitoring incoming requests associated with an authenticated session to ensure that it came from the same client. It does so by validating that the originating client address is consistent between requests. When the Shibboleth service provider is deployed behind a load balancer, it is common for a load balancer to mask the originating client addresses. It is also common for load balancers to change the masked client addresses as needed for its own purposes. This changing of the client address will cause the Shibboleth service provider to reject requests made on an authenticated

session and result in a Security Services Session Agent authentication failure. A common error message for this failure:

```
com.teamcenter.ss.SSOInvalidSessionException: Expected methodResponse
element, got html
```

To resolve this failure, update the Shibboleth configuration file (**C:\opt\shibboleth-sp\etc\shibboleth\shibboleth2.xml**) with the following setting:

```
<Sessions lifetime="28800" timeout="3600" relayState="ss:mem"
     checkAddress="false" consistentAddress=”false” handlerSSL="true"
cookieProps="https">
```

Sun Java System Web Server

Teamcenter Security Services application tokens are passed as cookies by some Teamcenter applications. When those applications are deployed on the Sun Java System Web Server, one of its default settings corrupts those tokens, preventing their successful validation and completion of logon.

How to Work Around or Avoid

Create a file, **sun-web.xml**, within the **WEB-INF** folder of each Teamcenter application to be deployed on the Sun Java System Web Server containing:

```
<sun-web-app>
      <property name="encodeCookies" value="false"/>
</sun-web-app>
```

JBoss EAP requires changes to enable gateway authentication

Beginning with Enterprise Application Platform (EAP) 7.1, JBoss updated the embedded servlet container web component from **JBossWeb** to the new JBoss **Undertow** web component. This update changed the authentication architecture for Apache JServ Protocol (AJP) connected configurations used by the Teamcenter Security Services gateway mode. To enable authentication by an external authenticating proxy, the previous **tomcatauthentication=false** AJP connector setting is no longer valid. Instead, perform the following required steps to enable gateway authentication and allow Teamcenter Security Services to access the **remote_user** variable from the gateway.

Teamcenter Security Services Login Service changes

When using JBoss Enterprise Application Platform (EAP):

1. Set your Teamcenter Security Services Login Service context parameter gateway settings as follows:

```
tcsso.behind_sso_gateway=true
```

```
tcsso.gateway.field.type=remote_user
tcsso.gateway.field.name=Proxy-Remote-User
```

2. Add a new file, **jboss-web.xml**, to Teamcenter Security Services Login Service WAR file under the **WEB-INF** folder with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
     <security-domain>ExternalSecurityDomain</security-domain>
</jboss-web>
```

3. In the Teamcenter Security Services Login Service WAR file, update the **web.xml** file, and under the **WEB-INF** folder, add the following text anywhere within the **web-app** section:

```
<login-config>
     <auth-method>EXTERNAL</auth-method>
</login-config>
```

JBoss changes

In the JBoss **standalone.xml** configuration file, add the following text in the **security-domain** section within the **subsystem xmlns="urn:jboss:domain:security:2.0"** section:

```
<security-domain name="ExternalSecurityDomain" cache-type="default">
     <authentication>
          <login-module code="Client" flag="optional"/>
     </authentication>
</security-domain>
```

Kerberos considerations

Teamcenter service IP address must resolve into a fully qualified domain name

For Kerberos authentication to work, a Teamcenter service (using Kerberos) IP address must resolve into a fully qualified domain name (FQDN). For example:

```
74.125.236.48 → maa03s04–in-f16.1e100.net
```

Use one of the following ways to resolve the IP address into FQDN:

- In the DNS server, add an IP address to the FQDN entry.

- In the hosts file, add an IP address to FQDN entry. For example, in Windows, the hosts file location is:

  ```
  C:\Windows\System32\drivers\etc\hosts
  ```

Configuring encryption types for Kerberos

When configuring TCCS to use Kerberos authentication, a **krb5.ini** file should not be specified in the TCCS configuration. Even if specified, the file is not used for Kerberos configuration.

Java applets do not support Kerberos

The deprecated Java applets only support Basic, Digest, and NT LAN Manager (NTLM) authentication. They do not support Kerberos authentication. To support the use of the Java applets, you must enable either Basic, Digest, or NTLM, along with Kerberos in your web server.

Kerberos authentication

When forward proxy is in use, web browsers do not support Kerberos authentication.

Port number in Kerberos Service Principal Name is not supported

Inclusion of the port number in Kerberos Service Principal Name (SPN) is not supported. An example of a supported SPN is:

HTTP/<*WEB_SERVER_FQDN*>

A. ApacheDS installation and configuration

Install and launch ApacheDS

> Note:
>
> The ApacheDS LDAP server is part of your Security Services installation.

1. Extract the **TcSecurityServices***version***.zip** file to a local directory.

2. Copy the contents of the **TcSecurityServices***version***\TcSecurity\ApacheDS** directory to **C:\apps\ApacheDS** on Windows systems, or choose an appropriate path on Linux systems.

3. Remove read-only permission for all files and directories under the **ApacheDS** directory.

   - Windows systems:

     a. Right-click the **ApacheDS** folder, choose **Properties**, clear the **Read-only** check box in the **Attributes** section, and click **OK**.

     b. Click **OK** to confirm the changes are applied to the subfolders and files.

   - Linux systems:

     - Remove read-only permissions for the **ApacheDS** directory and subdirectories.

4. Launch ApacheDS from a command prompt:

   - Windows systems:

     a. Change to the **C:\apps\ApacheDS***ApacheDS-version***\bin** directory.

     b. Run the **apacheds.bat** file.

   - Linux systems:

     a. Change to the *ApacheDS-path***/bin** directory.

     b. Type the following command:

        ```
        sh bin/apache.sh
        ```

Note:

Alternatively, you can launch ApacheDS automatically as a Windows service or /Linux daemon.

- Windows systems:

  a. Change to the **C:\apps\ApacheDS***ApacheDS-version***\bin** directory.

  b. Type the following command:

  ```
  wrapper -i ..\instances\default\conf\wrapper.conf
  set.INSTANCE_DIRECTORY=..\instances\default
  set.INSTANCE=default
  ```

  where the `set.INSTANCE` switch sets the instance name used in the Windows service and the `set.INSTANCE_DIRECTORY` switch forces the read of the default instance to pick up the specific Teamcenter password policies.

- Linux systems:

  a. Change to the *ApacheDS-path***/bin** directory.

  b. Start ApacheDS using the **sudo** command:

  ```
  sudo apacheds start default
  ```

ApacheDS configuration

Install Directory Studio

Note:

Apache Directory Studio is a graphical user interface client used to configure ApacheDS.

1. Install a current version of Java.

2. Select the installation executable found in the Teamcenter Rich Client install kit under the **additional_applications/ApacheDirectoryStudio** directory:

   `ApacheDirectoryStudio-`*platform-version*`.exe`

   where *platform-version* is the version of Apache Directory Studio.

3. Follow the steps to install. You are prompted to select:

- The installation directory.
- The location of the Java home directory.

Connect to ApacheDS

Configure Directory Studio to connect to the ApacheDS LDAP server.

1. Run Apache Directory Studio.
2. From the LDAP file menu, create a connection to ApacheDS.
3. In the **New LDAP Connection** dialog box:
   a. Enter the following network parameters:
      - A unique connection name.
      - Host name: **localhost**
      - Port: **10389**
   b. Enter the following authentication parameters:
      - Authentication method: **Simple Authentication**
      - Bind DN or user: **uid=admin,ou=system**
      - Bind password: **secret**
   c. Click **Finish**.

Create a partition in ApacheDS

You must create a partition in ApacheDS to match the domain with which to be authenticated. An object class index is required as part of creating a partition.

1. Create a partition:
   a. Under **ou=config, ads-directoryServiceId=default**, right-click **ou=partitions**.
   b. Choose **New→New Entry**.
   c. Select **Create entry from scratch** and click **Next**.

d. Select and add the **ads-jdbmPartition** object class and click **Next**.

e. Select RDN **ads-partitionId** and enter **domain** or your own required domain name and click **Next**.

f. In the DN editor, enter **dc=domain,dc=com** using your own required domain name and click **OK**.

g. Click **Finish**.

2. Add an index to the partition.

a. Under **ou=config, ou=partitions**, right-click **ads-partitionId=domain**.

b. Choose **New→New Entry**.

c. Select **Create entry from scratch** and click **Next**.

d. Select and add the **organizationalUnit** object class and click **Next**.

e. Select RDN **ou** and enter **indexes** and click **Next**.

f. Click **Finish**.

g. Under **ads-partitionId=domain**, right-click **ou=indexes**.

h. Choose **New→New Entry**.

i. Select **Create entry from scratch** and click **Next**.

j. Select and add the **ads-jbdmindex** object class and click **Next**.

k. Select RDN **ads-indexAttributeId**, enter **objectClass**, and click **Next**.

l. When prompted for additional **ads-indexHasReverse** attributes, enter **true**, and click **Finish**.

3. Restart ApacheDS.

4. Right-click **Root DSE** and choose **New Context Entry**.

5. Select **Create entry from scratch** and click **Next**.

6. Select and add the **domain** object class and click **Next**.

7. When prompted for the distinguished name, enter **dc=domain,dc=com** using your own required domain name. Click **Next** and then click **Finish**.

Add users

To add users, create an organizational unit under the partition and then add users to the organizational unit.

1. Create an organizational unit (**ou=staff** in **dc=domain, dc=com**).

    a. Right-click **dc=domain, dc=com**.

    b. Choose **New→New Entry**.

    c. Select **Create entry from scratch**. Click **Next**.

    d. Select and add the **organizationalUnit** object class. Click **Next**.

    e. Select RDN **ou** and enter **staff**. Click **Next** and then click **Finish**.

2. Configure value editors.

    a. In Apache Directory Studio, choose **Window→Preferences**.

    Apache DS displays the **Preferences** dialog window.

    b. Expand **Apache Directory Studio** and then expand **LDAP Browser**.

    Alternatively, you can type **Value Editors** in the **type filter text** box, and then select **Value Editors**.

    c. In the **Value Editors by Attribute Types** section on the right, click **Add...**.

    d. In the **Attribute Type of OID** box, type **pwdAttribute**.

    e. From the **Value Editor** dropdown list, select **In-Place Text Editor**.

    f. Click **OK**, and then click **Apply and Close**.

3. Add a user in the organizational unit (**cn=user1** in **ou=staff**).

    a. Right-click **ou=staff**.

    b. Choose **New→New Entry**.

    c. Select **Create entry from scratch**. Click **Next**.

    d. Select and add **pwdPolicy**.

e. Select and add **inetOrgPerson**. Click **Next**.

f. Add the **cn=user1** attribute and click **Next**.

g. Add the following attributes and click **Finish**:

- **sn=user1**
- **uid=user1**
- **pwdAttribute=userPassword**

h. Set the password.

A. Open **cn=user1**.

B. Right -click and choose a select new attribute.

C. Select the **userPassword** attribute and click **Finish**.

D. Enter the password:

- Password: **user1**
- Hash: **SHA-256**

Create a remote LDAP reference

An LDAP reference allows one LDAP server to forward a lookup request to another remote LDAP server following a similar search hierarchy.

1. Create a partition that matches the domain of the remote LDAP server that you are referencing.
2. Right-click **dc=domain, dc=com**.
3. Choose **New→New Entry**.
4. Select **Create entry from scratch** and click **Next**.
5. Select the **extensibleObject** object class and click **Add**.
6. Select the **referral** object class and click **Add**. Click **Next**
7. Select RDN **ou** and enter the organizational unit of the remote LDAP domain and click **Next**.

8. When prompted for additional **ref** attributes, enter the URL and DN of the remote LDAP server, for example:

```
ldap://my.otherdomain.com:389/ou=People,dc=otherdoman,dc=com
```

9. Click **Finish**.

Password policy

ApacheDS password policy

For information about ApacheDS password policy entries, go to the LDAP browser tab at the top left-hand corner. Click the following node:

```
ads-pwdId=default,ou=passwordPolicies,ads-
interceptorId=authenticationInterceptor,
     ou=interceptors,ads-directoryServiceId=default,ou=config
```

All available password policy entries appear on the right. For a description of all password policy entries, see the Apache documentation:

http://directory.apache.org

In contrast, **ads-pwdnustchange**, with a value of **TRUE**, indicates that any newly created Teamcenter user, or a Teamcenter user whose password has been modified by an LDAP administrator, is automatically tagged to have the LDAP force its password change upon the next successful logon.

Add a Teamcenter password policy entry

> Note:
>
> Teamcenter extended the default embedded LDAP with additional password policy features, which are configured in a separate interceptor, **authenticationInterceptorTc**.

1. Navigate to the LDAP browser tab in the top left-hand corner, and then click the following node to display all available password policy entries:

```
ads-pwdId=default,ou=passwordPolicies,
     ads-interceptorId=authenticationInterceptorTc,ou=interceptors,
     ads-directoryServiceId=default,ou=config
```

   Following are Teamcenter password policy entries:

   - **tc-pwdnotifyemailserver** (required)

Specifies the name of an email server that is used when sending various password policy-related email notifications.

- **tc-pwdBlacklist** (optional)

  Specifies a list of all blacklisted passwords.

- **tc-pwdRegEx** (optional)

  Specifies a regular expression identifying a required password pattern.

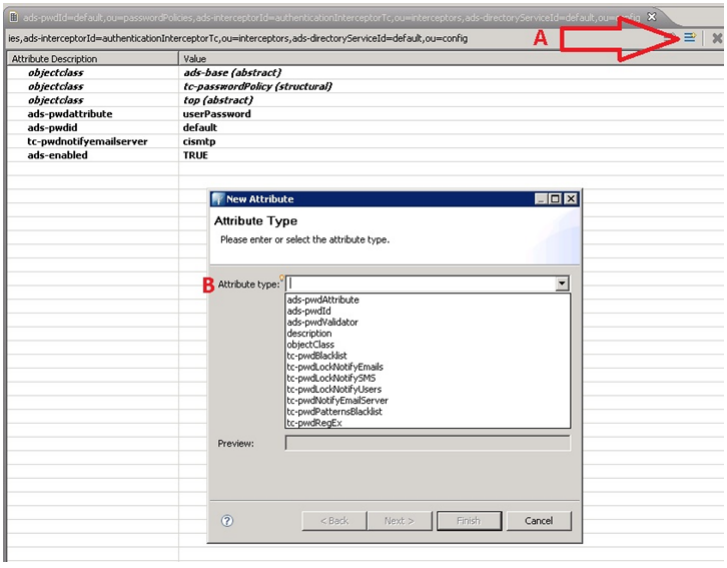
- **tc-pwdBlacklistPatterns** (optional)

  Specifies a list of entries that are not allowed to be part of a password.

- **tc-pwdLockNotifyEmails** (optional)

  Specifies a list of email addresses to send notifications to regarding Teamcenter users being locked out of their accounts.

2. Add a Teamcenter password policy entry:



a. Click the **New Attribute** icon.

b. Select Teamcenter password policy entry from the **Attribute type** list.

c. Click **Finish**.

d. Add a value for the new entry.

Sample password pattern

The optional attribute type, **tc-pwdRegEx**, specifies a regular expression identifying a required password pattern.

This field can be set with a regular expression pattern that implements custom password requirements for your site. For example, a sample regular expression pattern is:

```
^(?=.*\d{1,})(?=.*[$#!]{1,})(?=.*\w).{8,}$
```

where:

- **(?=.*\d{1,})**

  Specifies at least one digit.

- **(?=.*[$#!]{1,})**

  Specifies at least one special character of **$**, **#**, or **!**.

- **(?=.*\w)**

  Specifies letters.

- **.{8,}**

  Specifies at least 8 in length.

B. Add localization to Security Services

Security Services supports *localization* of text on user interface elements (such as buttons, tabs, and boxes) into the language of a particular locale. Security Services is installed in English by default, but provides localizations for the following locales:[1]

Chinese (Simplified): **zh_cn**	German: **de_de**	Polish: **pl_pl**
Chinese (Traditional): **zh_tw**	Italian: **it_it**	Portuguese (Brazilian): **pt_br**
Czech: **cs_cz**	Japanese: **jp_jp**	Russian: **ru_ru**
French: **fr_fr**	Korean: **ko_ko**	Spanish: **es_es**

To an existing Security Services installation, you can add a supported localization by performing the following steps:

1. Security Services uses UTF-8 as the default character encoding scheme to support the localization of text. To use this feature, configure the `-Dfile.encoding=UTF-8` Java option before JVM startup. This Java option must be set for the JVM on which the application server runs as well as the JVM on which the session agent runs. This option can be set for the session agent in the *SessionAgentStart.bat* file as follows:

   ```
   "%JAVA_HOME%\bin\javaw" -Dfile.encoding=UTF-8 –cp ...
   ```

2. Expand the Security Services ICD files for the locale you want to add:

   Windows:

   a. Browse to the *TCSS_ROOT***TcSecurity***locale* directory.

   b. Double-click the **INSTALL_SSO_***locale***.EXE** program icon.

   7-Zip displays a self-extractor dialog box.

   c. In the **Extract to** box, type the path to *TCSS_ICD*, and then click **Extract**.

   After 7-Zip extracts the ICD files, close the self-extractor dialog box.

   Linux:

   a. Change to the *TCSS_ICD* directory.

   b. Type the following command to extract Security Services ICD files to your host:

1 Locale codes denote language and country in two-letter codes for each (*language_country*).

```
cat TCSS_ROOT/locale/INSTALL_SSO_locale.TZ | uncompress -c | tar
xvf -
```

> Note:
>
> On Red Hat Linux systems, use the **gzip** command instead of **uncompress** to extract **INSTALL_SSO.TZ** file:
>
> ```
> cat TCSS_ROOT/locale/INSTALL_SSO_locale.TZ | gzip -d | tar
> xvf -
> ```

3. Launch the Web Application Manager.

   Windows: Browse to the *WEB_ROOT* directory and double-click the **insweb.bat** program icon.

   Linux: Change to the *WEB_ROOT* directory and type the **insweb** command.

4. Load Security Services ICD files for the localized solutions you want to install.

   a. Click **Copy ICDs**.

   b. In the **Copy ICD Files** dialog box, click **Browse**.

   c. Browse to the *TCSS_ICD* directory, select the **icd** directory, and then click **Open**.

   d. In the **Copy ICD Files** dialog box, click **OK** to load ICD files.

5. In the **Web Applications** list, select the application to which you want to add a locale, and then click **Modify**.

   The Web Application Manager displays the **Modify Web Application** dialog box.

6. Update software locations:

   a. Click **Modify Disk Locations**.

   b. Click **Add**.

   c. In the **Add Disk Location** dialog box, browse to the path to the locale you want to add, and then click **OK**:

      Windows: *TCSS_ROOT***TcSecurity***locale*

      Linux: *TCSS_ROOT*/**TcSecurity**/*locale*

7. Click **Add Solutions**.

8. In the **Add Solutions** dialog box, select the localized solution you want to add, and then click **OK**.

   For example, if you are updating the Login Service web application, select **Teamcenter Security Services Login Service Web Application Localization** *language***)**. If you are updating the Identity Service web application, select **Teamcenter Security Services Identity Service Web Application Localization** *language***)**.

9. Click **Generate Deployable File** to rebuild the WAR file for the web application.

10. Close the Web Application Manager.

11. Locate the updated WAR file in the staging location for the web application you updated.

12. Deploy the updated web application on a supported application server. Deployment procedures for Teamcenter web applications on supported application servers are described in *Web Application Deployment*.

C. Configuring Teamcenter products

Using Security Services with Teamcenter products

Security Services supports connections with the following Teamcenter products:

- Teamcenter
- Teamcenter Enterprise
- Engineering Process Management
- Portfolio, Program and Project Management
- Systems Engineering and Requirements Management
- Community Collaboration
- Lifecycle Visualization

Connecting these products to Security Services requires the following general configuration settings. For full configuration details to support Security Services, see the documentation for these products available on Support Center.

https://support.sw.siemens.com/

The following examples assume Security Services web applications deployed on web server named **tcsshost** using port **8080**, with WAR files named **TcLoginService.war** and **TcIdentityService.war**. The URLs for these applications are:

- Login Service URL:

  http://tcsshost:8080/TcLoginService

- Identity Service URL:

  http://tcsshost:8080/TcIdentityService

Teamcenter applications require some or all of the configuration variables in the following table to be set to enable Security Services.

Field	Description
Application ID	Specifies the ID associated with the Teamcenter application installed on the system. This value must correlate with the IDs entered in the Teamcenter Security Services component's application registries, for example, **TCEnterprise**.
SSO Login Service URL	Specifies the URL for the Login Service. For example:

Field	Description
	http://tcsshost:8080/TcLoginService
SSO Service URL	Specifies the URL for the Identity Service. For example: **http://tcsshost:8080/TcIdentityService**
SSO enabled	Provides a flag that indicates whether Security Services is enabled.

Note:

Each product has naming conventions and schemes for setting these variables. See the product documentation for full details about configuring Security Servicesfor each product.

Teamcenter configuration

Security Services is shipped in the Teamcenter software kit.

If you use Security Services with Teamcenter in *single sign-on mode*, set all the Teamcenter environment variables described in the following table. However, if you use Security Services in *authentication-only mode*, do not set the variables pertaining to the Login Service. Set only the **TC_SSO_SERVICE** and **TC_SSO_APP_ID** variables.

Variable	Description
TC_SSO_SERVICE	Specifies the URL of the Security Services Identity Service, for example: `TC_SSO_SERVICE=http://tcsshost:8080/TcIdentityService` Set this variable in the **tc_profilevars** file.
TC_SSO_LOGIN_PAGE	Determines the URL to which the client redirects the logon by setting the complete URL for the Security Services logon window. For example: **http://tcsshost:8080/TcLoginService/weblogin/login_redirect** Set this variable in the **tc_profilevars** file.

Variable	Description
	Note: • This URL must end in **/weblogin/login_redirect**. • This environment variable should only be set if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
TC_SSO_CHANGE_PASSWORD_PAGE	Determines the URL to which the web tier redirects the change password window by setting the complete URL for the local change password window provided using the identity provider used with Security Services. This environment variable is optional, and when available is set through the **tc_profilevars** file. Note: This environment variable should be set only if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
TC_SSO_LOGIN_URL	Directs the rich client to use the Security Services application client library to obtain an **appToken** from the Login Service. For example: **http://tcsshost:8080/TcLoginService** Set this environment variable in the rich client's properties file. For the two-tier rich client, the file is the **client_specific.properties** file. For the four-tier rich client, the file is the **site_specific.properties** file. Note: Set this environment variable only if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
TC_SSO_APP_ID	Sets the application ID and is required if Security Services is enabled. For example:

Variable	Description
	TC2412 Set this variable in both the **tc_profilevars** file for the server and the client's **site_specific.properties** file. Note: If you have more than one web tier sharing an instance of Teamcenter server, you must include each or their application IDs and separate them with a comma or a space in the **tc_profilevars** file. Set this environment variable only if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
MOZ_PLUGIN_PATH	Sets the JRE 1.5 (or later) plug-in path on a Linux platform. For example: `/admin/java/Solaris_JRE_1.5.0_11/plugin/sparc/ns7` Note: You can use this variable for both Mozilla and Firefox browsers.

Note:

On Windows systems, the Dispatcher client cannot start a new session using Security Services when Security Services is enabled on the host. If your Teamcenter configuration includes the Dispatcher client, you must configure it to bypass Security Services.

See *Dispatcher — Deployment and Administration*.

Teamcenter Enterprise configuration

Overview of Teamcenter Enterprise configuration

If you use Security Services with Teamcenter Enterprise, you must do the following:

- Set context parameters in the Web Application Manager.
- Set configuration variables using the Configuration Editor.

- Make certain to put a password on the User objects of all users expected to use Security Services.

  When you enable Security Services, every user defined in the Teamcenter Enterprise database can log on through non-Security Services means unless you set a password on every **User** object. The user does not need to know the password.

Setting context parameters

You must set the following context parameters in the Web Application Manager:

- **mwau.login.sso.enable**

  Set this to **TRUE**.

- **mwau.login.sso.application.id**

  For example:

  ```
  TCEnterprise
  ```

- **mwau.login.sso.login.service.url**

  For example:

  ```
  http://cvgm074:8080/ssoLOGINSERVICE
  ```

- **mwau.login.sso.service.url**

  For example:

  ```
  http://cvgm074:8080/ssoSERVICE
  ```

For more information about these context parameters, see the appropriate Teamcenter Enterprise server installation manual (for Windows or Linux).

Setting configuration variables in Teamcenter Enterprise

If you use Security Services with Teamcenter Enterprise, you must set the Teamcenter Enterprise configuration variable described in the following table using the Configuration Editor. The procedure for adding configuration variables is documented in the *Network and Database Configuration Guide*.

Variable	Description
SSO_SERVICE_URL	Specifies the URL to the Security Services Identity Service web application. This value must match the value of

Variable	Description
	the **mwau.login.sso.service.url** context parameter you set when you install Teamcenter Enterprise. For example: **http://cvgm074:8080/ssoSERVICE**

Engineering Process Management configuration

If you use Security Services with Engineering Process Management, you must set the Engineering Process Management environment variables described in the following table.

Variable	Description
TC_SSO_SERVICE	Enables Security Services functionality by setting the complete URL the ITK server uses to communicate with Security Services. For example: **http://cvgm074:8080/ssoSERVICE** Set this variable in the **iman_profilevars** file.
TC_SSO_LOGIN_PAGE	Determines the URL to which the web tier redirects the logon by setting the complete URL for the Security Services logon window. For example: **http://cvgm074:8080/ssoLOGINSERVICE/weblogin/login_redirect** Set this variable in the **iman_profilevars** file. Note: • This URL must end in **/weblogin/login_redirect**. • This environment variable should only be set if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
TC_SSO_CHANGE_PASSWORD_PAGE	Determines the URL to which the web tier redirects the change password window by setting the complete URL for the local change password window provided using the identity provider used with Security Services. This

Variable	Description
	environment variable is optional, and when available is set through the **iman_profilevars** file. Note: This environment variable should be set only if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
TC_SSO_LOGIN_URL	Directs the rich client to use the Security Services application client library to obtain an **appToken**. For example: **http://cvgm074:8080/ssoLOGINSERVICE** Set this in the **site_specific.properties** file. Note: This environment variable should be set only if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.
TC_SSO_APP_ID	Sets the application ID and is required if Security Services is enabled. Use this environment variable when multiple Engineering Process Management sites are served by a single identity provider or when Security Services is configured to use an ID other than Engineering Process Management. For example: `TCEngineering` Set this variable in both the **iman_profilevars** file for the server and the client's **site_specific.properties** file. Note: This environment variable should be set only if Security Services is enabled using the **TC_SSO_SERVICE** environment variable.

For more information about configuring Security Services in your Engineering Process Management environment, see the chapter on configuring Security Services in the Engineering Process Management *Configuration Guide*. Also, refer to the Engineering Process Management *Release Bulletin* for release notes pertaining to Security Services.

Portfolio, Program and Project Management configuration

If you use Security Services with Teamcenter portfolio, program and project management 2005 SR1 MP2 and later (with JRE 1.5 and later), you must set up Security Services using the **Configure Single Sign On** dialog box in the JDOT Server Administration program. You must supply a value for the fields described in the following table.

Variable	Description
SSO Service URL	Enables Security Services functionality by setting the complete URL that the ITK server uses to communicate with Security Services, for example: **http://cvgm074:8080/ssoSERVICE**
SSO HTML Login URL	Specifies the logon URL of the Security Services Login Service, for example: **http://cvgm074:8080/ssoLOGINSERVICE /weblogin/login_redirect** Note: This URL must end in **/weblogin/login_redirect**.
Application ID	Specifies the ID associated with the Teamcenter application installed on the system. This value must correlate with the IDs entered in the Security Services component's application registries, for example, **TCProject**.

For more information about configuring Security Services in your Portfolio, Program and Project Management environment, see the *Installation and Configuration Manual*.

Systems Engineering and Requirements Management configuration

If you use Security Services with Systems Engineering and Requirements Management, you must modify the parameters shown in the following table using the Configuration web window.

Variable	Description
SSO.AppID	Specifies the ID that Security Services uses to identify Teamcenter systems engineering, for example, **TCRequirements**.
SSO.Enabled	Indicates whether Security Services is enabled. Set this to **true**.
SSO.LoginURL	Specifies the URL of the Security Services Login Service, for example: **http://cvgm074:8080/ssoLOGINSERVICE**
SSO.ServiceURL	Specifies the URL of the Security Services Identity Service, for example: **http://cvgm074:8080/ssoSERVICE**

Community Collaboration configuration

If you use Security Services with Community Collaboration, you must configure Community Collaboration using the Community Collaboration Central Administration pages.

For complete instructions on integrating Security Services in Community Collaboration, see the *Community Collaboration Administration Help* and the *Community Collaboration Installation and Upgrade Guide* for your appropriate SharePoint server installation.

Lifecycle Visualization configuration

If you use Security Services with Lifecycle Visualization, you must enable Security Services using the following steps:

1. Choose **File→Preferences→Security Services**.

2. In the **Security Services Preferences** dialog box, select the **Perform Single Sign On Authentication** check box.

3. In the **SSO Login URL** box, type the Teamcenter Security Services URL, for example:

   http://cvgm074:8080/ssoLOGINSERVICE

Enable Teamcenter utilities to run with Security Services

While Teamcenter is configured with Teamcenter Security Services for authentication and single sign-on (SSO), you can configure non-interactive Teamcenter utilities (for example, SOA-based utilities or ITK utilities) for authentication with Teamcenter.

These utilities use administrative accounts (for example, **TcAdmin** and **Dc_Proxy**) to execute background processes. Since Teamcenter administrative user accounts do not exist in a corporate LDAP server, you must configure a special lightweight LDAP server using the Identity Service, in addition to the corporate LDAP Server. You can configure the Identity Service with multiple LDAP servers. You can use the ApacheDS LDAP server (supplied with Teamcenter Security Services install kit) or any external V3-compliant LDAP server to store these administrative users.

There are two common deployment scenarios:

- **Non-gateway mode**, where Security Services is configured with two LDAP servers.
- **Gateway mode**, where Security Services is configured with a reverse proxy server for delegated authentication with commercial authentication providers, such as IBM WebSEAL, CA SiteMinder, Microsoft Azure Active Directory, and Microsoft Active Directory Federation Services (ADFS).

For example, **TcFTSIndexer** is a non-interactive SOA-based utility that requires user ID/password-based authentication. To run this with Security Services, you must configure the Security Services **Identity Service** with an LDAP server that stores the user ID and password for the account used by the FTS indexer.

Run Security Services in non-gateway mode

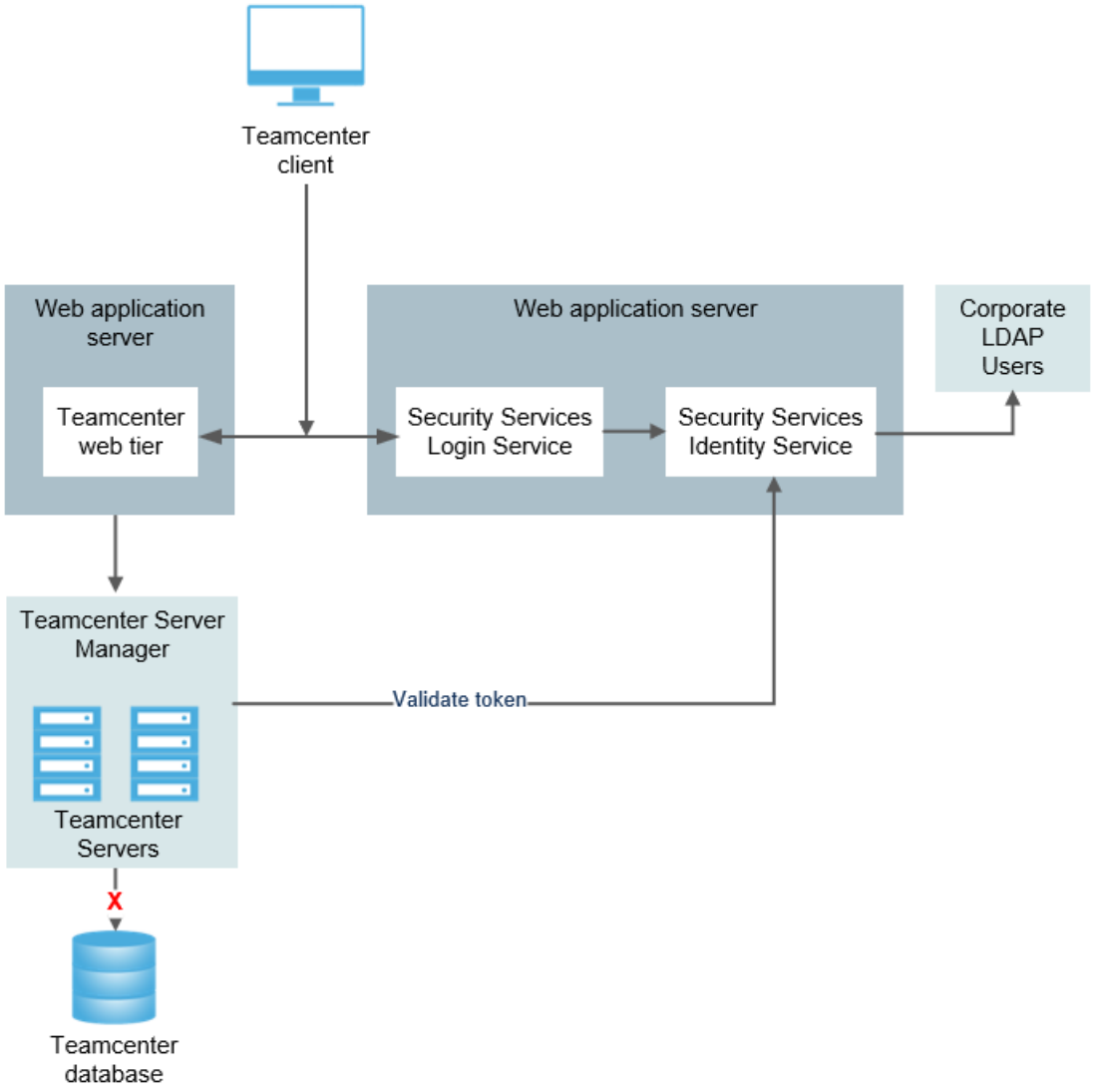
If you run Security Services in non-gateway mode, you must configure Security Services with two LDAP servers:

- A corporate LDAP server where regular users can be stored.
- A Security Services LDAP server where Teamcenter administrative users are stored.

Use Case 1 - Non-gateway authentication

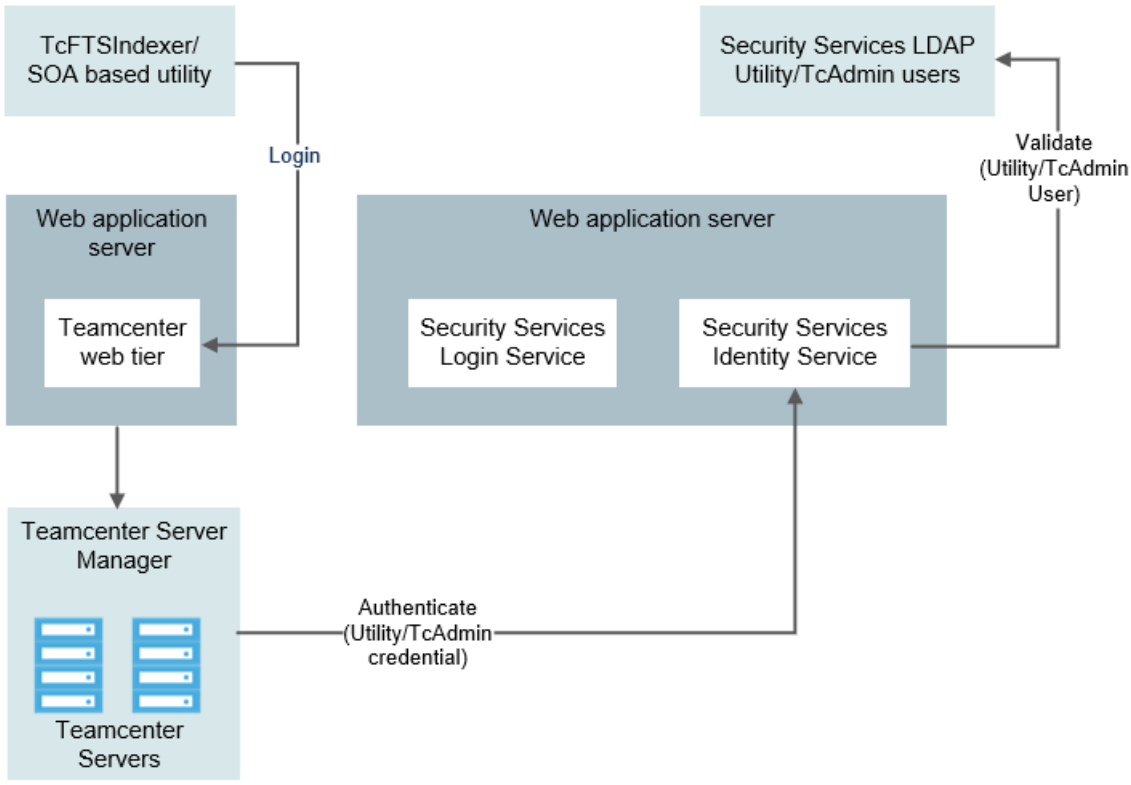
> Note:
>
> The following three scenarios (**A**, **B**, and **C**) can coexist in the same deployment.

- **Scenario A** represents authentication in Security Services done using the Login Service. Regular users are authenticated against the Corporate LDAP server.



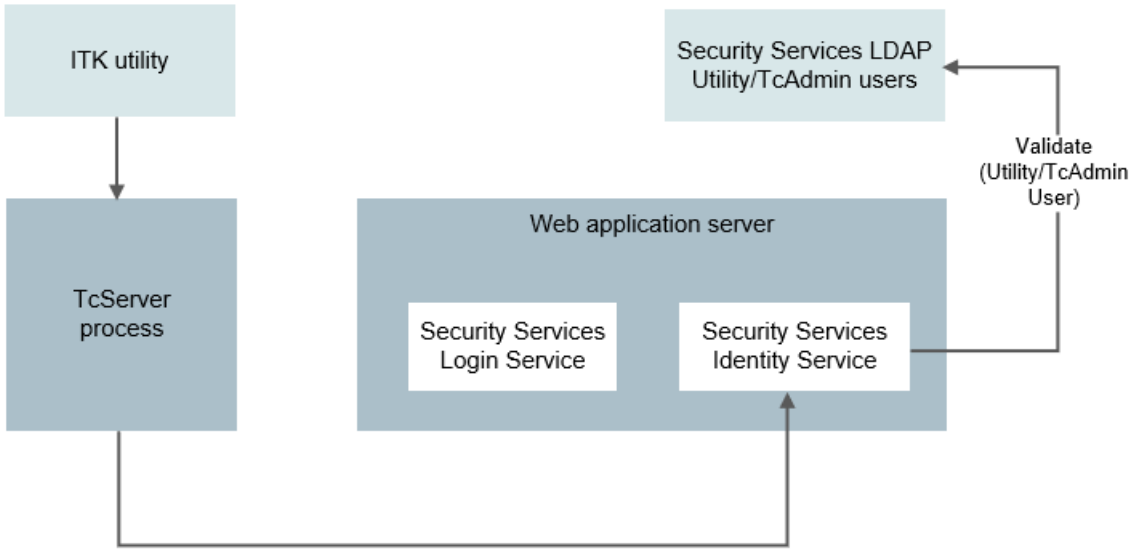
- **Scenario B** represents the path of authentication for SOA-based utility programs.

  In this mode, authentication is done using the user ID/password stored in Security Services LDAP. These are typically four-tier SOA-based utilities that use the SOA login API through which the user ID/password are supplied to the Teamcenter server. The TcServer validates this credential with the Identity Service, which validates the user with the configured LDAP server.



- **Scenario C** represents the authentication path for ITK utilities.

  In this mode, authentication is performed using the user ID/password stored in the Security Services LDAP server. This validation utility is typically a four-tier SOA-based utility, such as TcFTSIndexer, which uses the ITK logon API through which the user ID/password are supplied to the TcServer. The TcServer validates this credential with the Identity Service, which in turn validates the user using the configured LDAP server.



Run Security Services in gateway mode

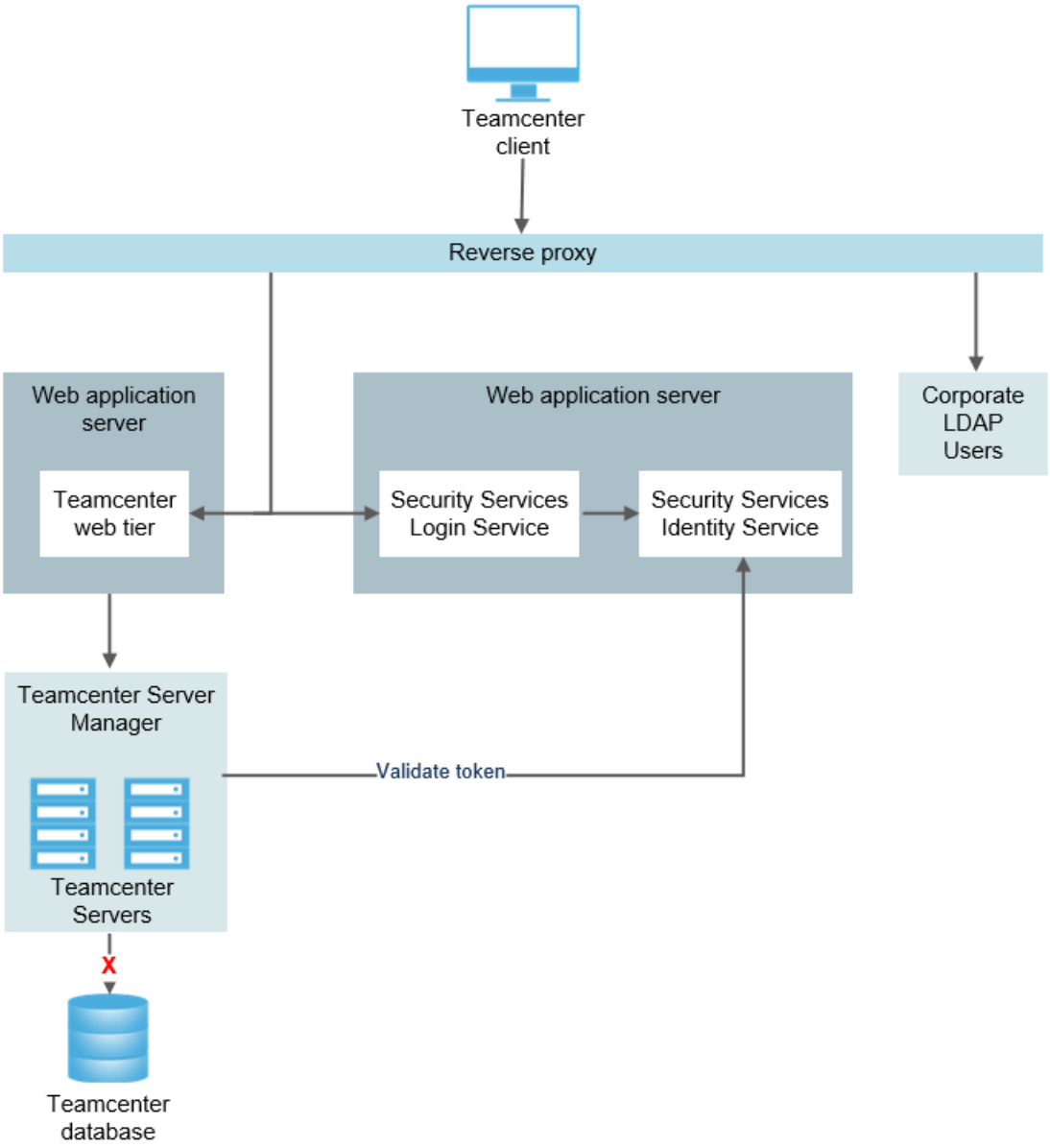
If you run Security Services in gateway mode, you must configure Security Services with the following servers:

- A reverse proxy server where regular users are authenticated.
- A Security Services LDAP server where Teamcenter administrative users are stored.

Use Case 2 - Non-corporate Teamcenter user authentication (Gateway Authentication)

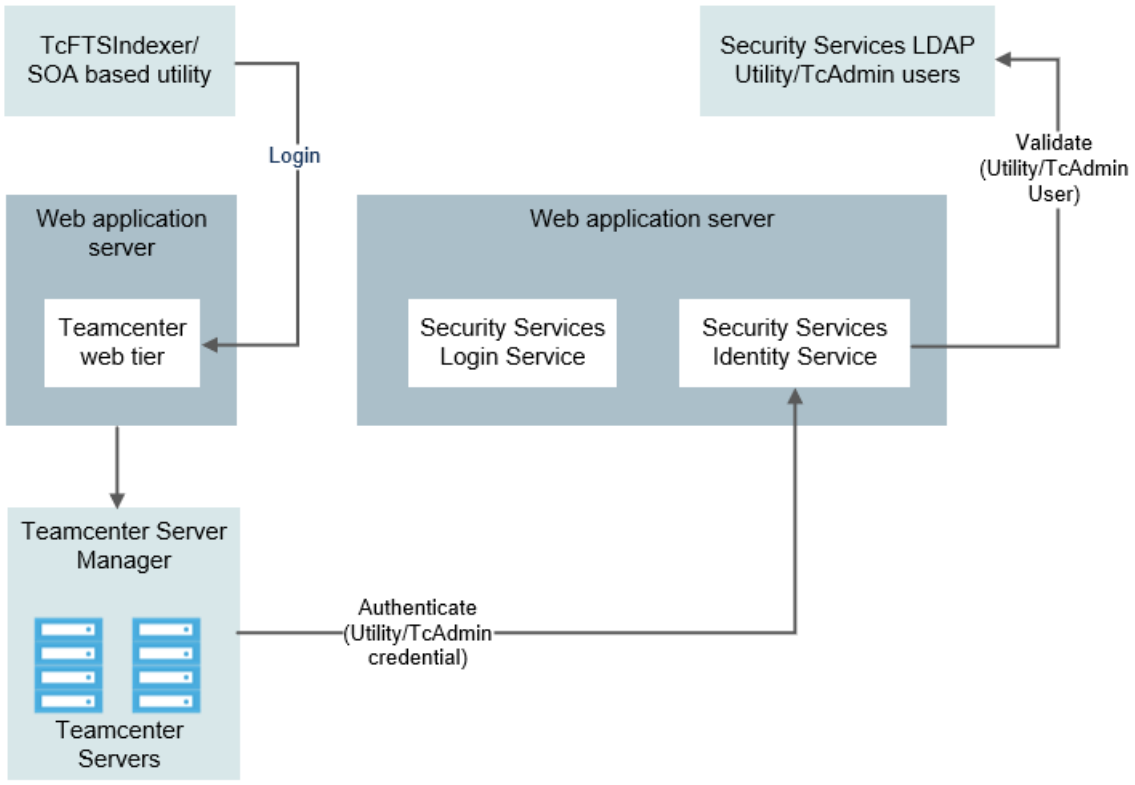
> Note:
>
> The following three scenarios (**A**, **B**, and **C**) can coexist together in the same deployment.

- **Scenario A** represents authentication in Security Services done using the Login Service. Regular users are authenticated using the Reverse Proxy (SiteMinder/WebSEAL/SAML/SmartCard).



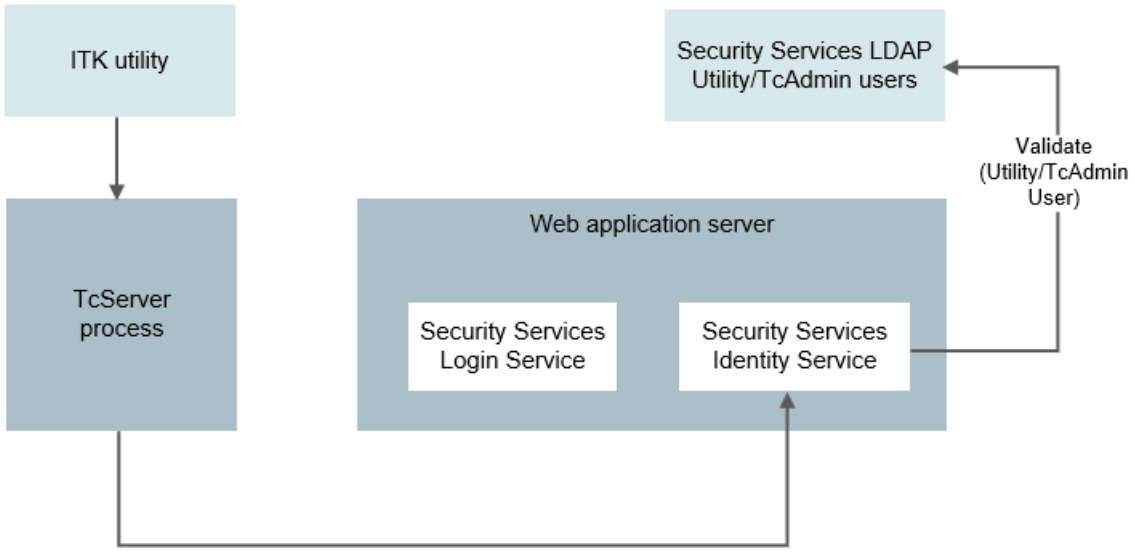
- **Scenario B** represents the path of authentication for SOA-based utility programs.

  In this mode, authentication is done using the user ID/password stored in Security Services LDAP. These are typically four-tier SOA-based utilities that use the SOA login API through which the user ID/password are supplied to the Teamcenter server. The TcServer validates this credential with the Identity Service, which validates the user with the configured LDAP server.



- **Scenario C** represents the authentication path for ITK utilities.

  In this mode, authentication is performed using the user ID/password stored in the Security Services LDAP server. This validation utility is typically a four-tier SOA-based utility, such as TcFTSIndexer, which uses the SOA logon API through which the user ID/password are supplied to the TcServer. The TcServer validates this credential with the Identity Service, which in turn validates the user using the configured LDAP server.



D. Reverse proxy support for external identity providers, WebSEAL, SiteMinder, File Management System (FMS), and two-way SSL

Overview of reverse proxy and external identity provider support

Teamcenter supports logon identities that originate from external identity provider products, such as SiteMinder, WebSeal, Microsoft Azure AD/ADFS, and Shibboleth. This is typically accomplished by configuring Teamcenter applications behind a reverse proxy or service provider. Reverse proxy, in turn, gets configured with one of these external identity providers for authentication.

Reverse proxy is typically a standard HTTP server product, such as Apache HTTP server, Microsoft IIS, or NGINX HTTP server, running in reverse proxy mode and enabled to do authentication using a web server plugin or module, which is typically provided by the external identity provider. The role of the reverse proxy is to intercept unauthenticated traffic intended for a protected target application and redirect the client to the external identity provider for authentication. The external identity provider can, in turn, choose any supported mode of authentication, such as a smart card, a secure ID, or a user Id/password. Once authentication is complete, the identity provider redirects the client back to the reverse proxy, which then permits and forwards the original request to the target application, in this case, the Security Services Login Service. Along with forwarding the original request, the reverse proxy passes the trusted user identity to the Security Services Login Service in one of the following forms: header, parameter, cookie, principal, or remoteuser. Based on the user identity in the request, Security Services can generate the necessary SSO tokens to authenticate the user with Teamcenter.

When Security Services is configured with authenticating reverse proxy, it can share authenticated SSO gateway session cookies with client-side Teamcenter applications to overcome the authentication challenge while configured via the reverse proxy. Providing the authenticated session cookie from the Security Services session allows the Teamcenter client application to share the session without a further authentication challenge.

Reverse proxy support is accomplished by securing the Security Services Login Service behind an authenticating reverse proxy configured to authenticate with one of these external identity providers.

Security Services can be configured to share authenticated SSO gateway session cookies with client-side Teamcenter applications that are facing form-based or **401** error-based authentication challenges from the SSO gateway. Providing the authenticated session cookie from the Security Services session allows the Teamcenter application to share the session without a further authentication challenge.

Security Services initially acquires a session cookie for each Security Services session when it is created. However, these session cookies have a finite lifetime. Therefore, an application can request a session cookie while Security Services does not presently hold an authenticated one. In this situation, Security Services launches a web browser, which initiates an authentication challenge from the SSO gateway.

The end user responds directly to this authentication challenge. After authentication, Security Services is reprovisioned with an authenticated session cookie and shares it with the requesting application.

Security Services provides this feature for IBM WebSEAL and CA SiteMinder SSO gateway products.

There are multiple ways to achieve cookie sharing. For example, WebSEAL requires you to create a junction cookie block in the page trailer, using either the WebSEAL console or the command line. If you use the console, select the **Insert Script as Trailer** option. If you use the command line, include the **-J** option, for example:

```
server task instance-webseald-host create ... -j -J trailer ...
```

See *Customizing Security Services* and this appendix for more information on reverse proxy configurations.

Session cookie sharing with FMS

File Management System (FMS) clients can automatically detect and respond to forms or **401** authentication challenges from the SSO gateway, acquire the session cookie from Security Services, and maintain the network connections to the FMS server cache (FSC) server. The FMS clients that support this feature are Java-based interactive clients, such as the Java FMS client cache (FCC) and the **UploadApplet** (web client). This feature only works with IBM WebSEAL and CA SiteMinder, for both forms-based and **401** authentication mechanisms.

In addition, FMS clients and server applications use path extensions on all URL addresses in HTTP messages. FMS provides path extensions to make it easier to configure reverse proxies to properly route FMS traffic to FSCs.

Two-way authentication is configured by setting up the HTTPS listener on a specified FSC to require two-way authentication. This ensures that only clients with certificates that can be authorized by the FSC servers can connect.

In a two-way SSL configuration, both the client and the server provide certificates to each other. This provides symmetric authentication between the FSCs. This ensures that only FSCs that are configured with appropriate certificates can communicate with each other. All other connection requests are denied.

Reverse proxy servers

About reverse proxy servers

Teamcenter client communication system (TCCS) supports form-based challenge from reverse proxy servers:

- IBM WebSEAL

- CA SiteMinder

Security Services supports cookie sharing in both of these reverse proxy servers, but you must enable the feature for the given server before you can use it.

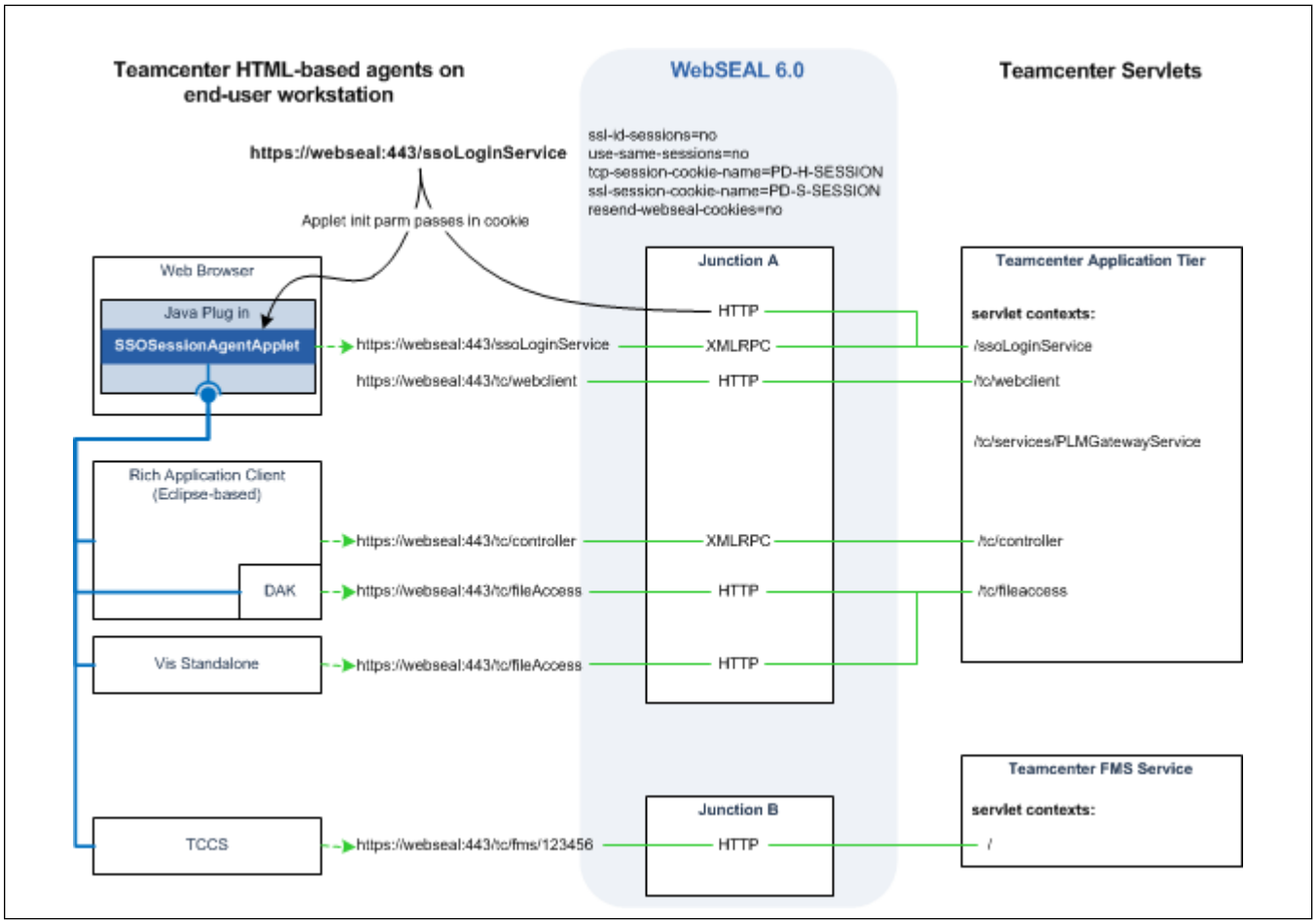
WebSEAL

Cookie sharing in WebSEAL

In the following figure, the black line indicates the HTTP/HTTPS session from which the junction cookie is shared. Client-side Teamcenter applications that encounter a form-based authorization prompt issue local XMLRPC calls to the Security Services Session Agent to obtain its shared WebSEAL cookie (blue lines). In turn, the shared cookie is inserted into an HTTPS request (dotted green lines) by individual Teamcenter agents making requests to the servlets.

> Note:
>
> The WebSEAL junctions uses junction mapping or transparent configurations. Otherwise, the junction name must be explicitly included.

> Note:
>
> Use only **$FMS_HOME**, not **$FSC_HOME**, in FMS configuration files. Always use Linux-style path separators (/).



WebSEAL session cookie sharing

Enabling WebSEAL proxy

To enable the WebSEAL feature within the clients:

- For TCCS or SOA communications, you must set the **TCSSO_LOGIN_SERVICE_URL** environment variable to the URL address of the Security Services Login Service as configured for the WebSEAL proxy server.

- You must create the **WEB_force_proxy_host** site preference and set its value to *WebSEAL-host*:*port-number*. For example, if WebSEAL is running on the **cologneibm2** host on port **9.1080**, set the value as follows:

  WEB_force_proxy_host=cologneibm2:9.1080

- The FSCs must be mapped from junction points on the WebSEAL proxy server.

- Junctions must be configured to include:

```
Client Identity Headers: User Name (Short)
Type: SSL or tcp
General Options: Insert WebSEAL Cookies
Scripting Support: Enabled
```

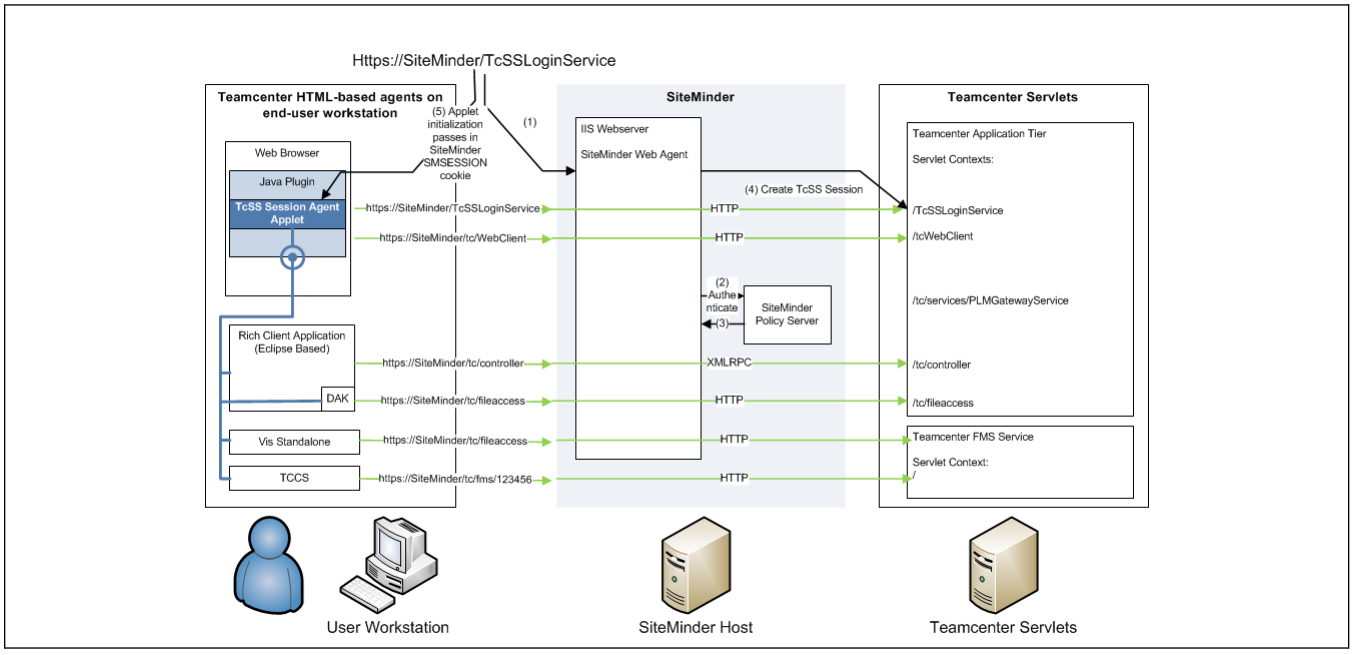
- In the WebSEAL configuration file, the **preserve-base-href** property value must be set to **yes**. This is the default value; ensure that it has not been changed.

SiteMinder

Cookie sharing in SiteMinder

In the following figure, the black line indicates the HTTP/HTTPS session from which the SiteMinder **SMSESSION** cookie is shared. Client-side Teamcenter applications that encounter an authorization prompt issue local XMLRPC calls to the Security Services Session Agent (**SSOSessionAgentApplet**) applet to obtain its session cookie (blue lines). In turn, the cookie is inserted into an HTTPS request (dotted green lines) by individual Teamcenter agents making requests to the servlets.

> Note:
>
> Use only **$FMS_HOME**, not **$FSC_HOME**, in FMS configuration files. Always use Linux-style path separators (/).



SiteMinder session cookie sharing

Enabling SiteMinder proxy

To enable the SiteMinder feature within the clients:

- For TCCS communications, you must set the **TCSSO_LOGIN_SERVICE_URL** environment variable to the URL address of the Security Services Login Service as configured for the SiteMinder Web Agent.
- For SOA client communications, you must set the **TC_SSO_LOGIN_URL** environment variable to the URL address of the Security Services Login Service as configured for the SiteMinder Web Agent in your TCCS environment.
- You must create the **WEB_force_proxy_host** site preference and set its value to *SiteMinder-host*:*port-number*. For example, if SiteMinder is running on the **svv6s072** host on port **80**, set the value as follows:

  WEB_force_proxy_host=svv6s072:80

- If you use SiteMinder for PKI authentication, perform the following steps to ensure SiteMinder uses the correct Security Services user ID.

  1. Disable the default headers at the agent level by setting **DisableUserNameVars** to **Yes**.
  2. Create a new rule, for example, **IdentityRule**, for the **OnAuthAccept** action.
  3. Add the new rule to the Security Services domain.
  4. Create a new response, **UserID**, with the variable name as **SM_USER**, the attribute name as **sAMAccountName**, and the attribute type as **User Attribute**.
  5. Add this response of the Security Services domain.

Troubleshooting reverse proxies

Several reverse proxy situations may result in problems. These problems and the likely solutions are described in the following sections.

Situation 1: The reverse proxy gives a "415 Unsupported media type" response, or other TCCS connection errors.

For example:

```
Connection failure: Attempted read from a closed stream
```

This failure can show up as a number of different errors and may not be limited to the ones listed here.

Affected versions, platforms and products

Teamcenter 12.2.0.2 and later.

Teamcenter web tier

Solution

To avoid this error, add the following line to the TCCS configuration file under the **ProgramData** folder if utilizing a shared TCCS configuration, or within the user's home folder if using a private TCCS configuration. For example, in a shared configuration:

1. Modify the **tccs.xml** file located here:

```
C:\ProgramData\Siemens\cfg\tccs\Teamcenter\tccs.xml
```

2. Add the following line within the **httpconfig** tag:

```
<usegetonredirectedpost overridable="true" value="true"/>
```

Cause

The web tier is deployed behind an authenticating reverse proxy.

When the Teamcenter web tier is deployed behind and protected by an authenticating reverse proxy server, use Teamcenter client communication system (TCCS) to handle the proxy authentication challenge.

As part of the authentication process, many Teamcenter rich clients send authentication information intended for Teamcenter Webtier in the form of an HTTP POST request. During authentication, this POST request is intercepted and considered invalid by many reverse proxies. This typically elicits an invalid HTTP response from the reverse proxy ending in communication failure. The result is that Teamcenter rich clients fail authentication even after a successful browser login with Security Services.

Situation 2: ERROR - 2020/10/07-10:04:59.206 UTC - [18] TcSSCommonHttpSessionHandler:getSession(-1): org.eclipse.jetty.http.BadMessageException: 500: Request header too large

This error occurs during logon by a rich client using the Security Services Session Agent. In the Session Agent log file, a similar error message is found.

Affected versions, platforms and products

Teamcenter 12.3 and later.

Security Services Session Agent and Teamcenter rich client

Solution

Set a larger buffer size using the **TCSSO_SA_REQ_BUFFER_SIZE** setting in the **SessionAgentStart.bat** or **SessionAgentStart.sh** file as shown below. When unset, the default value is **4096**; the standard maximum allowable value is **65536**.

```
@echo off
Setlocal
```

```
set TCSSO_SA_REQ_BUFFER_SIZE=65536
IF NOT DEFINED JAVA_HOME GOTO JAVA_HOME_NOT_DEFINED
```

Cause

Session Agent request header size

Security Services maintains session cookies within the Session Agent that are acquired from the reverse proxy. In some cases, proxy session cookie values may be large and exceed the default maximum size configured by the Session Agent.

Situation 3: Frequent and intermittent logon failure when using the Shibboleth Service Provider reverse proxy behind a load balancer.

Affected versions, platforms and products

All versions of Teamcenter and Shibboleth Service Provider versions 2 and 3.

Security Services Session Agent and Shibboleth Service Provider

Solution

Add and set the **consistentAddress** setting to **false** in the **shibboleth2.xml** configuration file inside the Sessions element.

```
<Sessions lifetime="28800" timeout="3600" relayState="ss:mem"
     checkAddress="false" handlerSSL="false" cookieProps="http"
     consistentAddress=”false”>
```

Cause

Shibboleth Service Provider behind a load balancer

The Shibboleth Service Provider maintains a session with client applications based on the original client IP address. When deployed behind a load balancer, the originating client IP address is masked and can alternate based on load balancer configuration. This violates the Shibboleth Service Provider's ability to confirm the source IP address, which results in an invalidation of the session.The reverse proxy gives a `415 Unsupported media type` response, or other TCCS connection errors.