TEAMCENTER

Security Administration

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

Getting started with security administration

Secure your Teamcenter implementation 1-1
Privileges you need to administer security tasks 1-2
Securing Teamcenter: Process overview 1-3
- When does security planning and implementation take place? 1-3
- Process flows for planning, implementing, and maintaining Teamcenter security 1-3
- Business process for authorization 1-4
- Business process for audit 1-5

Levels of Teamcenter security 1-5

Understanding Teamcenter security administration

What you need to know for Teamcenter security administration 2-1
Teamcenter object model hierarchy 2-1
What is authentication? 2-2
Understanding authorization 2-2
- What is authorization? 2-2
- Rules-based protection 2-2
- Object access control lists 2-3
- Access control lists 2-5
- Access Manager rule tree 2-6
- How rules work 2-7
- Best practices for rules 2-15
- Cautions for using rule trees 2-18

What is project-level security? 2-19
What is group-level security? 2-19
What is authorized data access? 2-19
About effectivity and access control 2-20
Multi-Site Collaboration considerations 2-20
- Overview of Multi-Site Collaboration considerations 2-20
- Remote checkout privilege access 2-21

Controlling access to working data

About controlling access to working data 3-1
About configuring access to working data 3-1
- Overview to configuring access to working data 3-1
- Access to effectivity 3-2

Guidelines for applying the delete and change privileges 3-4
Example of defining access controls on object class, type, and name 3-4
Control access to revision rules 3-8
Example of controlling access to effectivity 3-9
Using sponsored authentication 3-11
- Sponsored authentication overview 3-11

Example of sponsored authentication 3-12
Security logs track events 3-12

Controlling access to in-process data
About controlling access to in-process data 4-1
Workflow accessors and privileges 4-1
Workflow ACL example 4-3
Parallel task and parallel process ACL conflict resolution 4-3
Workflow access examples 4-3

Controlling access to scheduling data 5-1

Configuring external group security 6-1

Configuring security for remote export and remote checkout 7-1

Configuring group-level security
About configuring group-level security 8-1
Example of configuring security to prevent suppliers from viewing internal data 8-2
Example of configuring security for data owned by a supplier (external data) 8-3
Example of configuring supplier security using hierarchical groups 8-3
Example of configuring security for special project data using hierarchical groups (fully restrictive external group security) 8-5

Configuring security for project and program data
About configuring security for project and program data 9-1
What are projects and programs? 9-1
What are groups? 9-2
Assigning security classification on a workspace object 9-2
Applying project and program security (Access Manager) 9-3
Overview of applying project and program security (Access Manager) rules 9-3
Preferences related to project and program security 9-4
Default security rules for project and programs administration 9-4
Access rules for projects and programs 9-4
In Current Program rule 9-5
In Inactive Program rule 9-5
Is Program Member rule 9-6
In Invisible Program rule 9-6
In Project rule 9-7
Is Project Member rule 9-7
Is Owned by Program rule 9-8
Project-level security based on groups 9-8
Granting user-based access to project data 9-9

Introduction to granting user-based access to project data 9-9
Configuring user-based access to project data 9-10
Granting role-based access to projects 9-13
Overview to granting role-based access to projects 9-13
Configuring role-based access to project data 9-13
Configuring security when a user is a privileged member of multiple projects 9-16
Configuring security to protect competitive data when multiple suppliers are members of a common project 9-18
Implementation considerations for project-level security 9-19
Placement of rules in the Access Manager rule tree 9-19
Set security precedence 9-19
Program security examples 9-19
About the program-level security examples 9-19
Example 1 — Grant read access to only team members of object's assigned projects 9-20
Example 2 — Grant read access to all team members 9-21
Example 3 — Grant read access to item to all team members to all data in project 9-22
Example 4 — Grant access to single item 9-23
Example 5 – Deny read access 9-24
Implementation considerations for program-level security 9-25
Using Security Logs to track events 9-25

Configuring access based on geography

About geography access 10-1
Configure geography access 10-1
Configure a custom confidentiality agreement 10-3

Managing user consent

About GDPR 11-1
Steps to create and configure a user consent GDPR statement 11-1

Configuring ADA License

Getting started with ADA License 12-1
What is ADA License? 12-1
Before you begin 12-2
ADA License interface 12-3
Authorized data access licensing workflow 12-5
Types of Authorized Data Access (ADA) licenses 12-5
Basic tasks using ADA License 12-6
About configuring ADA License 12-7
Configuring ADA access 12-8
Overview of configuring ADA access 12-8
Controlling access to authorized data 12-9
Controlling access to the ADA License application 12-9
Recommended Access Manager rules 12-10

Controlling access to view and apply ADA licenses 12-11
Defining license categories 12-13
Setting user citizenship 12-20
Controlling license propagation 12-21
Configure audit log access 12-22
Good ADA license practices 12-23
Configuring ADA License for IP 12-23
About configuring ADA License for IP 12-23
Applying Access Manager concepts to IP classified data 12-23
Scenario – Implementing ADA for IP based on roles and projects 12-28
Scenario – Implementing ADA for IP based on groups 12-34
Scenario – Implementing ADA for IP at the dataset level 12-40
Implementation considerations 12-44
Basic tasks for configuring and administering ADA for IP data 12-47
Configuring ADA for ITAR support 12-55
About configuring ADA for ITAR support 12-55
Applying Access Manager concepts to technical data subject to ITAR 12-56
Scenario for implementing ADA for government classified data 12-68
ITAR implementation considerations 12-72
Basic tasks for configuring and administering authorized data access for ITAR restricted data 12-75
Managing ADA licenses 12-85
Overview of managing ADA licenses and required permissions 12-85
About setting a lock date 12-85
About setting an expiry date 12-86
About setting user or group access 12-87
Create licenses 12-88
Modify licenses 12-90
Delete licenses 12-91
Managing ADA licenses on workspace objects 12-91
Overview of managing ADA licenses on workspace objects and required permissions 12-91
Attach licenses to workspace objects 12-92
Detach licenses from workspace objects 12-93
View licenses attached to a workspace object 12-94
Managing ADA license audit logs 12-95
Audit log contents 12-95
View audit logs in the Summary view using the current Audit Manager 12-96
Exporting an audit log using the current Audit Manager 12-97

Customizing ADA security mapping

About customizing ADA security mapping 13-1
Creating a custom ADA security map 13-1
Step 1. Create a custom LOV for EAR levels 13-1
Step 2. Create a custom LOV for security levels 13-3
Step 3. Add a custom persistent property 13-3
Step 4. Add property constants 13-4
Step 5. Attach the security level LOV 13-4

Step 6. Add a custom clearance property for the user 13-5
Step 7. Create a custom license subtype 13-6
Step 8. Creating ADMIN and CLASSIFIER privileges 13-7
Step 9. Using the ADA_Custom_Security_Mapping preference 13-7
Step 10. Using Access Manager conditions with custom security groups 13-8
Step 11. Custom classification propagation 13-12

Configuring PKI digital signature

What is PKI digital signature? 14-1
Configuring your system for PKI digital signature 14-2
Install Teamcenter 14-2
Step 1: Enter digital signature certificate information 14-3
Step 2: Apply business constants 14-5
Step 3: Edit XRT (style sheet) 14-6
Step 4: Create the rich client plug-in 14-7
Step 5: Set the preference to display digital signature icons 14-15
Step 6: Set PKI digital signature environment variables 14-16
Step 7: Generate Access Manager rules 14-17
Step 8: Apply PKI digital signature to objects in workflow 14-21
Step 9: Add property to property policy file 14-21
Using PKI digital signature 14-22
Using your smart card 14-22
Applying a PKI digital signature 14-23
Apply a valid PKI digital signature 14-23
Void a PKI digital signature 14-24
Delete a PKI digital signature 14-25
View digital signature audit logs 14-26
Suppress digital signature commands 14-26
Troubleshooting digital signature 14-27
What do I do if my certificate expires or I have a new certificate? 14-27
How do I customize a hash algorithm for digital signature? 14-28

Controlling access to classification objects

Classification access control overview 15-1
Component display suppression 15-3
Hierarchy component protection 15-3
ICO protection 15-3
Restrictions 15-5
Classification access privileges 15-6
Applying access controls examples 15-7
Example: controlling the display of the hierarchy tree for Classification users 15-7
Example: controlling access to hierarchy definitions 15-8
Example: controlling access to ICOs 15-10
Creating access rules 15-11
Create a named access control list (ACL) 15-12
Modify access control list entries 15-13
Delete access rules 15-14

Controlling access based on compound property values

About controlling access based on compound property values 16-1
Has Property condition example 16-1

Configuring Oracle PKI

Configure Oracle Wallet 17-1
Create the root wallet and trusted certificate 17-2
Create the database server wallet and its certification 17-3
Create the user wallet and its certificates 17-4

Rule conditions, accessor types, and privileges

What are access rules composed of? A-1
Access conditions by group A-1
Accessor types by category A-6
Accessor precedence A-15
Access privileges A-16

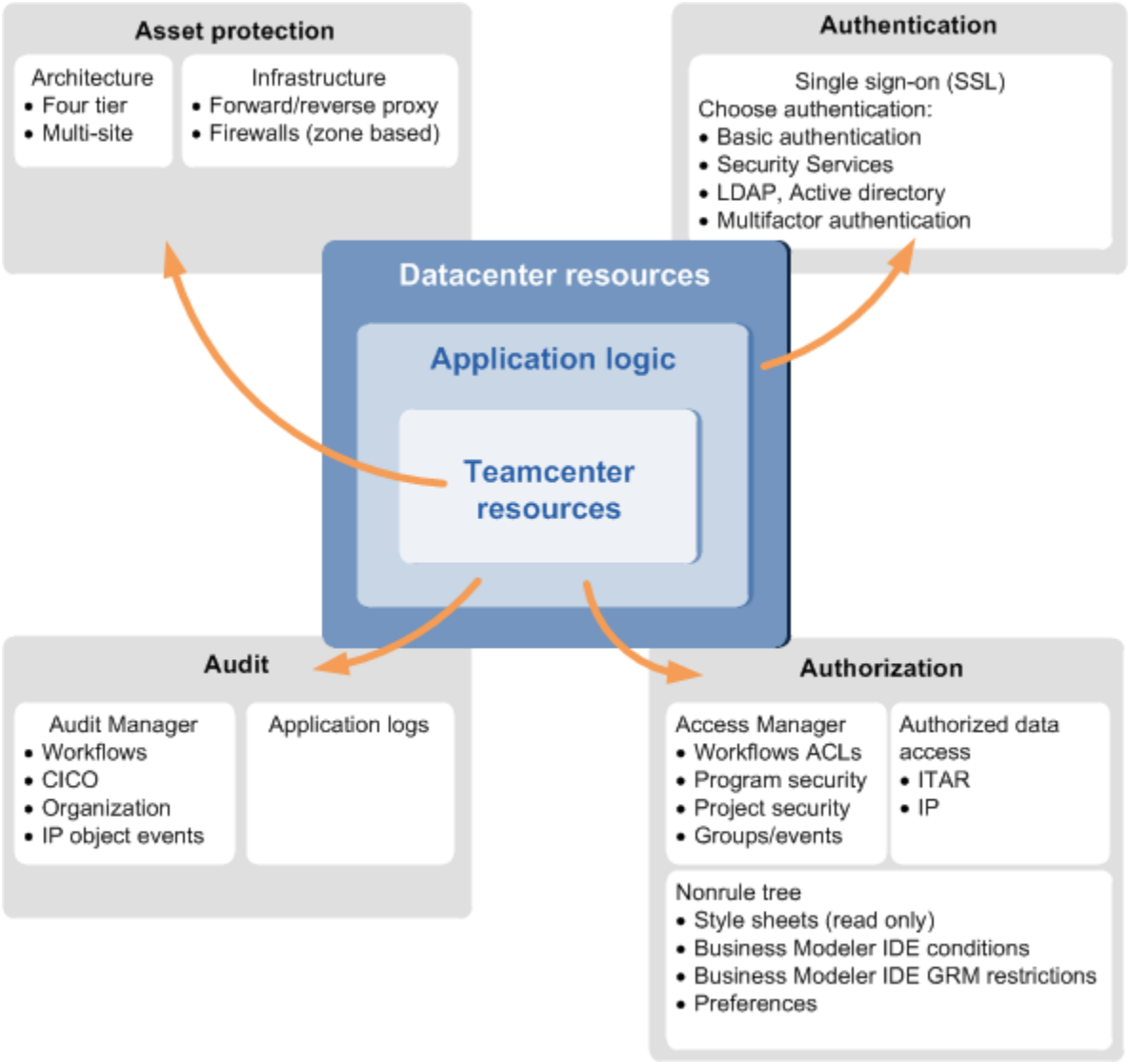
1. Getting started with security administration

Secure your Teamcenter implementation

Managing how users access your company data is an important factor in information security. Users may be employees within your company, or they may be external users such as suppliers and contractors.

To secure your Teamcenter implementation, consider these areas of security:

- **Authentication** determines how a user's identity is verified at logon. This includes Security Services.
- **Authorization** specifies the functionality available to a user.
- **Audit** generates checks and reports to help maintain security settings (Audit Manager, Organization, and Workflow Designer applications).
- **Asset protection** indicates the protections that are in place to protect data assets (Access Manager and ADA License applications).



As a Teamcenter administrator, you define and implement security controls for each of these areas using third-party tools or Teamcenter applications. You should be familiar with the Teamcenter data model, the organizational structure of the enterprise, and the business rules that are used to govern the enterprise's processes.

Privileges you need to administer security tasks

You must have administrative privileges to perform most security-related tasks. In addition, some tasks require you to have administrative privileges related to the type of security being implemented. For example, you must have project administrator or project team administrator privileges to perform project-related security tasks.

Securing Teamcenter: Process overview

When does security planning and implementation take place?

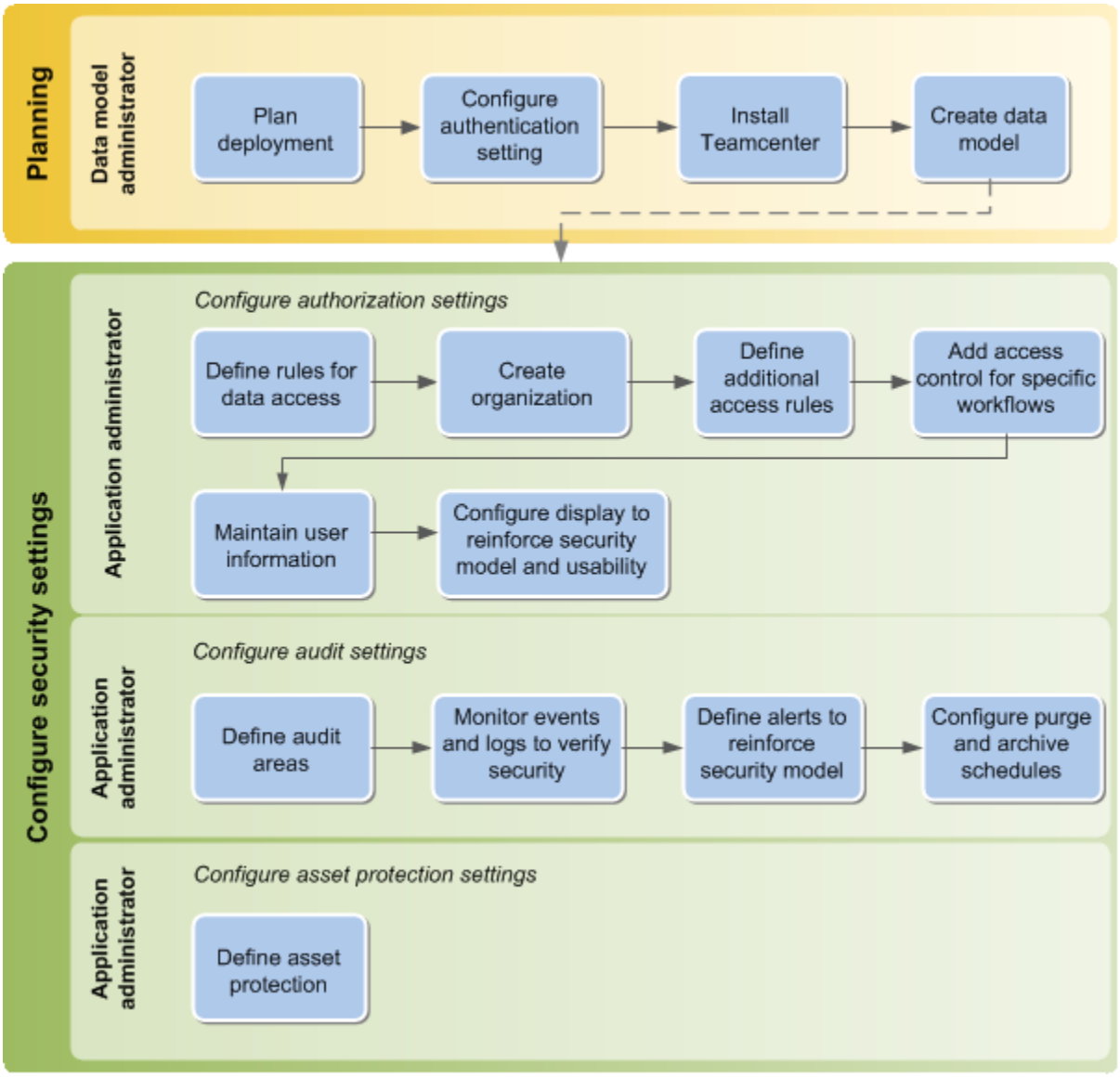
You should include security in your overall Teamcenter deployment planning activities before you install Teamcenter. The decisions you make as part of this planning should take your security needs into account. In terms of your business processes, the breakdown of particular roles may align to the following process.

Business process	Description	Role
Planning	The *planning* business process provides requires work done prior to installation to plan the deployment and determine how the authentication setting is configured for the Teamcenter installation.	Data model administrator
Configure security settings	The *configure security settings* business process includes the following: • Configure authorization settings using Access Manager, creating the organization with the Organization application, and then continuing by adding access rules and specific workflows. • Configure audit settings using Audit Manager and monitor events and logs to verify security. • Configure asset protection settings.	Application administrator

Process flows for planning, implementing, and maintaining Teamcenter security

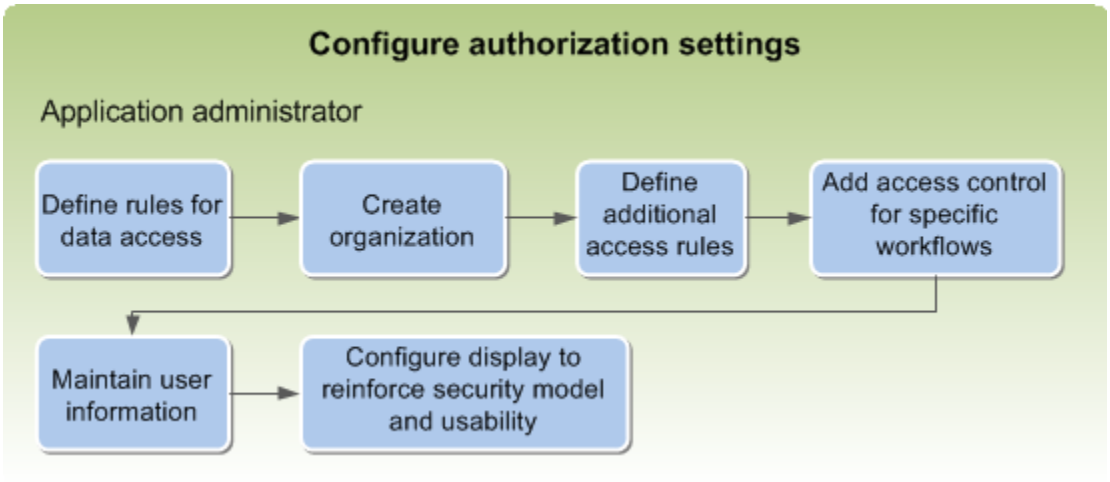
As part of Teamcenter deployment, your data model administrator addresses and configures authentication settings. Typically, your application administrator:

- Sets up the rules, organizational structure, and access controls that configure authorization settings.
- Configures settings to put audit processes in place to help maintain and monitor security controls.
- Configures settings to protect data assets.



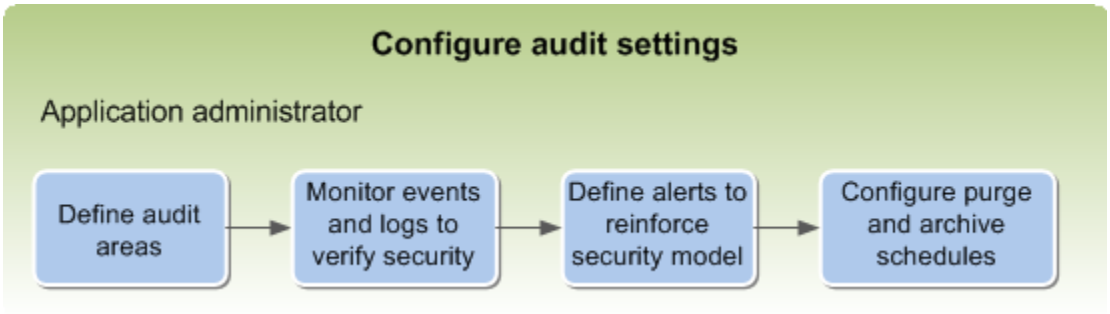
Business process for authorization

The typical process for **configuring authorization** settings begins with defining access rules. You can base your rules on a global or object-by-object approach.



Business process for audit

The typical process to follow when configuring audit settings begins with identifying what events you want to monitor and the logs you want to generate. You can view audit logs using Teamcenter applications, such as My Teamcenter, ADA License, Structure Manager, Multi-Structure Manager, Manufacturing Process Planner, Schedule Manager, Workflow Viewer, and Organization.



Levels of Teamcenter security

Consider three levels of Teamcenter security. Each security level has advantages and disadvantages, as shown in the following table.

Security level	Description	Advantage	Disadvantage	Example
Low	The out-of-the-box level of security in Teamcenter. This security level is best suited for a development and test environment with	Simple to design and implement.	Limited security of user identity and data.	An environment used for development and testing with a small level of intranet usage. It

Security level	Description	Advantage	Disadvantage	Example
	a small level of intranet usage. **Relevant documentation:** *Organization Management Using Groups, Roles, and Users*.			involves internal users only. There are no external suppliers or partners.
Medium	This level of security leverages special security features and/or modules along with industry standard secure protocols. In this scenario, Teamcenter is deployed with a multi-tiered infrastructure. **Relevant documentation:** *Security Services Configuration*, third-party directory service documentation, and web application server documentation.	Provides a reasonable level of protection for identity and data and is suited for global usage.	Requires security expertise across all areas of deployment. Therefore, Teamcenter requires higher cost for design, deployment, and administration.	A business confined to a single country or a few countries. This requires groups and projects, as well as a firewall, a proxy server, and involves multi-site implementation for external access.
High	This level of security uses Teamcenter's complex features. For example, at this level, you implement authorized data access features of ITAR and intellectual property because Teamcenter users may access international data that may be highly sensitive or proprietary. **Relevant documentation:** *Security Services Configuration*, *Audit Logs Management*, and *Security Administration*.	Supports corporate and regulatory compliance requirements for the protection of identity and data.	Requires security expertise across all deployment aspects. Design, deployment, and administration at this level is more expensive.	An international company that accesses data stored worldwide. This requires authorized data access (ITAR and IP).

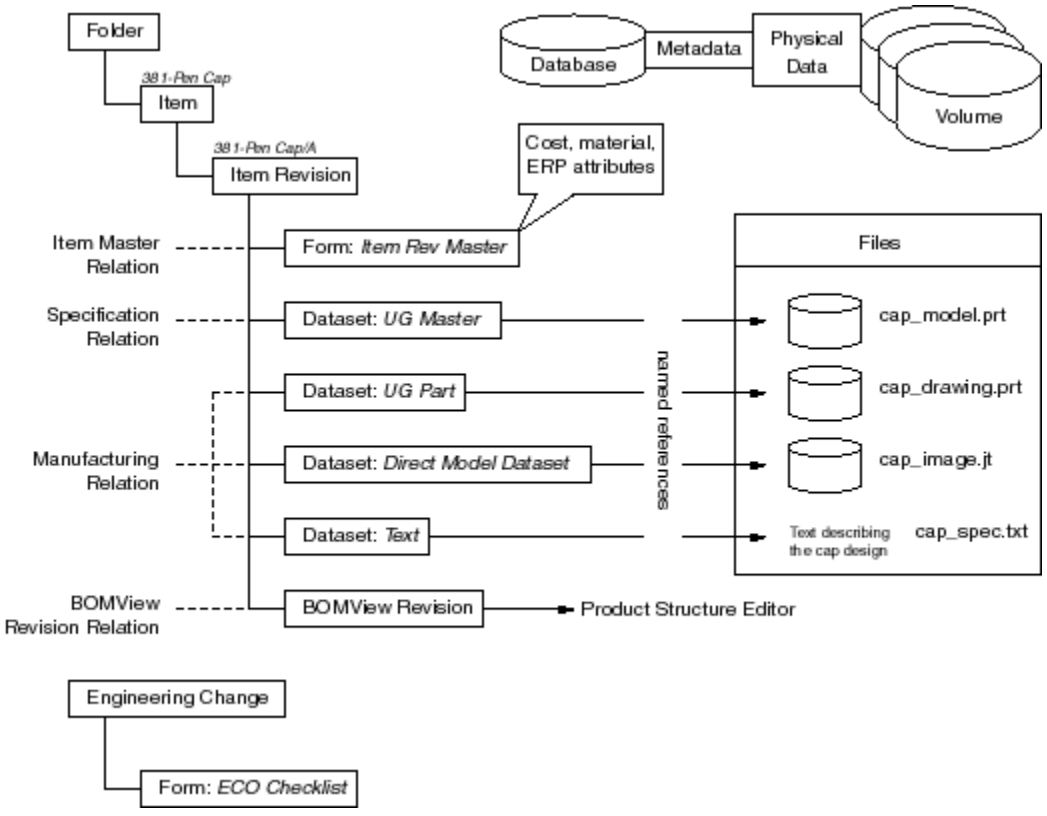
2. Understanding Teamcenter security administration

What you need to know for Teamcenter security administration

Teamcenter security administration requires an understanding of the object model hierarchy and the concepts of authentication and authorization. Authorization is based on rules arranged hierarchically in the Access Manager rule tree and on object-based protection that grants exceptions to the rules.

Teamcenter object model hierarchy

The following is a simplified illustration of the major concepts of the Teamcenter object model hierarchy. It is important to understand the object model, as some of the security implementations described in this guide propagate information down the hierarchy structure.



Note:

Master forms inherit their permissions from their parent item or item revision. You can use the **TC_MASTERFORM_DELEGATE** environment variable to change the default behavior.

What is authentication?

Authentication refers to gaining access to a Teamcenter application or product solution. Authentication to load applications, such as Structure Manager, in your Teamcenter session is provided by the Siemens Digital Industries Software Common Licensing Server daemon.

In addition to application authentication within Teamcenter, Security Services allows users to move from one Teamcenter product solution, such as Teamcenter Enterprise, to another solution, such as Teamcenter lifecycle visualization, without encountering multiple authentication challenges. Security Services includes the following features:

- Single sign-on to Teamcenter products for the rich client.
- Common authentication through LDAP v3-compliant directory servers, such as Microsoft Active Directory and the Sun iPlanet Directory Server, which can be customized to work with other authentication services.
- Interoperation with commercial single sign-on products.
- Lightweight directory access protocol (LDAP) referrals, which allow users to be distributed across multiple LDAP servers.

Security Services is installed and configured separately from Teamcenter 2412.

Authentication can be implemented for performing Do tasks, Perform signoff tasks, Condition tasks, and Route tasks using the **require-authentication** handler.

Understanding authorization

What is authorization?

Authorization refers to the implementation of rules that control access to specific data stored in the Teamcenter database. Authorization to interact with data is controlled by a combination of global rules (*rules-based protection*) and object access control lists (ACLs) applied to specific objects that allow for exceptions to the global rules (*object-based protection*).

Using rules and ACLs in combination with information about the user, such as their group and project membership, nationality, and clearance level, enables you to design and implement sophisticated security models to protect your data.

Rules-based protection

Rules provide security for your Teamcenter data by:

- Controlling access to data on a global basis.

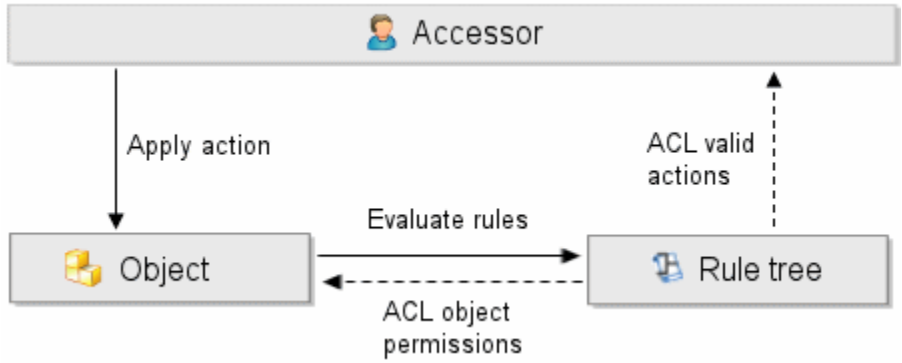
- Determining whether a user has permission to view or perform an action on an object.
- Filtering data according to the attributes of the data.
- Granting privileges to the data according to the users' IDs and their session context (the group and role they used to log on).

> Note:
>
> Rules do not control the creation of objects. They only determine what operations can be performed on existing objects.

Rules are defined by a combination of:

- A condition.
- A value for the condition.
- An access control list (ACL) that grants privileges to accessors.

The condition and value identify the set of objects to which the rule applies; the ACL defines the privileges granted to users (accessors).



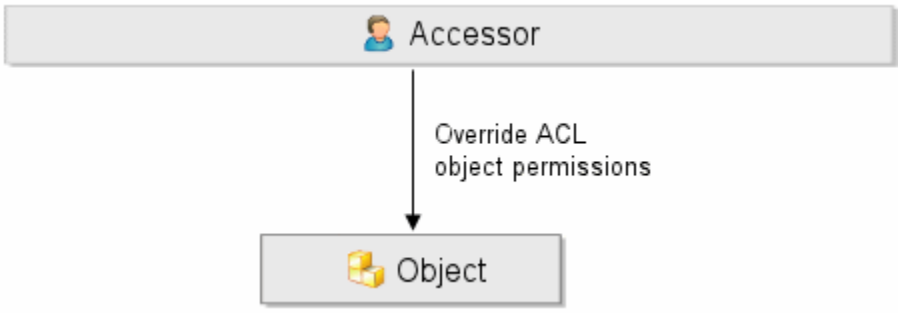
User actions against objects cause the rule tree to be evaluated to dynamically build an access control list for the object. The ACL controls permissions for the object and determines who (accessors) can do what (actions) to the object.

Object access control lists

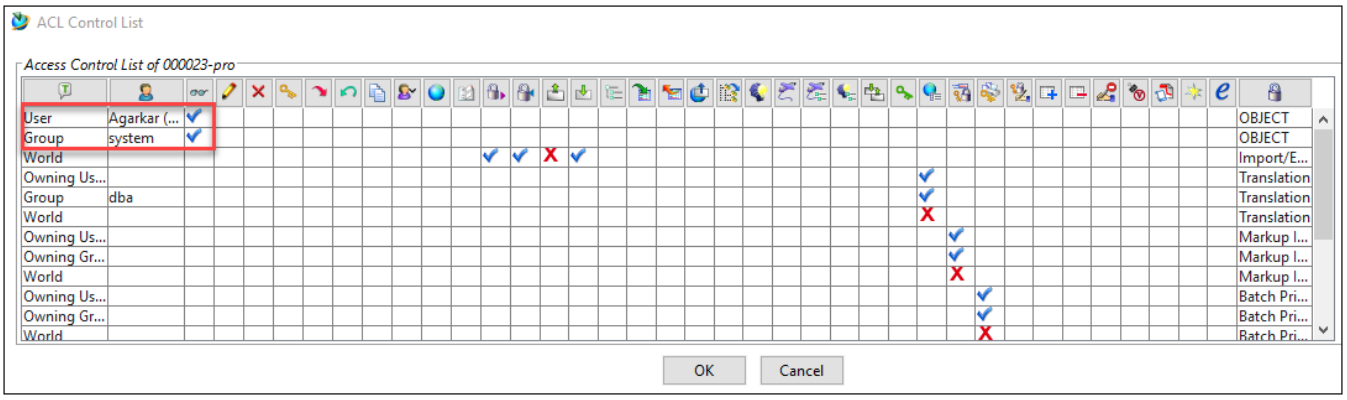
Object-based protection uses access control lists (ACLs) to create exceptions to rules-based protection on an object-by-object basis.

Object ACLs are most useful when you need to:

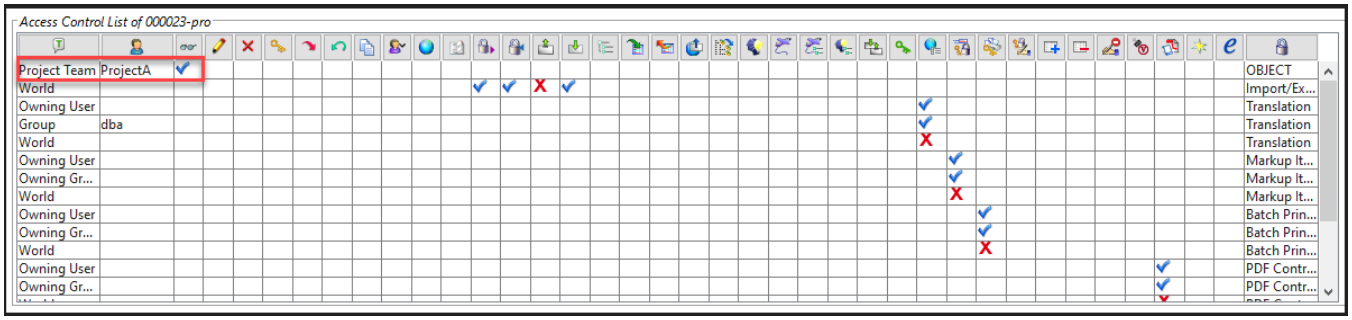
- Grant wider access to a specific object.
- Limit access to a specific object.



Teamcenter uses ACLs to determine access to an object only by users and groups. For example, the object ACLs (**User-Agarkar** and **Group-system**) are valid.



In contrast, the object ACL (**Project Team-ProjectA**) is *invalid*, as object ACLs are not used to determine access to an object by *Project Team*.



Users with proper permissions can override the ACL for an object to grant or deny permissions for certain users but only when the rule tree allows.

For example, the rule tree does not allow object-based access rules to override the rules-based protection when:

- An object has an assigned status.
- The object access rule is granted in a workflow.

> Note:
>
> Object ACLs do not control the creation of objects. They only determine what operations can be performed on existing objects.
>
> - Each object ACL contains a list of accessors and the privileges granted, denied, or not set for each accessor.
> - Each individual pairing of an accessor with their privileges is considered a single access control entry (ACE).

Access control lists

Access control lists (ACLs) contain a list of accessors and the privileges granted, denied, or not set for each accessor. *Accessors* are collections of users who share certain common traits, such as membership in the group that owns the object or membership in the project team. Just as rules have a precedence weighting in the rule tree, **accessor precedence** weighting is considered when the ACL is evaluated.

Each pairing of an accessor with corresponding **privileges** in the list is referred to as an *access control entry (ACE)*. An ACL can be comprised of one or many ACEs.

ACLs are associated with conditions in the rule tree as part of a rules-based security model, and they can be used in more than one rule.

In addition, object ACLs grant exceptions to rules-based protection and are created by users with change privileges.

Access control lists display the current protections for an object.

> Note:
>
> - If an ACL is modified by a user, other users who are logged on at the same time are not affected by the updated ACL until they log off and log on again.
> - ACLs do not control the creation of objects. They only determine what operations can be performed on existing objects.

System Administrator			✓	✓				✓		✓
World	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗

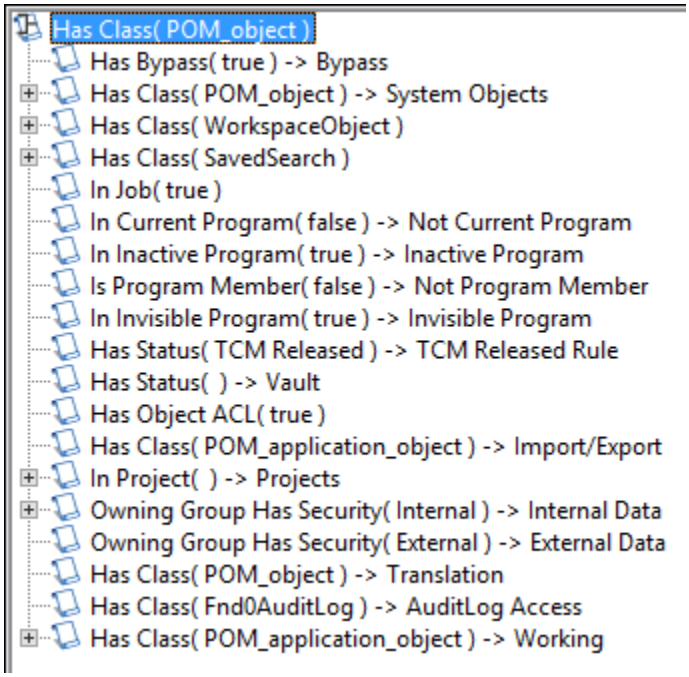
Access Manager rule tree

Rules are organized in the Access Manager rule tree and are evaluated based on their placement within the tree structure. The default rule tree included in your Teamcenter installation assumes that users are granted privileges unless explicitly denied.

The rule tree acts as a filter that an object passes through when a user attempts to access the object. When conditions that apply to the selected object are met, the privileges defined in the ACL are applied.

- The rules are evaluated from the top to the bottom of the tree.
- Rules at the top take precedence over rules at the bottom of the tree.
- Subbranches always take precedence over parent branches in the tree.

The rule tree appears to the left of the Access Manager window.



For a list of default rule conditions, see Access Management Using Rules and ACLs.

How rules work

How rules are defined

Rules are defined by a combination of a condition, a value for that condition, and an access control list (ACL) that grants privileges to accessors.

- The condition and value identify the set of objects to which the rule applies.
- The ACL defines the privileges that are granted to users (accessors) specified in the ACL.

IF *condition = value* is **TRUE**, **THEN** apply ACL to object.

Example ACL

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy
World		✓	✗	✗	✗	✗	✗	✓

Rule syntax

The following syntax applies to rules:

Condition {Value} –> ACL

The parts of the rule can be thought of as an **IF** clause and a **THEN** clause.

- The condition and value supply the **IF** part of the rule and examine the object with Boolean logic.
- The access control list (ACL) supplies the **THEN** part of the rule by describing the access permission.

For example:

Has Type {UGMASTER} –> UG Model

In this example, **Has Type** is the condition, **UGMASTER** is the value, and **UG Model** is the name of the ACL.

Rule evaluation assumptions

When a user attempts to access data, the rule tree is evaluated to determine the privileges to be granted or denied. The following assumptions apply to the evaluation:

- Rules higher in the rule tree are more global in nature and apply to all object types.
- Lower-level rules refine access to more specific objects such as **UGMASTER** datasets. For example:

  Has Class(POM_application_object)

  Has Class(Dataset)

  Has Type(UGMASTER)

- **Precedence** determines the privileges granted. Rule precedence is from top to bottom in the tree, with the highest rule having greatest precedence and the lowest rule having least precedence.
- **Accessor precedence** in the ACL and rule precedence within the tree are both considered when granting access privileges. Accessors have a predefined precedence in the system.

> Note:
>
> The way Access Manager evaluates Master forms does not follow the normal rules. Master forms inherit access privileges from the parent item or item revision, so if you change access privileges to an item or item revision, you affect the privileges on the Master form. You can use the **TC_MASTERFORM_DELEGATE** environment variable to change the default behavior.

Evaluating the rule tree for the effective ACL

The rule tree evaluation results in an *effective ACL*. The effective ACL represents the cumulative compilation of all the named ACLs that apply to the object the user is trying to access.

The rule tree is evaluated as follows:

- Trim rules that do not apply to the object because their conditions are false.

  > Note:
  >
  > The rules are not removed from the tree, but they are ignored during evaluation.

- Evaluate rules in order of precedence, from top to bottom.
- Evaluate the subbranch of a rule before evaluating the parent rule.
- Evaluate subbranch rules in order of precedence, from top to bottom, in the event that there are multiple subbranch rules.

The *effective ACL* is determined by compiling the ACLs in the order that the tree is traversed.

Example rule tree evaluation by order of precedence

This example rule tree shows the order of precedence in the left column, assuming all conditions are met.

- The first two rows are the first two rules evaluated because they are highest in the tree and have no subbranch.
- The third row only gets evaluated after all its subbranches are evaluated.

```
1     Condition {Value} –> Named ACL
2     Condition {Value} –> Named ACL
15  - Condition {Value} –> Named ACL
9      - Condition {Value} –> Named ACL
3            Condition {Value} –> Named ACL
4            Condition {Value} –> Named ACL
7          - Condition {Value} –> Named ACL
5                Condition {Value} –> Named ACL
6                Condition {Value} –> Named ACL
8            Condition {Value} –> Named ACL
14     - Condition {Value} –> Named ACL
10           Condition {Value} –> Named ACL
13         - Condition {Value} –> Named ACL
11               Condition {Value} –> Named ACL
12               Condition {Value} –> Named ACL
```

Example of compiling an effective ACL

When the user attempts to access a **UGMASTER** dataset, the **rule tree** is trimmed to reflect only those rules that apply to the object.

Has Class(POM_object)

Has Class(POM_application_object) –> Working

Has Class(Dataset)

Has Type(UGMASTER) –> UGMASTER

Based on the trimmed rule tree, the effective ACL is compiled by evaluating the tree (from bottom to top) as follows:

1. Find the topmost leaf node in the tree, in this case, **Has Type(UGMASTER) –> UGMASTER**. Add the **UGMASTER** ACL to the effective ACL.

2. Find the next node, **Has Class(Dataset)**. This node has no associated ACL, so it does not contribute to the effective ACL.

3. Find the next node, **Has Class(POM_application_object) –> Working**. Add the **Working** ACL to the effective ACL.

4. Find the next node, **Has Class(POM_object)**. This node has no associated ACL, so it does not contribute to the effective ACL.

The rule tree evaluation results in the following effective ACL.

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy	ACL
Role in Owning Group	Designer		✓					✓	**UGMASTER**
World			✗		✗			✗	**UGMASTER**
Owning User			✓	✓	✓				**Working**
Group Administrator				✓	✓				**Working**
Owning Group			✓						**Working**
System Administrator				✓	✓				**Working**
World		✓	✗	✗	✗	✗	✗	✓	**Working**

The effective ACL is evaluated when a user attempts to access a **UGMASTER** dataset. The lines that do not apply to the user are ignored. For example, if you are a designer in the owning group of the **UGMASTER** dataset, but you are not the owning user, system administrator, or group administrator, the following entries in the ACL are applied when you try to access a **UGMASTER** dataset.

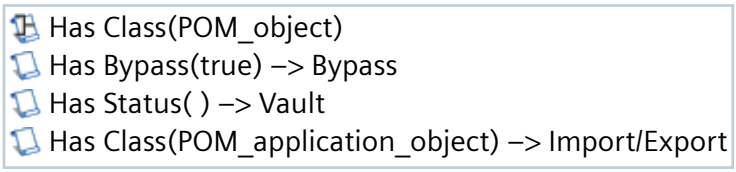
Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy
Role in Owning Group	Designer		✓					✓
World			✗		✗			✗
World		✓		✗		✗	✗	

After the effective ACL is trimmed to include only the entries that apply to the user attempting to access the dataset, the privileges in the remaining ACL entries are evaluated. This is done by working down each privilege column until you encounter a granted ✔ or denied ✘ symbol.

In this example, the privilege evaluation grants the accessor read, write, and copy privileges and denies the accessor delete, change, promote, and demote privileges.

Simple rule tree evaluation example

This simplified view of the default rule tree is used in the following example:



A user, Jim Smith, attempts to open the **MyDataset** text dataset with released status. To perform this action, Jim Smith needs read privileges on the dataset.

The following ACLs are considered when the sample rule tree is evaluated:

1. The **Has Bypass(true) –> Bypass** rule is evaluated. This high-level rule grants system administration privileges to users.

   Result: Jim does not have bypass set, nor is he a system administrator; therefore, this rule condition is false and the **Bypass** ACL is not applied. The evaluation moves down the tree to the next branch.

2. The **Has Status() –> Vault** rule is evaluated. This rule evaluates whether the object has an attached status type. If yes, the **Vault** ACL is applied.

   Result: The **MyDataset** dataset is in released status; therefore, the rule condition is true and the **Vault** ACL is applied.

   Vault ACL

   The **Vault** ACL grants all users read and copy privileges and denies all users write, delete, change, promote, and demote privileges. The **World** accessor represents all users.

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy	CICO
World		✔	✘	✘	✘	✘	✘	✔	✘

3. The **Has Class(POM_application_object_) –> Import/Export** rule is evaluated. This rule evaluates whether the object is of the **POM_application_object** class. If yes, the **Import/Export** ACL is applied to the object.

   Result: All workspace objects, including datasets, are subclasses of the **POM_application_object** class; therefore, the rule condition is true and the **Import/Export** ACL is applied.

   Import/Export ACL

   The **Import/Export** ACL grants all users (world) export, import and transfer in privileges and denies all users transfer out privileges. In addition, this ACL grants remote site users import privileges and denies remote site users transfer in privileges. The **Import/Export** ACL neither explicitly grants or denies read privileges.

Accessor	User	Read	Export	Import	Transfer out	Transfer in
World			✓	✓	✗	✓
Remote Site				✓		✗

Complex rule tree example

This view of the default rule tree is used in the example that follows:

- Has Class(POM_object)
  - Has Bypass(true) –> Bypass
  - In Job(true)
  - Has Status( ) –> Vault
  - Has Object ACL(true)
  - Has Class(POM_application_object) –> Working
    - Has Class(Item) –> Items
    - Has Class(Item Revision) –> Item Revs
    - Has Class(Dataset)
      - Has Type(UGMASTER) –> UGMASTER

A user, Jim Smith (**jsmith**), a designer in the engineering group, attempts to modify the **MyPart UGMASTER** dataset with working status. To perform this action, Jim Smith needs write privileges on the dataset.

The following ACLs are considered when the sample rule tree is evaluated:

1. The **Has Bypass(true) –> Bypass** rule is evaluated. This high-level rule grants system administration privileges to users.

   Result: Jim does not have bypass set, nor is he a system administrator; therefore, this rule condition is false and the **Bypass** ACL is not applied. The evaluation moves down the tree to the next branch.

2. The **In Job(true)** rule is evaluated. This rule evaluates whether the object is in a workflow.

   Result: No ACL is defined, therefore, the condition being true has no effect. The evaluation moves down the tree to the next branch.

3. The **Has Status() –> Vault** rule is evaluated. This rule evaluates whether the object has an attached status type. If yes, the **Vault** ACL is applied.

   Result: The **MyPart** dataset is in working status; therefore, the rule condition is false and the **Vault** ACL is not applied.

4. The **Has Object ACL(true)** rule is evaluated. This rule evaluates whether an ACL exists for the object.

   Result: No object ACL is defined by a user; therefore, the condition is false and has no effect. The evaluation moves down the tree to the next branch.

5. The **Has Class(Item) –> Items** rule is evaluated. This rule evaluates whether the object is of class item. If yes, the **Items** ACL is applied.

   Result: The **MyPart** is of class dataset not item; therefore, the rule condition is false and the **Items** ACL is not applied.

6. The **Has Class(Item Revision) –> Item Revs** rule is evaluated. This rule evaluates whether the object is of class item revision. If yes, the **Items** ACL is applied.

   Result: The **MyPart** dataset is of class dataset not item revision; therefore, the rule condition is false and the **Item Revs** ACL is not applied.

7. The **Has Type(UGMASTER) –> UGMASTER** rule is evaluated. This rule evaluates whether the object is of class UGMASTER. If yes, the Items ACL is applied.

   Result: The **MyPart** dataset is of class **UGMASTER**; therefore, the rule condition is true and the **UGMASTER** ACL is applied.

UGMASTER ACL

The **UGMASTER** ACL explicitly grants write access to users who fill the **Designer** role in the owning group and explicitly denies write access to all other users in the owning group.

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy
Role in Owning Group	Designer		✓					
Owning Group			✗					

8. The **Has Class(Dataset)** rule is evaluated. This rule evaluates whether the object is of class dataset.

   Result: The **MyPart** dataset is of class dataset; therefore, the rule condition is true. No ACL is defined, therefore the condition being true has no effect.

9. The **Has Class(POM_application_object) –> Working** rule is evaluated. This rule evaluates whether the object is of the **POM_application_object** class. If yes, the **Working** ACL is applied to the object.

   Result: All workspace objects, including datasets, are subclasses of the **POM_application_object** class; therefore, the rule condition is true and the **Working** ACL is applied.

   Working ACL

   The **Working** ACL explicitly grants write, delete, and change privileges to owning users and write privileges to the owning group. It also grants delete and change privileges to the group administrator and the system administrator. All other users are granted read and copy privileges and explicitly denied write, delete, change, promote, and demote privileges.

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy
Owning User			✓	✓	✓			
Group Administrator				✓	✓			
Owning Group			✓					

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy
System Administrator				✓	✓			
World		✓	✗	✗	✗	✗	✗	✓

Result: After all the rules are evaluated, the following is the result. Note that the **Working** ACL grants the owning group write permission, but the **UGMASTER** ACL already removed that privilege. The figure also shows the applied named ACL.

Accessor	User	Read	Write	Delete	Change	Promote	Demote	Copy	Named ACL
World									Import /Export
Remote Site									Import/ Export
Role in Owning Group	Designer		✓						UGMASTER
Owning Group			✗						UGMASTER
Owning User			✓	✓	✓				Working
User	tsproxy (tsproxy)		✓	✓					Working
Group Administrator				✓	✓				Working
Owning Group			✓						Working
System Administrator				✓	✓				Working
World		✓	✗	✗	✗	✗	✗	✓	Working

Best practices for rules

- **Understand your organization's business rules.**

  A thorough understanding of your organization's business rules enables you to model access rules that support your business processes and are transparent to users. When modeled correctly, Access Manager rules grant users the privileges required to perform the tasks associated with their jobs while denying them access to data that is released or out of the scope of their functional role.

- **Document the business rules and the rule tree developed to meet them.**

  Every rule in the rule tree and the named ACLs associated with the rules are included for a purpose. For maintenance purposes, Siemens Digital Industries Software strongly recommends that you document the purpose of the rules, how they are populated, and why they have been populated. Future versions of Teamcenter add new rules and accessors. Merging new rules and accessors is a manual process, which is simplified if you have thoroughly documented the Access Manager rule tree.

- **Export the rule tree before and after making changes.**

  When new rules do not work as expected, you must be able to restore an earlier, working version of the rule tree. A backup copy is essential to restoring rules back to their original state.

- **Update your rule tree after changing attributes of organizational and administrative objects.**

  When you change attributes on organizational objects (for example, **User**, **Group**, and **Site**), and administrative objects (**Type** and **Business Rule**), determine if any rules that use these objects are affected by the attribute change and update your rules appropriately.

  For example, if you modify a user ID, this may affect one or more rules that use this organizational object.

- **Add new rules for working data in the Working data branch of the tree.**

  The proper location to add new rules for working data is under the **Working** data branch in the rule tree. This helps you customize your rule tree and identify working data.

  ```
  Has Class(POM_application_object) –> Working
  ```

- **Whenever possible, leave privileges unset.**

  Leaving privileges unset in ACLs allows rules to accomplish focused objectives, and it also allows objects and accessors to filter through rules that do not apply to them.

- **Populate access control lists (ACLs) sparingly.**

  Explicitly grant privileges, and only deny privileges when you must block users from access that would otherwise be implicitly granted.

- **Use the Has Attribute condition to create custom rules based on any attribute of an object of a given class.**

  For example:

  ```
  WorkspaceObject:object_name=*x
  PublicationRecord:security=suppliers
  ```

The class and attribute names are not case sensitive. The attribute type can be **string**, **double**, **integer**, **logical**, or **reference**.

This rule supports custom attributes.

- **When using ip_classification or gov_classification attributes, use the Has IP Classification rule or the Has Government Classification rule, instead of the Has Attribute rule.**

  For example, use the **Has IP Classification** rule:

  ```
  Has IP Classification ( secret ) -> Secret ACL
  ```

  Instead of using the **Has Attribute** rule:

  ```
  Has Attribute ( WorkspaceObject:ip_classification=secret ) -> Secret
  ACL
  ```

- **Use the Has Property condition to create custom rules based on the value of compound properties.**

  For example:

  ```
  Item:my_custom_prop=my_custom_prop_value
  ```

  In this example, **Item** is the type name and **my_custom_prop** is the compound property.

- **Set security precedence.**

  You can embed type-level security rules under project-level security rules to give the type-level security rules higher precedence than the project-level security rules. For example, the project administrator can add a subbranch under the **Has Class (Form)** rule entry to control access to certain form types that contain sensitive data. The rule for the form type is written as follows:

  ```
  Has Class(Form)
      Has Type(Finance) –> finance_acl
  ```

  If your site requires that project-level security rules take precedence over type-level security rules, you must embed project-level security rules under the type-level security rules. However, Siemens Digital Industries Software does not recommend this practice.

- **Define relevant ACL names.**

  ACL names are displayed in the rule tree and in dialog boxes throughout the Teamcenter interface. You can significantly enhance overall usability by defining these names carefully. For example, when creating an ACL for working data, name it according to the data type (for example, item, item revision, or **UGMASTER**) rather than a role name or some other description.

> Note:
>
> ACLs can be referenced in more than one rule.

- **Use discretion in applying the Bypass ACL.**

  The **Bypass** ACL grants all privileges to system administrators who have selected the user **Bypass** setting. Use discretion in applying this ACL.

- **Changing an Item's ID.**

  Not even Bypass status will allow you to change an Item's ID if it is being referenced by another object, like a release status, for example. If you have a need to change an Item's ID, you must remove all references to and from other objects, make your change, and then re-reference everything. For this reason, it is recommended not to make changing Item IDs a part of your business process. If there is a release status attached, then use one of the following methods to remove the reference.

  - The release_man utility with the -**unrelease** argument.
  - Create a workflow which uses the EPM-set-status action handler with the -**action=delete** argument.

- **Do not create GRM relations**

  Do not create Generic Relationship Management (GRM) relationships between Teamcenter business objects, such as BOM View, and Access Manager objects, such as AM Tree, Named ACL, and AM_ACE. Creating such relationships can result in unpredictable behavior with Access Manager during run time.

Cautions for using rule trees

- **Do not modify access control lists (ACLs) referenced by rules on the System Objects branch.**

  Adding new rules, deleting rules, or in any way modifying existing rules on the **Systems Objects** branch of the rule tree may result in unpredictable behavior or loss of data. Modifying the **Systems Objects** branch of the rule tree is not supported unless specifically advised to do so by Siemens Digital Industries Software.

- **Do not modify the upper area of the rule tree.**

  Deleting or changing the order of the branches in this area of the rule tree may result in unpredictable behavior or loss of data.

- **Do not use a text editor to modify rule tree files.**

  Rule tree files are simple ASCII files and conform to a particular format. You can read rule tree files using any text editor; however, modifying them with a text editor can easily corrupt the file.

- **Do not use the infodba account to change object ACLs.**

  It is assumed that objects owned by **infodba** are seed parts or other special-case objects.

What is project-level security?

Project-level security refers to a security scheme based on a combination of projects and access rules in Teamcenter. Projects are entities that correlate groups of users, potentially at different physical sites, with the data associated with a given project or subset of a project.

In many industries, it is typical for a site to own a lead program in the product development environment. Other sites may share a project within the program or add projects to the program. Users must have access privileges to the data that they own as well as to data assigned to the project that they work on that is owned by other sites.

When suppliers are brought into the product development environment, the program owner, in addition to providing access to data owned by the supplier and data assigned to the projects the supplier works on, must also control access to data among the suppliers. For example, Supplier A must not have access to data owned by Supplier B, and conversely Supplier B must not have access to data owned by Supplier A.

What is group-level security?

Groups represent collective bodies of users (group members) who share data. Group members are assigned functional roles within a group. Users can be assigned multiple roles within a group and they can also be members of multiple groups. Groups and roles within groups are often used as the basis for granting access privileges to data in Teamcenter.

Groups can be arranged in a hierarchy, which provides a powerful way of defining access rules for high-level groups that are implicitly inherited by lower-level groups in the hierarchy. However, you can explicitly define access for a lower-level group that is subject to implicitly inherited access rules, and the explicitly defined access rules override the inherited rules for that group.

Group-level security can also be combined with project-level security to control data access privileges for internal users as well as for external users, such as suppliers.

What is authorized data access?

Authorized data access (ADA) is a generic term that applies to the configuration of Teamcenter security for **intellectual property (IP)** data and for data that is deemed military in nature and is, therefore, subject to **International Traffic in Arms Regulations (ITAR)** policies.

ADA controls access to classified data using user clearance and authorizing documents (licenses) that grant limited-time access to specific users or groups of users.

About effectivity and access control

Access Manager allows you to control user access to create and edit effectivity. Effectivity is the point at which an object becomes effective or valid and can be tracked by date or unit number. It is used in structure-based applications to indicate ranges of dates or unit numbers for which the revision is effective.

You can specify a closed-ended or open-ended effectivity and make the effectivity specific to an item.

Date effectivity allows you to specify a valid range of dates for a particular item revision. Unit effectivity allows you to specify a valid range of unit numbers for a particular item revision. It is always specified in the context of an end item to which the units apply. You can specify a discrete, noncontinuous range, if appropriate. To know how you can use rules to determine who can create and edit effectivity, see *Access to effectivity*.

Multi-Site Collaboration considerations

Overview of Multi-Site Collaboration considerations

Multi-Site Collaboration security mechanisms only apply Access Manager at the site level. The remote site's privileges are checked against the owning site's Access Manager rule tree, but access to individual objects at the owning site are not validated against the individual remote user's privileges. An individual remote user's privileges are currently enforced by preferences set at the site protection level. This is an inadequate security mechanism.

Enhanced Multi-Site Collaboration security improves security of multi-site operations by allowing access to remote operations based on a user ID. This extends the current multi-site security mechanism, which applies Access Manager rules only at the site level. The administrator can set the preference, **TC_check_remote_user_priv_from_sites**, to turn on enhanced security for a remote user. This provides the ability to control the user's privileges to perform remote operations while maintaining site-level control.

This security mechanism is optional and is disabled by default. It can be enabled and configured using the preference **TC_check_remote_user_priv_from_sites**. The security checks are implemented in the IDSM server, allowing the enhanced security feature to apply to client sites that do not implement the feature.

If enhanced security is turned on for a remote site, the following IDSM preferences for that remote site are ignored and the AM rule tree is used instead:

- **IDSM_permitted_users_from_site_***site-name*
- **IDSM_permitted_transfer_users_from_site_***site-name*
- **IDSM_permitted_checkout_users_from_site_***site-name*

Remote checkout privilege access

The **Remote Checkout** privilege allows a user to check out objects that are not normally modifiable, such as a released item revision. The intended purpose is to allow additional attachments or other incremental changes that do not require write access to the object itself. If you import and check out an object that is not modifiable, the local object permissions show that you have write access even though it is unmodifiable at the owning site. Any changes to the local object cause the checkin to fail at the owning site.

The primary access check for a remote checkout operation is the **Write** privilege at the owning site. The **Remote Checkout** privilege can be used to allow the remote checkout of objects in which **Write** access is denied. Released objects are the most common example where this is useful. Remote checkout is allowed if the **Write** or **Remote Checkout** privilege is granted at the owning site. A side effect of this special behavior is remote checkout is permitted when the **Remote Checkout** privilege is denied if the **Write** privilege is granted.

An example usage scenario is you have released an item revision at the owning site and you want to be able to run an analysis or tessellation on the replica side, attach the output to the replica, and send the output back to the owning site. Because released objects are write-protected, you cannot remote checkout the revision to do this. The solution is to enable this operation by granting the **Remote Checkout** privilege to the revision at the owning site. Additionally, you need a way to get write access to the replica revision at the replica site, such as by using the bypass rule.

3. Controlling access to working data

About controlling access to working data

Determining who should have privileges to access working data, and which privileges they should be granted, is an essential component of any Teamcenter security implementation. Privileges are assigned based on your company's business practices and are commonly implemented by determining who has responsibility for the data. For example, when determining who should have write access, you can grant it to users in the following categories:

- Owning users

  Granting write access to the owning user indicates that they are ultimately responsible for the content and handling of the data.

- **Owning groups**

  Granting write access to the owning group enables a teamwork approach to creating and maintaining data.

- **Project members**

  Granting write access based on the projects to which data is assigned enables a teamwork approach to creating and maintaining data and allows the data to be easily assigned to projects, upon which access is then defined. It also provides a mechanism to control access for suppliers.

- Authorized users for classified data

  Granting read access based on authorized data access concepts enables you to protect **intellectual property (IP)** and data subject to **International Traffic in Arms Regulations (ITAR)** policies using combinations of user authorization, object classification, and authorizing documents (IP and ITAR licenses).

> Note:
>
> Although multiple users can be granted write privileges to data, the data can only be modified by one user at any given time. This behavior is ensured by implicit and explicit checkout.

About configuring access to working data

Overview to configuring access to working data

Access to working data is configured by adding rules to the **Has Class(POM_application_object) –> Working** branch of the Access Manager rule tree. The **Working** ACL, which is delivered as part of your

Teamcenter installation, configures access to working data of all object types, as shown in the following table.

Owning User			✓	✓	✓				✓	✓	✓											✓
Owning Group			✓								✓											
System Administrator			✓		✓				✓		✓											✓
World		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓								✓			

The **Working** ACL specifies the privileges for the following accessors:

- **World**

  Grants all users (**World**) read and copy privileges to all data and denies write, delete, and other privileges to all other users. This is very important, because it prevents those privileges from being granted by rules elsewhere in the rule tree.

- **Owning User**

  Grants write and delete privileges to all data that the user creates. In addition, the ACL grants **change privilege**, which allows the owning user to create object ACLs for data they own. This is a significant privilege and is often granted only to managers rather than to general users or groups of users.

- **Owning Group**

  Grants write privileges to any user who is in the group that owns the data.

- **System Administrator**

  Grants delete and change privileges to administrators.

Access to effectivity

You can use Access Manager to control who can create and edit effectivity by creating the rules shown.

The release status rule controls write access to the release status and who can attach effectivity to it. This also determines who can initially create effectivity objects. Similarly, the effectivity rule controls who can edit an existing effectivity object.

> Note:
>
> You must ensure that the accessors who are granted write permissions in the release status rule are also specified in the effectivity rule. If the accessors are not specified in the effectivity rule, they cannot create effectivity objects.

For an example of controlling who can set effectivity, see *Controlling access to effectivity example*.

Has Class(POM_application_object) –> Working

Has Class(ReleaseStatus) –> CreateEffectivityUsers

CreateEffectivityUsers ACL

World		✓	✗								
Role	Designer	✓	✓								

Effectivity objects can be write-protected by selecting **Apply Access Manager effectivity protection (Effectivity Protection)** in the **Create or Edit Effectivity** dialog box. If the flag is set to true, users cannot modify effectivity details. This behavior can be customized through the **effectivity_protection** attribute of the **Effectivity** class.

Has Class(POM_object)

Has Class(Effectivity) –> EditEffectivityUsers

Has Attribute (Effectivity:effectivity_protection=0) –> ModifyProtectedEffs

Has Attribute (Effectivity:effectivity_protection=1) –> RemoveProtection

---

ModifyProtectedEffs and **RemoveProtection** are the two ACLs that set effectivity protection:

- **ModifyProtectedEffs**

  List of accessor groups who are given write access to modify effectivity details when the **Effectivity Protection** flag is set to false.

- **RemoveProtection**

  List of accessors who are given write access to modify effectivity details when the **Effectivity Protection** flag is set to true.

EditEffectivityUsers ACL

Role	Designer	✓	✓								
Group	Project Administration	✓	✓								
World		✓	✗								

Guidelines for applying the delete and change privileges

The following guidelines should be considered when applying the delete and change privileges:

- **Delete**

  Delete privilege is generally limited to the owning user. However, you can also grant delete privileges to group administrators and system administrators for the purpose of maintaining the database.

  > Tip:
  >
  > You can establish a folder in which users can place data that they want deleted, and an administrator can be assigned responsibility for deleting the data.

- **Change**

  Caution must be used when granting change privileges. Change privilege allows accessors to define object ACLs that take precedence over rules in the Access Manager rule tree. Therefore, you can subvert the access rules by creating an object ACL granting write privileges to the **World** accessor. For this reason, the **Has Object ACL** condition is placed lower in the rule tree than the **In Job** and **Has Status** conditions.

  If change privileges are granted to the owning user, restrictions on change privileges can be applied for specific data types lower in the tree.

Example of defining access controls on object class, type, and name

You can add rules to the **Has Class(POM_application_object) –> Working** branch to define access privileges at a more granular level, such as by object class and object type.

Rule tree configuration

The following rule tree configuration provides access controls based on object class, type, and name.

- Has Class(Item) –> Items
- Has Class(ItemRevision) –> Item Revisions
- Has Class(Dataset)
  - Has Type(UGMASTER) –> UG Master
  - Has Type(UGPART) –> UG Non Master
    - Has Name(*dwg) –> UG Drawing
    - Has Name(*cam*) –> UG CAM
- Has Class(PSBOMViewRevision)
  - Has View Type(Design) –> Design BOM
  - Has View Type(Manufacture) –> Manufacturing BOM
- Has Class(Form)
  - Has Type(ItemRevisionMaster) –> Master Data
  - Has Type(MM Basic) –> MPR Data
- Has Class(Folder)
  - Has Description(*Library*) –> Library Structure
  - Has Description(Library Node) –> Library Node

Roles used in working data rule tree example

The following roles are used in the working data rule tree example:

- **Designer**

  Designers can edit **UGMASTER** and **UGPART** datasets and BOM view revisions of type **design**. They can also create new item revisions.

- **Draftsmen**

  Draftsmen can only edit **UGPART** datasets which contain drawing files.

- **Production Engineers**

Production engineers can only edit **UGPART** datasets, which are CAM data with the string **cam** in the dataset name. They can also edit BOM view revisions of the view type **Manufacture**.

- **Manager**

  Managers can create new item revisions, and they have change privileges to item revisions.

- **Librarian**

  Librarians can write to folders that have the name **Library**. All other users only have write access to folders with the name **Library Node**.

Access control lists (ACLs)

The ACLs used in the working data rule tree example control privileges to working data.

Items ACL

											
Role In Owning Group	Manager										
Role In Owning Group	Designer										
Role	Marketing										
World											

Item Revisions ACL

											
Role In Owning Group	Manager										
World											

UG Master ACL

Owning User			✓								
Role In Owning Group	Designer		✓								
World			✗		✗						

UG Non-Master ACL

Owning User			✓								
Role in Owning Group	Designer		✓								
Role in Owning Group	Draftsman		✓								
Role in Owning Group	Production engineer		✓								
World			✗								

Design BOM ACL

Owning User			✓								
Role in Owning Group	Designer		✓								
Role in Owning Group	Configurator		✓								
World			✗								

Manufacturing BOM ACL

Owning User			✓								
Role in Owning Group	Production Engineer		✓								
World			✗								

MRP Data ACL

Role in Owning Group	Production Engineer		✓								
World			✗								

Master Data ACL

Owning User			✓								
Role in Owning Group	Designer		✓								
Role in Owning Group	Production engineer		✓								
World			✗								

Library Structure ACL

Role	Librarian		✓		✓						
World								✗			

Library Node ACL

Role	Librarian				✓						
World			✓					✗			

Control access to revision rules

Use Access Manager to control user access to revision rules. You can limit read access to control the users who can see and use a revision rule. You can use this technique to reduce the number of inapplicable revision rules that are presented to ordinary users, or to restrict rules to certain groups of users. You can use write access to control the users who can modify a revision rule.

You can apply an access rule globally to all rules using a class revision rule or other attribute, (for example, **OwningGroup**) if you created the revision rules appropriately. You can add object ACLs to specific revision rules for exception cases. A typical default access rule and rule tree ACL follow:

Access rule:

```
HasClass (RevisionRule) -> Private Rev Rule ACL
OwningGroup (dba) -> Public Rev Rule
```

Private revision rule ACL:

This ACL prevents Teamcenter displaying privately created revision rules to all users. Only the owning user and system administrator have access to the private rule. You can define an entry for **Owning User** that gives access to all users in the owning group. Alternatively, you could add it as an object ACL to the specific rule.

```
Owning User: Read, Write, Delete, Copy, Change
System Administrator: Read, Write, Delete, Change
World: No Read, No Write, No Delete, No Copy, No Change
```

Public revision rule ACL:

This ACL ensures that public revision rules are visible to all users. It also only allows users with a **configuration** role or members of a system administration group to modify public rules. You should control these permissions carefully, as unintended modification of revision rules can have significant consequences.

```
Role = Configurator: Write
System Administrator: Write, Delete, Change
World: Read, No Write, No Delete, No Copy, No Change
```

Example of controlling access to effectivity

This example shows how to configure the Access Manager rule tree to control who can create, modify, or delete effectivity on a release status object. It also explains how the Access Manager effectivity protection is configured and used. Effectivity protection provides a method of preventing modifications to an existing effectivity.

In the example, Company X would like to allow people in the role of engineering planner or supervisor/charge person to be able to create, modify, or delete the effectivity of in-work and pending release status objects. If the release status object is anything other than in-work or pending, no effectivity can be created against that object. Once the effectivity is created, Company X would like to lock it so it cannot be changed, or it can only be changed by a select group of users.

Add the following rules (those in bold) at the bottom of the rule tree:

Has Class(POM_object)

...

...

...

Has Class (ReleaseStatus) –> WorldNoWrite

Has Attribute (ReleaseStatus:name=In-Work) –> CreateEffectivityUsers

Has Attribute (ReleaseStatus:name=Pending) –> CreateEffectivityUsers

Has Class (Effectivity) –> WorldNoWrite

Has Attribute (Effectivity:effectivity_protection=1) –> RemoveEffProtect

Has Attribute (Effectivity:effectivity_protection=0) –> Modify Effectivity Users

---

The ACLs used in the controlling effectivity example control who can set effectivity.

WorldNoWrite ACL

Denies write privilege to the world. It is applied to the **Has Class (ReleaseStatus)** and **Has Class (Effectivity)** branches to prevent anyone from writing to release status or effectivity objects.

World			✗								

CreateEffectivityUsers ACL

Grants write privilege to the roles of engineering planner and supervisor/charge person. The ACL allows users with any of these roles to create new effectivities.

Role	Engineering Planner		✓								
Role	Supervisor/ Charge Person		✓								

AddEffectivityProtection ACL

Grants write privilege to the roles of DBA, engineering planner, and supervisor/charge person.

The ACL allows users with any of these roles to modify any existing effectivity, as long as that effectivity does not have effectivity protection set. It also allows them to set effectivity protection.

Role	DBA		✓								
Role	Engineering Planner		✓								
Group	Supervisor/ Charge Person		✓								

RemoveEffProtect ACL

Grants write privilege to the roles of DBA and **EffectivityProtectionRemover**.

The ACL allows users with any of these roles to remove effectivity protection so the effectivity can be edited.

Role	DBA		✓								
Role	EffectivityProtection Remover		✓								

Note:

- Any accessor in the **CreateEffectivityUsers** ACL that needs to create effectivities must also be in the **AddEffectivityProtection** ACL. This is because to create effectivity you must have write privilege to the release status and effectivity object.
- The Access Manager's role in the **Owning Group** accessor does not apply to nonworkspace objects, so you must use the role accessor. This means any engineering planner or supervisor/ charge person can create and edit effectivity on any pending or in-work release status object.

Using sponsored authentication

Sponsored authentication overview

Sponsored authentication allows an application that is integrated with Teamcenter to log onto Teamcenter using sponsored mode. This functionality uses two different user types:

- Sponsoring user

Allows integration applications to access Teamcenter and perform a task. Audit log information includes information about this user. This user belongs to the **Sponsor** group.

- Sponsored user

  Creates and accesses data. This user has a license and a user ID. Audit log information contains information about actions performed by this user.

Example of sponsored authentication

A typical example of sponsored authentication is when the sponsored user logs onto Teamcenter using the credentials of the sponsoring user. All data creation and data access is performed by the sponsored user. During the Teamcenter session, the audit log is populated with information about the sponsoring user and the actions performed by the sponsored user. When the sponsoring user logs off the session, the session is cleared for the sponsored user.

Before using the sponsored authentication functionality, you must:

- Define a new group, **Sponsor**, with its access rules and privileges (Is Sponsored Mode and Current Group Is). Any member of the **Sponsor** group can sponsor end users (sponsored users). By default, all Teamcenter users are nonsponsorable.

- Use the **TC_POM_SPONSORED_USER** environment variable to identify the user ID of the sponsored user. To set this environment variable to set Joe Gordon as a sponsorable user, enter:

  ```
  TC_POM_SPONSORED_USER=gordonj
  ```

  Note:

  After you set this environment variable for sponsored users, you can run Integration Toolkit (ITK) command line utilities using the sponsoring user ID for the **-u** option:

  utility_name `-u=`*sponsoring_user_id* `-p=`*password* `-g=`*group*

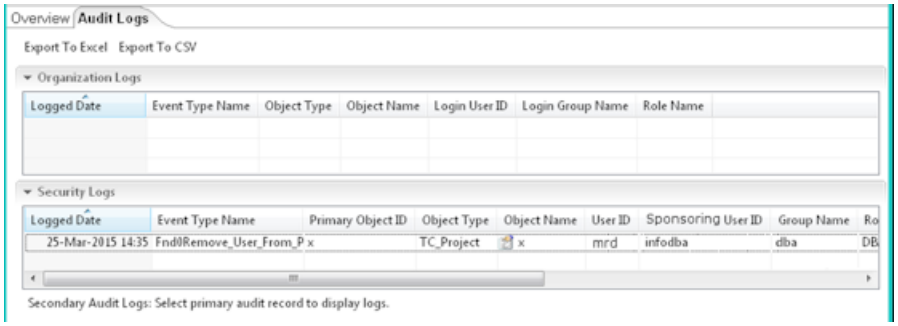
  The *Integration Toolkit Function Reference* is located in the Teamcenter documentation.

- To set a user as sponsorable, run the make_user utility. For example, to make Joe Gordon of the Engineering group a sponsorable user, enter:

  ```
  make_user -u=gordonj -p=gordonj -g=engineering -sponsorable=1 -update
  ```

Security logs track events

Actions performed by both the sponsored user and sponsoring user are logged in the **Security Logs** on the **Audit Logs** tab in Audit Manager.



4. Controlling access to in-process data

About controlling access to in-process data

Access privileges to data that is in process (data that is the target in a workflow process) are controlled using the **In Job** rule condition. Unlike other rule conditions, you do not associate an access control list (ACL) directly with the **In Job** condition. If the condition is evaluated as being true, the system applies the ACL associated with the current task in the workflow process. The system uses the **EPM-set-rule-based-protection** handler to determine the appropriate ACL to be applied.

> Note:
>
> If you associate ACLs directly with the **In Job** condition, they are ignored when the rule tree is evaluated. Only ACLs associated with the workflow process are used to grant access.

The **EPM-set-rule-based-protection** handler passes information to Access Manager to determine which named ACL to use while the associated task handler is current or started. For example, if this handler is placed on the **Start** action of a **Review** task, when the task starts, the named ACL specified in the handler's argument is the ACL used by Access Manager to determine access rights for the target objects of the workflow process. The ACL is applied to the task and all subsequent tasks in the workflow process unless it is changed by another instance of the **EPM-set-rule-based-protection** handler or the process completes.

If the **EPM-set-rule-based-protection** handler is not defined, the system uses the workflow ACL. The current workflow ACL stays in effect until the same workflow sets another ACL later in the process or the process completes.

Workflow ACLs are created in the Workflow Designer application within the context of a specific task and are considered an attribute of the task.

Workflow accessors and privileges

In addition to the **In Job** rule condition, the following accessors and privileges are used to control access to in-process data:

Workflow accessors

- **Approver (RIG)**

  Users who are members of a sign-off team in a workflow process with a specific role in a specific group (RIG).

> Note:
>
> This accessor is used only in workflow ACLs and must match the signoff role-in-group requirements for the release level associated with the workflow ACL.

- **Approver (Role)**

  Users in a specific role who are members of a sign-off team in a workflow process.

  > Note:
  >
  > This accessor is used only in a workflow ACL.

- **Approver (Group)**

  Users in a specific group who are members of a sign-off team in a workflow process.

  > Note:
  >
  > This accessor is used only in a workflow ACL.

- **Approver**

  Users who are members of a sign-off team in a workflow process regardless of their role and group.

  > Note:
  >
  > This accessor is used only in a workflow ACL.

- **Task Owner**

  User who is granted privileges for the task's target data.

- **Task Owning Group**

  Group that is granted privileges for the task's target data.

- **Responsible Party**

  Users responsible for performing a particular task. This ensures that only the user assigned as the responsible party is given privileges to the task's target data.

Workflow privileges

- **Promote**

Specifies whether the accessor is authorized to move a task forward in a workflow process.

- **Demote**

  Specifies whether the accessor is authorized to move a task backward in a workflow process.

Workflow ACL example

The **ApprovalACL** ACL grants privileges to sign-off team members using the **World** accessor (all users) and the **Approver(RIG)** accessor.

											
Approver(RIG)	Engineering Manager in high_performance										
World											

The **World** accessor is explicitly granted read and copy privileges and is explicitly denied write, delete, change, promote, demote, and change ownership privileges.

The **Approver(RIG)** (Engineering Manager in the **high_performance** group) is explicitly granted promote privileges.

In addition, the privileges set for the **World** accessor (with the exception of the promote privilege) are implicitly inherited by the **Approver(RIG)** accessor.

Parallel task and parallel process ACL conflict resolution

When multiple workflows set named ACLs concurrently, the logical **OR** applies to the competing workflow ACLs. To determine privileges allowed to a user of an object in process, the system uses a simplified processing scheme.

All ACLs associated with the object in the workflow process are taken into account.

- When one ACL grants a privilege, access is granted.
- When no ACL grants a privilege, but one or more ACLs denies it, access is denied.
- When no ACL grants or denies the privilege, access is neither granted nor denied.

Workflow access examples

The following examples illustrate possible scenarios in which two tasks compete to apply privileges to the same target object.

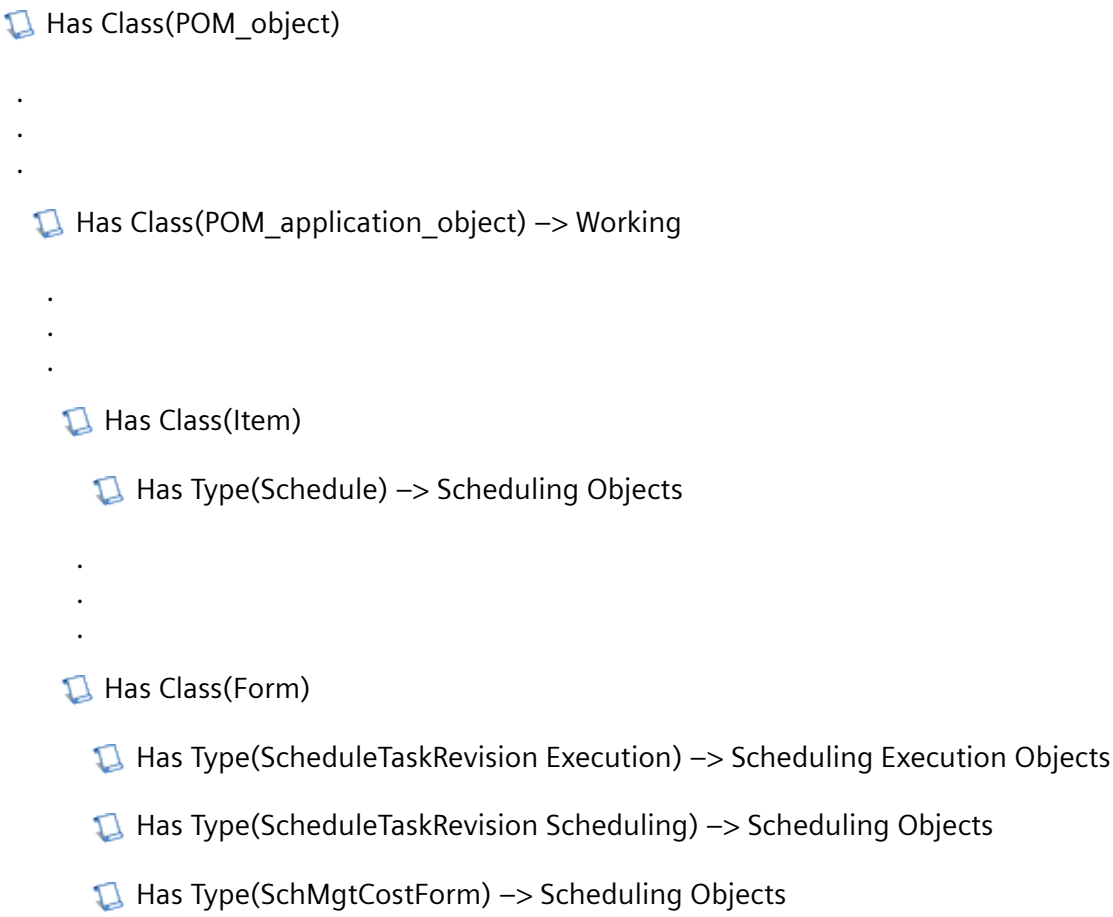
Scenario 1 The user is a member of **Group B**.		
The **Task 1** named ACL grants read privileges to **Group A**.	The **Task 2** named ACL grants read and write privileges to **Group B**.	**Result:** The user is granted read and write privileges.
Scenario 2 The user is a member of **Group B**.		
The **Task 1** named ACL grants read privileges to **Group B**.	The **Task 2** named ACL grants read and write privileges to **Group B**.	**Result:** The user is granted read and write privileges.
Scenario 3 The user is an approver on **Task 2**.		
The **Task 1** named ACL grants read privileges to the **Approver** accessor.	The **Task 2** named ACL grants read and write privileges to the **Approver** accessor.	**Result:** The user is granted read and write privileges.
Scenario 4 The user is an approver on **Task 1** and **Task 2**.		
The **Task 1** named ACL grants read privileges to the **Approver** accessor.	The **Task 2** named ACL grants read and write privileges to the **Approver** accessor.	**Result:** The user is granted read and write privileges.
Scenario 5 The user is the responsible party on **Task 2**.		
The **Task 1** named ACL grants read privileges to the **Approver** accessor.	The **Task 2** named ACL grants read and write privileges to the **Responsible Party** accessor.	**Result:** The user is granted read and write privileges.
Scenario 6 The user is the responsible party on **Task 1** and an approver on **Task 2**.		
The **Task 1** named ACL grants read privileges to the **Responsible Party** accessor.	The **Task 2** named ACL grants read and write privileges to the **Approver** accessor.	**Result:** The user is granted read and write privileges.
Scenario 7		

The user is the responsible party on both tasks.		
The **Task 1** named ACL grants read privileges to the **Responsible Party** accessor.	The **Task 2** named ACL grants read and write privileges to the **Responsible Party** accessor.	**Result:** The user is granted read and write privileges.
Scenario 8 The user is a member of the **task_owner_group** group on **Task 1** and is a member of the **approver_group** group on **Task 2**.		
The **Task 1** named ACL grants read privileges to the **task_owner_group** group.	The **Task 2** named ACL grants read and write privileges to the **approver_group** group.	**Result:** The user is granted read and write privileges.

5. Controlling access to scheduling data

Schedule Manager enables you to plan and track activities in Teamcenter. You can configure Access Manager rules to control which users have the privileges required to access scheduling objects in the database.

The Access Manager rule tree provides default rules for scheduling objects.



The following access control lists (ACLs) are used in the default rules to control access to scheduling objects:

- **Scheduling Objects**
- **Scheduling Fixed Cost**
- **Scheduling Execution Objects**

The **Scheduling Objects** ACL controls access to most scheduling objects and is configured as follows.

Owning User		✓	✓	✓							
Role	Resource Graph Viewers	✓									
System Administrator		✓									
Public Schedule		✓									
World		✗	✗	✗							

The **Scheduling Fixed Cost** ACL controls access to costing forms in a schedule and is configured as follows.

World		✗	✗	✗							

The **Scheduling Bill Rate** ACL controls access to the billing rates. Only users with the **CostDBA** role can change the overall user rates and define rate modifiers, unless you change the ACL.

Role	CostDBA	✓	✓	✓							
Group	dba	✓	✓	✓							
World		✓	✗	✗							

The **Scheduling Execution Objects** ACL controls access to execution objects. Execution objects contain data associated with tasks, such as the actual start date, actual end date, work complete, and percent complete.

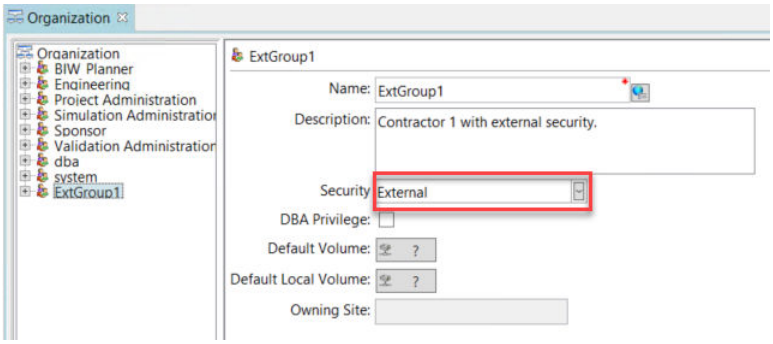
Owning User		✓	✓	✓							
Role	Resource Graph Viewers	✓									
System Administrator		✓									

Public Schedule		✓									
World		✗	✗	✗							

6. Configuring external group security

If you have multiple contractors/suppliers, also referred to as *external groups*, you want to ensure that one contractor/supplier cannot see another. When defining external groups, set the group security to **External**. For example, members of external groups, such as a supplier, can only have access to data in their group.

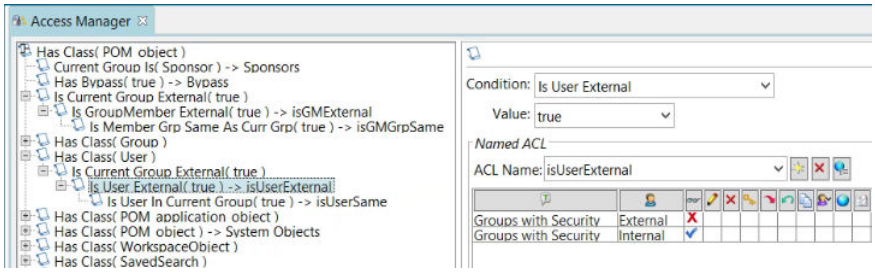
> Note:
>
> If you create subgroups under the external group, you must set security to **External** for each subgroup.



To enforce this security, you can configure Access Manager rules to hide external groups, group members, and users from each other by using the following conditions:

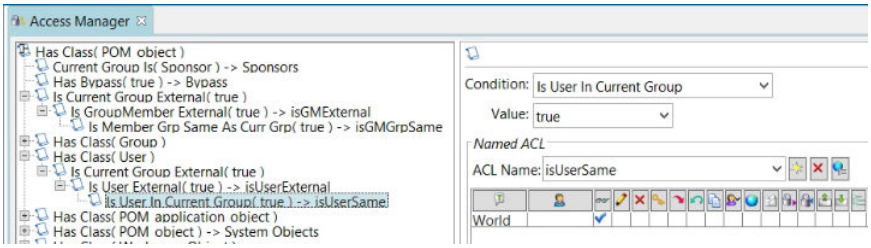
- **Is User External**

  If the user being evaluated is external, you can create an ACL condition that would deny read access. This would deny read access other external users.



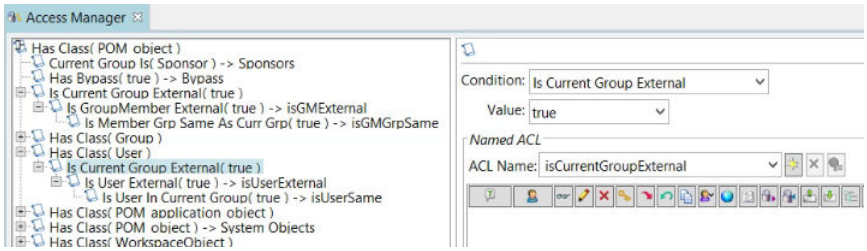
- **Is User In Current Group**

  If the user being evaluated is a member of the current group, you can create an ACL condition that would give read access to the world.



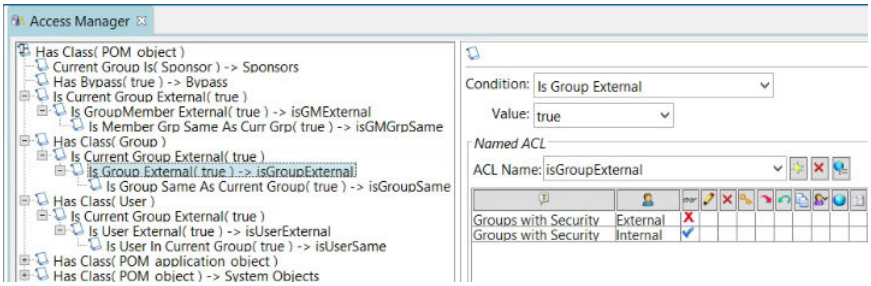
- **Is Current Group External**

  Use this condition to determine if the current logged in user's group has **External** security.



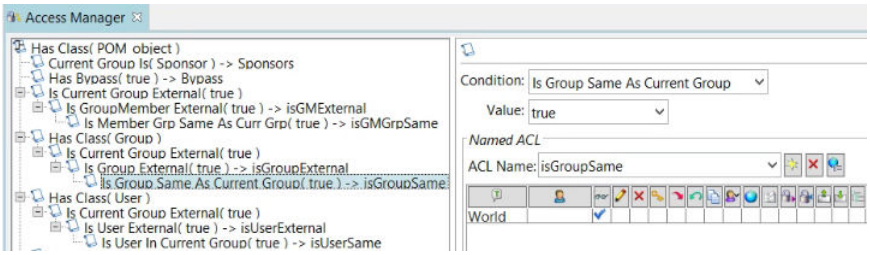
- **Is Group External**

  Use this condition to determine if the group being evaluated has **External** security.



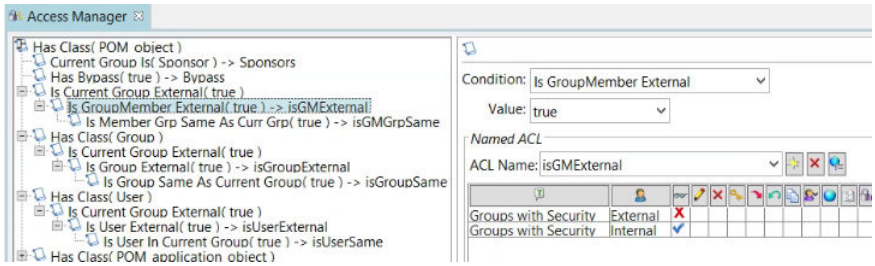
- **Is Group Same As Current Group**

  When the external group being evaluated is the same as the current logged in user's group, only then read access is allowed.



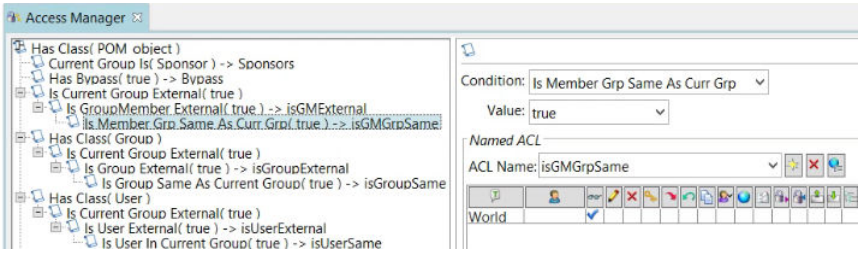
- **Is Group Member External**

If the group member being evaluated has external security, then you can create an ACL condition to deny other external group members access.



- **Is Member Group Same As Current Group**

  If the group of the group member is the same as the current logged in user's group, then you can create an ACL condition to allow read access.



7. Configuring security for remote export and remote checkout

You can gain access to the target data of tasks in your remote inboxes by using the **Remote Checkout** option or by using the **Remote Export** option.

The **Remote Checkout** option provides access to modifiable replicas of the target data associated with the tasks assigned to you. It gives you write access to the replica object and also prevents other users at other sites from modifying the object before you can complete your changes. A reservation is created on the master copy at the owning site. This reservation not only prevents other users from checking out the master copy but also prevents them from transferring site ownership by effectively placing a lock on the master copy.

The **Remote Export** option allows you to read-only replicas of data or to transfer site ownership of the data required to perform your tasks.

The **Remote Checkout** and **Remote Export** options require that certain Access Manager (AM) rules be configured for both the user performing the operation and for the target remote sites.

AM rules related to Multi-Site Collaboration activities are specified in the **Import/Export** ACL, which is configured as follows.

World												✓	✓	✗	✓
Remote Site													✓		✗

The export and transfer out privileges apply to local users at a local site using the **Remote Checkout** and **Remote Export** options. The import and transfer in privileges apply to the remote site that is the target of the operation.

For remote workflow operations, you can create AM rules using the **In Job** condition and the **Import/Export** ACL.

The **Import/Export** ACL grants the following privileges:

- All users have export privileges, which means that anyone can export an object without transferring ownership.
- All remote sites can receive the objects being sent by the operation (import privilege).
- Transfer of ownership is not allowed to any remote site. The transfer in privilege is denied.

8. Configuring group-level security

About configuring group-level security

Groups represent collective bodies of users (group members) who share data. Group members are assigned functional roles within a group. Users can be assigned multiple roles within a group and they can also be members of multiple groups. Groups and users roles within groups can be used as the basis for granting access to data in Teamcenter.

Groups can be arranged in a hierarchy, which provides a powerful way of defining access rules for high-level groups that are then implicitly inherited by groups that are lower in the hierarchy. However, if access is explicitly defined for a lower level group that is also subject to implicitly inherited access rules, the explicitly defined access rules override the implicitly inherited rules for that group.

> Note:
>
> Group inheritance does not apply to the **Owning Group** accessor type.

To better understand the examples, you should first understand the rule conditions, access control lists (ACLs), and accessors that comprise the Access Manager rules, as well as the groups and their specified security levels that are used in the examples.

Rule condition	Description
Owning Group Has Security	Evaluates whether the owning group of the object has a security string. This condition is true only if the security value of the owning group is equal to the value of this condition.
Owning Group	Evaluates whether the object is owned by the group specified in the **group-name** argument of the condition. Wildcard characters can be used with the **Owning Group** condition to allow you to define rules applying to a group and all its subgroups. For example, assume that the **Design** group has two subgroups: **Analysis.Design** and **Development.Design**. By defining a value for the **Owning Group** condition using a wildcard, you can define a general rule to control privileges to all data owned by the **Design** group and its subgroups. Note: Subgroup names, when displayed out of the context of the tree structure, are formatted with the lowest group in the hierarchy listed first and the highest group listed last. For example, the subgroup name

Rule condition	Description
	Analysis.Design indicates that the **Design** group is a parent of the **Analysis** subgroup. For example: Owning Group("*Design) –> design_group_acl

ACL	Description
internal_group_acl	Grants read and modify privileges to the group that owns the data, grants read privileges to other internal groups, and denies privileges to external groups.

Accessor	Description
Owning Group	Group that owns the object. Usually, it is the group of the user creating the object. Additional privileges (for example, write) may be granted to the owning group, because it is common for users to share data with other members of their group.
Groups with Security	Users who have the given security value, either **Internal** or **External**. This value is used to distinguish between groups in the parent company (internal) and suppliers (external).

Groups	Security level
Design	Internal
Development	Internal
Manufacturing	Internal
Supplier 1	External
Supplier 2	External

Example of configuring security to prevent suppliers from viewing internal data

In this example, access to data owned by internal groups is granted to users in internal groups, but access is denied to external suppliers.

Use the following rule that specifies privileges for all data owned by users who are members of internal groups:

Owning Group Has Security(Internal) –> internal_group_acl

The **internal_group_acl** ACL controls read access and also allows the owning user to modify access privileges (define object ACLs for exceptions to the rule-based security).

Owning Group		✓	✓
Groups with Security	Internal	✓	
Groups with Security	External	✗	

Example of configuring security for data owned by a supplier (external data)

In this example, read access to data owned by a specific external supplier is granted to users in internal groups, and access is denied to external groups other than the owning group. The external owning group, supplier, has read and write access to their data.

Use the following rule that defines privileges for all data owned by users who are members of external groups:

Owning Group Has Security("External") –> external_group_acl

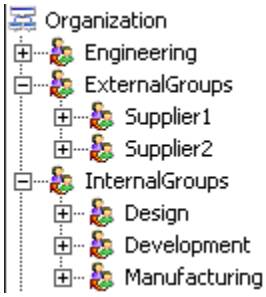
The **external_group_acl** ACL controls read and write privileges for the owning group and grants read privileges to internal groups. External groups are denied read privileges.

Owning Group		✓	✓
Groups with Security	Internal	✓	
Groups with Security	External	✗	

Example of configuring supplier security using hierarchical groups

The security implemented in *Example of configuring security to prevent suppliers from viewing internal data* and *Example of configuring security for data owned by a supplier (external data)* can also be implemented by using the **Owning Group** condition to define a security rule granting all internal users read privileges to data owned by internal groups. To do this, you must first create a hierarchical group

structure (using the Organization application) with one root group containing all internal groups and another root group containing all external groups.



Based on this hierarchical group structure, you can grant all internal groups read privileges to company-owned data and deny supplier groups read privileges to company-owned data by writing the following rule:

Owning Group("*InternalGroups") –> group_read_acl

Owning Group		✓			✓						
Group	InternalGroups	✓									
Group	ExternalGroups	✗									

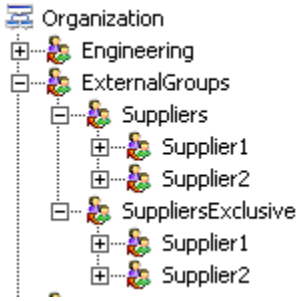
You can set the **Owning Group** condition to define a security rule that grants all external users read privileges to data owned by external groups. Again, this method requires the appropriate hierarchical group structure.

Owning Group("*ExternalGroups") –> suppliers_acl

Owning Group		✓	✓								
Group	InternalGroups	✓									
Group	ExternalGroups	✗									

Example of configuring security for special project data using hierarchical groups (fully restrictive external group security)

In this example, ABC Part Company wants to implement a strict security rule to protect some of the data owned by one of their suppliers. To do this, they extended the hierarchical group structure.



> Note:
>
> Subgroup names, when displayed out of the context of the tree structure, are formatted with the lowest group in the hierarchy listed first and the highest group listed last. For example, the subgroup name **Analysis.Design** indicates that the **Design** group is a parent of the **Analysis** subgroup.

Based on this hierarchical group structure, the following rule can be defined to prevent access to supplier data by users who do not belong to the specific supplier group:

Owning Group("*.SuppliersExclusive.*") –> suppliers_exclusive_acl

> Note:
>
> To make the restricted security of this rule work effectively, the supplier group must only transfer existing data and create new data when they are logged on as a member of one of the subgroups of the **SuppliersExclusive** group.

The **suppliers_exclusive_acl** ACL has the following definition.

Owning Group		✓	✓								
Group	InternalGroups	✗									
Group	ExternalGroups	✗									

Tip:

The security implemented in this example can also be achieved by using the **In Project** condition and adding the data to be protected to a special project.

9. Configuring security for project and program data

About configuring security for project and program data

Project-level security refers to a security scheme based on a combination of projects and access rules in Teamcenter. Projects are entities that correlate groups of users, potentially at different physical sites, with the data associated with a given project or subset of a project.

Program-level security refers to a security scheme based on a combination of projects that are configured to use program security and the corresponding program access rules in Teamcenter.

What are projects and programs?

Projects and programs organize data and are the basis for granting data access to project and program team members. The following concepts apply to projects and programs:

- Only privileged team members or regular team members who are explicitly granted **ASSIGN_TO_PROJECT** or **REMOVE_FROM_PROJECT** privileges can assign data to and remove data from projects and programs.

  The **TC_project_validate_conditions** preference controls which team roles and access privileges are required to add and delete projects.

- Project and program names must be unique within your site. Projects and programs cannot have the same name as any group at the site.

- Data can be assigned to or removed from projects and programs manually or automatically when the data item is created, and items can be assigned to more than one project or program.

- Propagation rules define the associated data that is implicitly assigned to a project or program when a primary item is assigned to the project or program.

- All items in a complete product structure can be assigned to a project or program using the **update_project_bom** utility.

- Ownership of data can be assigned to a project or program by configuring the **autoAssignToProject** extension in the Business Modeler IDE.

- Creation and maintenance of data can be restricted to within the context of a program.

Note:

When the program security attribute on a project is set to **true**, the project is considered to be a program and is subject to program-level access rules.

Programs offer all the basic features of projects, but in addition you can:

- Control access to program data at a higher level than typically applied to project data.
- Share data between programs by assigning the data to multiple programs.

What are groups?

Groups organize users and play an important role in access control for projects and programs. The following concepts apply to groups:

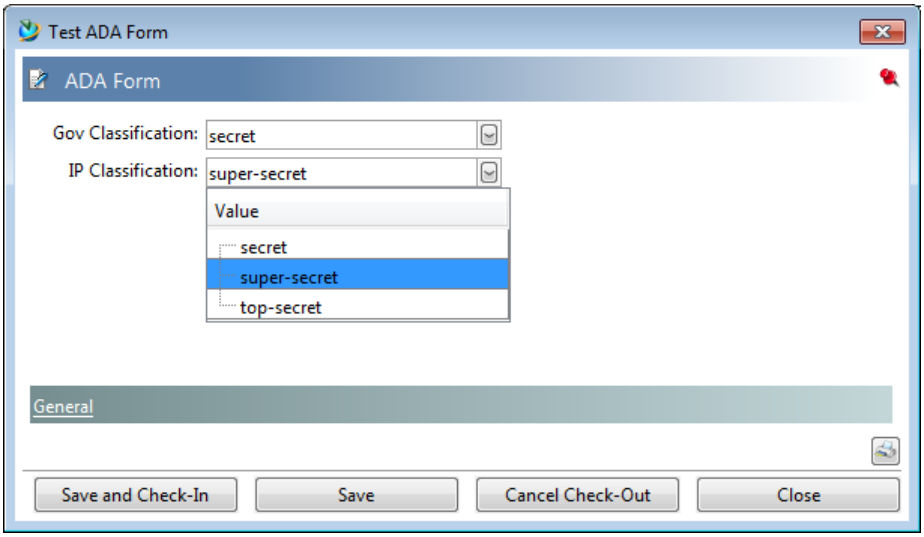
- Access to data is generally determined based on the owning group.
- Groups can be categorized as being **Internal** or **External**. This allows you to differentiate between internal users and suppliers when granting data access. It also allows you to protect supplier data from being accessed by unauthorized internal users or by other suppliers.
- Groups can be structured in a hierarchy that allows access controls to be inherited by subgroups, because members of subgroups are implicitly members of the parent group on which the access controls are implemented.

Note:

Groups are defined and maintained using the Organization application.

Assigning security classification on a workspace object

You can use the Business Modeler IDE to create a custom form (for example, **ADA Form**) containing security classification properties that can be used in a workflow to set classification on an object. The workflow administrator creates a workflow that copies the classification value from the custom form to the target object.



The workflow designer configures the task template to:

- Display the custom form using the **EPM-display-form** handler.
- Create and relate the custom form to the EPM task using the **EPM-create-form** handler.
- Copy the classification value from the custom form to the target object using the **EPM-set-property** handler.

Applying project and program security (Access Manager)

Overview of applying project and program security (Access Manager) rules

Project administrators with **DBA** privileges can extend the default security rules, which grant read access to project or program data to members of the project or program team, on a project-by-project or program-by-program basis.

> Note:
>
> Project administrators with **DBA** privileges only have access to the **In Project() –> Projects** branch of the rule tree.

Using the Project branch in the rule tree, you can:

- Grant or deny access to a particular group of users by applying the **Owning Group** condition.
- Grant or deny access to groups of users based on the group's categorization as internal (OEM) or external (supplier) by applying the **Owning Group Has Security** condition.

- Grant access to data assigned to projects by applying the **In Project** condition.

  > Note:
  >
  > This rule is applied by default to any object assigned to an active project.

- Grant or deny access to users based on their membership in a project by applying the **Is Project Member** condition.
- Grant or deny access to users based on their membership in a program by applying the **Is Program Member** condition.
- Deny users access to data if the owning program is not the active program in the user's session by applying the **In Current Program** condition.
- Deny users access to data if the owning program is inactive by applying the **In Inactive Program** condition.
- Deny users access to data if the owning program is invisible by applying the **In Invisible Program** condition.
- Grant or deny access to program data by applying the **Is Owned By Program** condition.

Preferences related to project and program security

The preferences in the following table affect the way that security rules are evaluated for data in projects and programs.

Preference	Description
AM_PROJECT_MODE	Determines whether the system evaluates roles in all active projects or whether only the role in the user's current project is evaluated.
TC_project_validate_ conditions	Determines how the **Assign to project** and **Remove from project** access privileges are validated in conjunction with privileged membership validation.

Default security rules for project and programs administration

Access rules for projects and programs

The Access Manager rule tree delivered as part of the standard Teamcenter installation includes the following rules related to programs and projects:

- **Is Project Member(true) –> Project Objects**

- In Project() –> Projects
- Is Program Member(true) –> Not Program Member
- In Current Program(false) –> Not Current Program
- In Inactive Program(true) –> Inactive Program
- In Invisible Program(true) –> Invisible Program
- Is Owned By Program() –> Projects

In Current Program rule

The **In Current Program(false) –> Not Current Program** rule denies write, delete, change, and export privileges to users if the owning program of the data is not the active program for the user's session.

The **Not Current Program** ACL denies the privileges to the data in a program, as follows.

		Read				
World			✗	✗	✗	✗

The **World** accessor denies write, delete, change and export privileges to users if the owning program of the data is not the active program for the user's session.

In Inactive Program rule

The **In Inactive Program (true) –> Inactive Program** rule denies write, delete, change and export privileges to users if the owning program of the data is in the inactive state.

The **Inactive Program** ACL denies write, delete, change and export privileges to the data in the program, as follows.

		Read				
World			✗	✗	✗	✗

The **World** accessor denies write, delete, change and export privileges to users if the owning program of the data is in the inactive state.

Is Program Member rule

The **Is Program Member(false) –> Not Program Member** rule denies read access to users if the user is not a member of the owning program or shared program.

The **Not Program Member** ACL denies read access to the data in the program, as follows.

		Read									
World		✗									

The **World** accessor denies read access to users if the user is not the member of the owning program or the shared programs.

> Note:
>
> This rule evaluates to true when the object is owned by a program and the session user is member of the object owning program. Program is basically a project with “program level security” turned on. When the **autoAssignToProject** extension is configured and a program is set in the session, then any object created in that session is owned by that session program. This makes the **Is Program Member** condition apply to the object. It evaluates to true if a given session user is a member of the program.

In Invisible Program rule

The **Is Invisible Program (true) –> Invisible Program** rule denies read access to users if the owning program of the data is in the invisible state.

The **Invisible Program** ACL denies read access to the data in the program, as follows.

		Read									
World		✗									

The **World** accessor denies read access to program data if the program is in the invisible state.

In Project rule

The **In Project() –> Projects** rule grants access to data assigned to projects. This default rule is applied to any object that is assigned to an active project.

The **Projects** ACL grants read privileges to the data in a project, as follows.

		Read									
Project Teams		✓									

The **Project Teams** accessor gives all team members read privileges to the data in a project. For example, if the **Design**, **Validation**, and **Documentation** groups are selected as a project team, the **Project Teams** accessor grants privileges to all members of each group; therefore it is not necessary to use the **Group** accessor to grant privileges to each group individually.

The project administrator can create or modify project security rules to meet the requirements of a specific project by creating a new named ACL for the project or by adding rules under the **In Project** condition in the rule tree. Project administrators can modify rules using Project.

> Note:
>
> Only an administrator with privileges to use Access Manager can change the placement of the **In Project** rule in the AM rule tree. They can also modify the order of the child nodes of the **In Project** branch of the rule tree.

Is Project Member rule

The **Is Project Member(true) –> Project Objects** rule specifies whether the user's membership in the project is evaluated. This condition is true only when the user is a current member of the project.

> Note:
>
> The **Is Project Member(true) –> Project Objects** rule can only be modified by an administrator using the Access Manager application. It cannot be modified from the Project application.

The **Project Objects** ACL grants project administrators and project team administrators privileges to modify *projects in which they are members*. These privileges apply to the project metadata, not to the data assigned to projects.

The ACL is defined as follows.

Accessor Type	Accessor ID	Read	Write	Delete	Change	Change Ownership
Owning User		✓	✓	✓	✓	✓
Regular project member		✓				
Administrator project member		✓	✓	✓	✓	✓
Team Admin project member		✓				
Privileged project member		✓				
World		✗	✗	✗	✗	✗

Note:

You can modify the **Project Objects** ACL to meet the project access requirements at your site.

Is Owned by Program rule

The **Is Owned By Program() –> Projects** rule grants or denies access to data based on program or project ownership.

The **Projects** ACL grants read access to the data in the program, as follows.

		Read									
Project Teams		✓									

The **Is Owned By Program** rule can be configured to Enable the exchange of Aerospace and Defense program data between databases. Exchanging program data requires that the user initiating the import or export is a member of the program to which the objects being exchanged are assigned.

Project-level security based on groups

Project-level security is more flexible than group-level security, because data can be added to and removed from projects without requiring the data properties to be changed or the ownership of the data to be transferred from one group to another. However, projects can be used in conjunction with groups

when developing a security solution. Access controls based on projects and groups can be applied in a Multi-Site Collaboration environment.

Typically, three Access Manager rule conditions are applied to grouped data when configuring project-level security:

- **Owning Group(***group-name***)**

  Defines security on data based on group ownership and hierarchical group behavior, which dictates that subgroups inherit the access controls defined for parent groups.

- **Owning Group Has Security(***group-name***)**

  Defines security on data based on ownership by groups with a specified security categorization, either **Internal** or **External**.

- **In Project(***project-id***)**

  Defines security on the data owned by multiple groups with different security categories.

Granting user-based access to project data

Introduction to granting user-based access to project data

To better understand the examples, you should first understand the rule conditions, access control lists (ACLs), and accessors that comprise the Access Manager rules used in the examples.

> Note:
>
> Rule conditions and accessors are supplied as part of your Teamcenter installation. The ACLs described in this section must be created manually; however, the **Projects** ACL, which is associated with the **In Project** rule condition, is delivered as part of your Teamcenter installation. This ACL grants read access to project teams.

Rule condition	Description
In Project	Specifies a project to which the object must be assigned. The condition is evaluated as being true when the active project to which the object is assigned matches the project specified for this rule condition. If you use an empty string as the value for this condition, the condition is deemed true if the object is assigned to any active project.
In Current Project	Specifies the project ID against which the object is evaluated. The condition is evaluated as being true when the object is in the current active project of the logged-on

Rule condition	Description
	user, and the project ID of the current project matches the value for this condition.

ACL	Description
project_acl	Grants read privileges to the **Project Teams** accessor.
current_project_acl	Grants write and delete privileges to the **Project Teams** accessor.
world_read_revoked_acl	Denies read access to project data to any user who is not a project team member.
combo_project_acl	Grants read privileges to the **Project Teams** accessor and grants write and delete privileges to the **Current Project Teams** accessor.

Accessor	Description
Current Project Teams	Users who are members of current project teams. Applicable only when the object is in the current project of the team members and the current project is active.
Project Teams	Team members in any active project to which the object is assigned.
World	All users regardless of group, role, or project membership.

Configuring user-based access to project data

The ABC Part Company wants to secure their design data to grant members of project teams access to the data assigned to a new project. Users who are members of any project should have read access to the new project data. However, write and delete access must be restricted to users who are actively working on the new project. There are two options for accomplishing this security objective: option 1 implements three rules based on the **In Project()** condition with corresponding ACLs. Option 2 implements one rule based on the **In Project()** condition and a single ACL that grants privileges.

Option 1

In Project() –> project_acl

In Project() –> current_project_acl

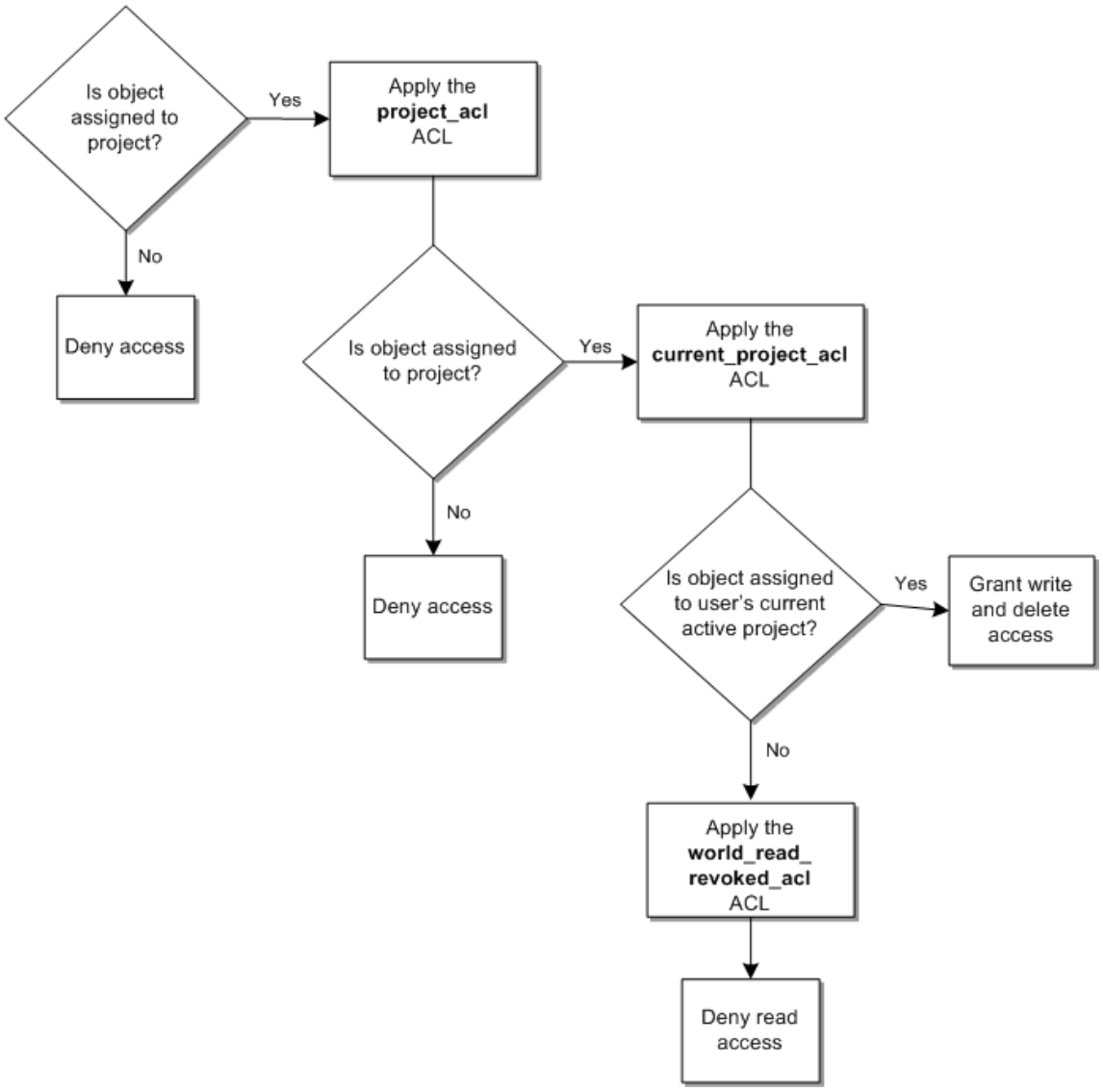
In Project() –> world_read_revoked_acl

The **In Project() –> project_acl** rule is placed higher in the rule tree than the **In Project() –> current_project_acl** rule; therefore, it is evaluated first, as shown in the flowchart.

> Note:
>
> The **In Project** condition can specify a project. However, in this example the value is null; therefore, the object is evaluated for membership in any project.

If the conditions for the **In Project() –> project_acl** rule are met, the **In Project() –> current_project_acl** rule is evaluated. The **current_project_acl** ACL defines privileges for a single accessor, **Current Project Teams**.

If the user accessing the data is not a project team member, the **In Project() –> world_read_revoked_acl** rule is evaluated. The **world_read_revoked_acl** denies read access to any user who is not a project team member.



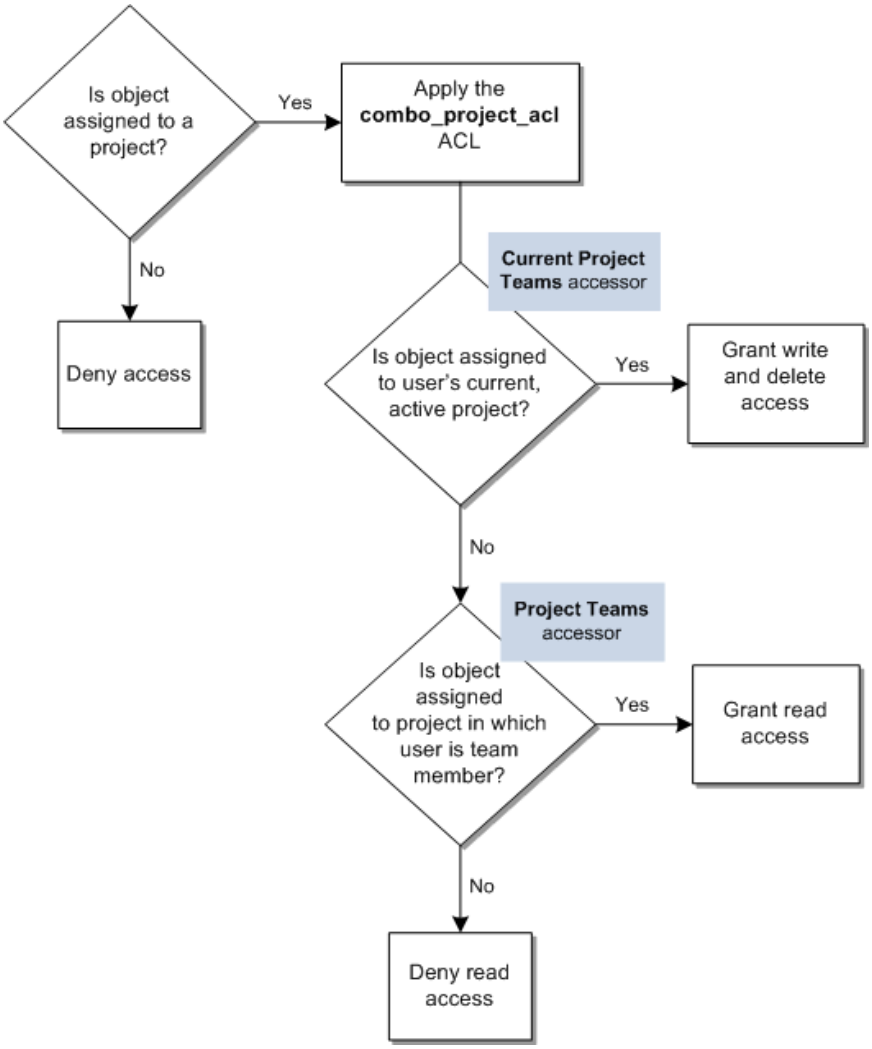
Option 2

In Project() –> combo_project_acl

The **In Project() –> combo_project_acl** rule is evaluated, as shown in the flowchart,

> Note:
>
> The **In Project** condition can specify a project. However, in this example the value is null; therefore, the object is evaluated for membership in any project.

The **combo_project_acl** ACL is then evaluated if the object is assigned to a project. The **combo_project_acl** ACL defines privileges for two accessors, **Current Project Teams** and **Project Teams**.



Granting role-based access to projects

Overview to granting role-based access to projects

To better understand the examples, you should first understand the rule conditions, access control lists (ACLs), and accessors that comprise the Access Manager rules used in the examples.

Rule condition	Description
In Project	Specifies a project to which the object must be assigned. The condition is evaluated as being true when the active project to which the object is assigned matches the project specified for this rule condition. If you use an empty string as the value for this condition, the condition is deemed true if the object is assigned to any active project.

ACL	Description
role_based_acl	Grants privileges to users based on their role in specific projects.

Accessor	Description
Role in Projects of Object	Users who have a specific role in one of the projects of the object. This accessor is affected by the values set in the **AM_PROJECT_MODE** preference. If this preference is not set, only the current active project of the logged on user is evaluated. If the preference is set to **true**, all of the logged on user's active projects are evaluated. It is effective only when the user is logged on with the specified role in the current project, and the current project is one of the projects assigned to the defined object.
Role in Project	Project members with a specific role in a specific project. This is affected by the values set in the **AM_PROJECT_MODE** preference.

Configuring role-based access to project data

The ABC Part Company is developing two new products and has created two projects, **Proj6000** and **Proj7000**, to organize the work in Teamcenter, and they plan to implement security based on membership in those projects. Rather than granting user-based access, they plan to implement access controls based on the roles to which users are assigned.

Users who fill the role of **Designer** (in the **Product Design** group) or **Checker** (in the **Validation** group) must have access to the data associated with both projects. Designers require read and write access to the data, while checkers require only read access.

Initially, two users are assigned to the project: Lois Parker, who is a designer, and John Smith, who is a checker.

There are two options for implementing this security: **In Project() –> project_role_acl** and **In Project() –> project_objects_acl** rules.

> Tip:
>
> In this example, Option 1 can result in an ACL with a large number of entries. Defining access in terms of a specific role within a specific project can be cumbersome because many combinations of roles and projects must be considered. Option 2, defining access in terms of a specific role in one of the projects to which an object is assigned, results in a more manageable ACL, because the number of roles is generally smaller than the number of projects to which an object is assigned.

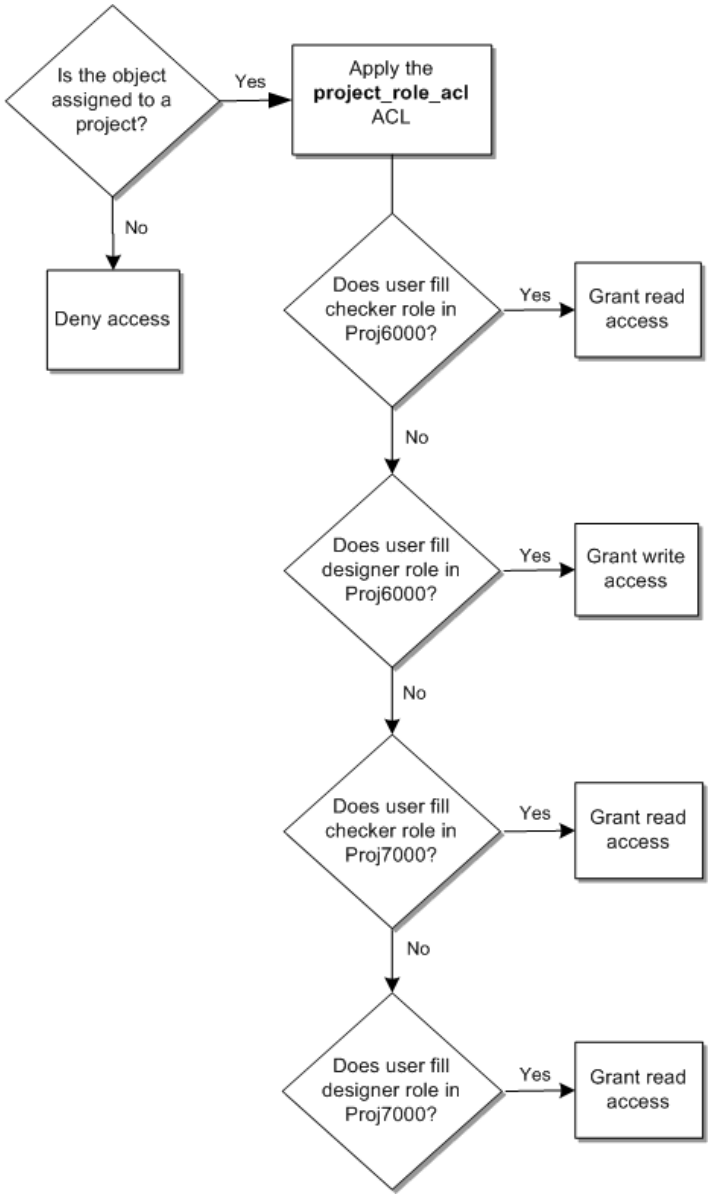
Option 1

In Project() –> project_role_acl

Accessor	Role										
Role in Project	Checker in Proj6000	✓									
Role in Project	Designer in Proj6000		✓								
Role in Project	Checker in Proj7000	✓									
Role in Project	Designer in Proj7000		✓								

Based on the **Project()** condition and **project_role_acl** ACL, the rule is evaluated as shown in the flowchart.

> Note:
>
> The **In Project** condition can specify a specific project. However, in this example the value is null so the object is evaluated for membership in any project.
>
> If the object is assigned to a project, the **project_role_acl** ACL is applied. The **project_role_acl** ACL defines privileges for a single accessor, **Role in Project**. However, the **Role in Project** accessor is used in multiple ACEs to reflect both roles in both projects.



Option 2

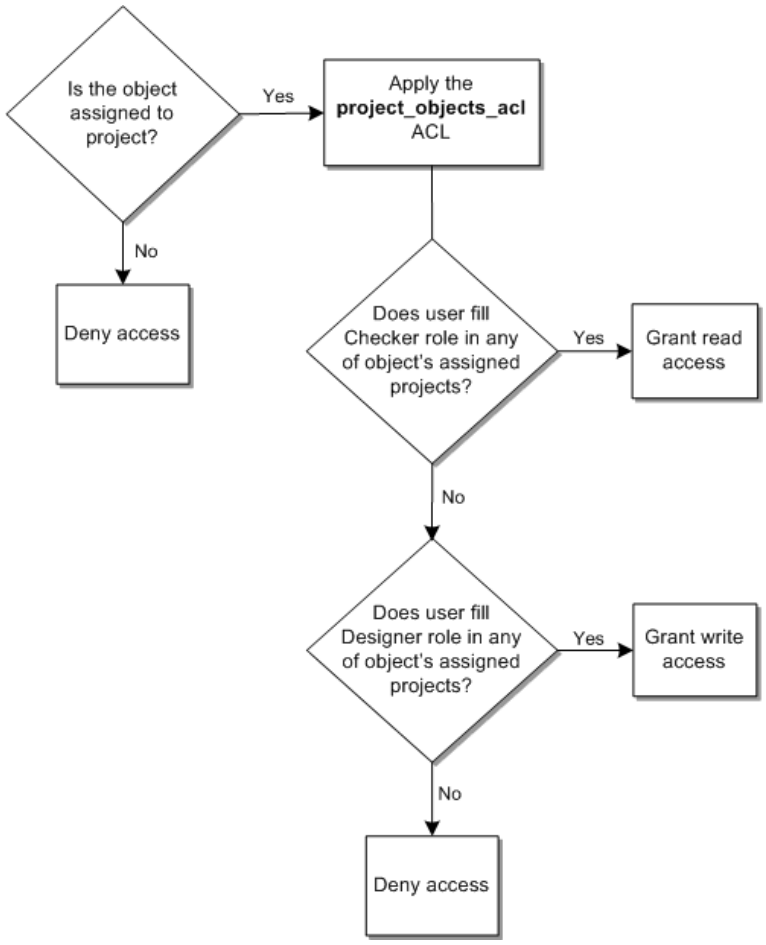
In Project() –> project_objects_acl

Accessor	Role										
Role in Projects of Object	Checker	✓									
Role in Projects of Object	Designer		✓								

Based on the **Project()** condition and **project_objects_acl** ACL, the rule is evaluated, as shown in the flowchart.

> Note:
>
> The **In Project** condition can specify a specific project. However, in this example the value is null so the object is evaluated for membership in any project.

If the object is assigned to a project, the **project_objects_acl** ACL is applied. The **project_objects_acl** ACL defines privileges for a single accessor, **Role in Projects of Object**. However, the **Role in Projects of Object** accessor is used in two ACEs to reflect both roles.



Configuring security when a user is a privileged member of multiple projects

Frank Jones, a designer at ABC Part Company, is a privileged member of two projects, **Project A** and **Project B**. According to the default project security rules, Frank, as a privileged team member, is allowed to assign and remove objects from both projects.

Project A contains sensitive data that must only be viewed by members of the project. If Frank assigns an object in **Project A** to **Project B**, the members of **Project B** can view the sensitive data. To prevent objects in **Project A** from being assigned to **Project B**, the project administrator applies rules and ACLs that include the **Assign to project** and **Remove from project** accessor privileges.

The **TC_project_validate_conditions** preference controls how the **Assign to project** and **Remove from project** accessor privileges are evaluated in conjunction with privileged membership validation and can be configured to control access as follows:

- The user is required to be a privileged project member.
- The user is required to be either a privileged project member *or* have **Assign to project** and/or **Remove from project** privileges.
- The user is not required to be a privileged project member but must have **Assign to project** and/or **Remove from project** privileges.
- The user is required to be a privileged project member *and* have **Assign to project** and/or **Remove from project** privileges.

Assign to project and **Remove from project** are object privileges that are assigned to a user on a per-object basis.

The **In Project() –> Projects** rule restricts the privileges to assign and remove projects to the owning user.

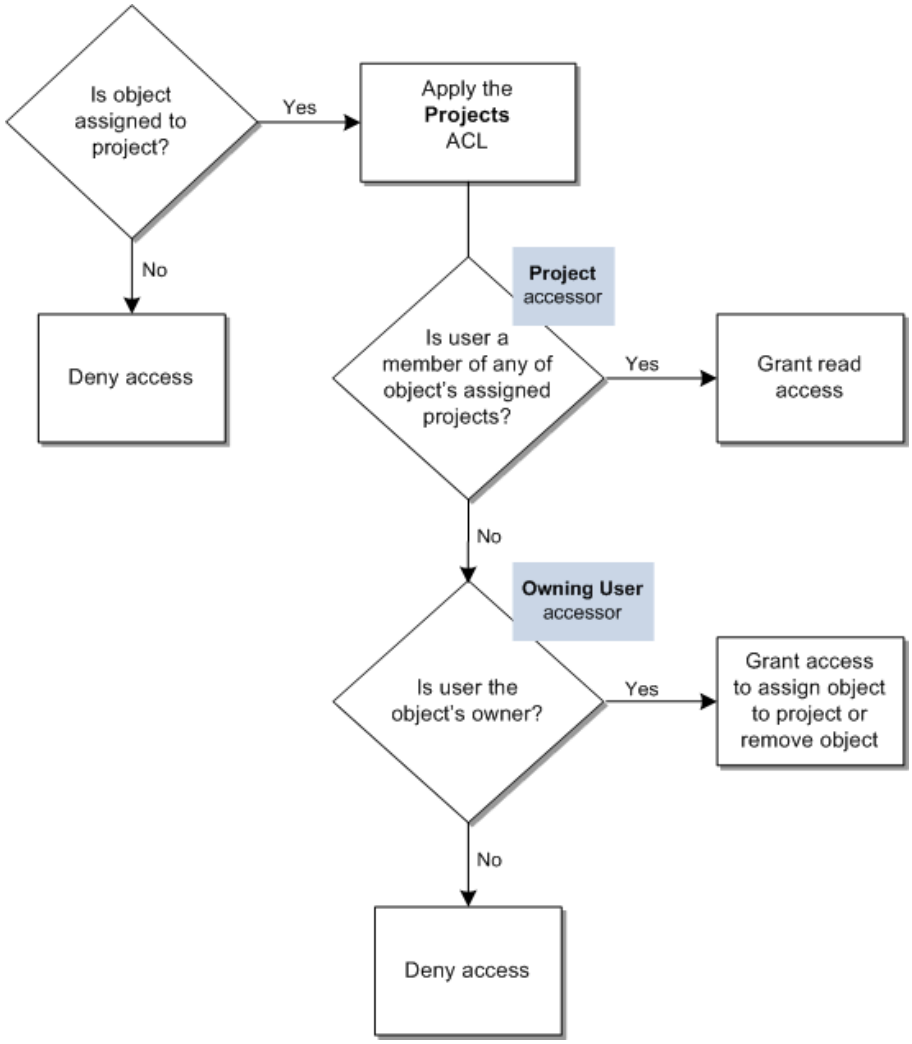
In Project() –> Projects

Accessor																			
Project Teams		✓																	
Owning User																	✓	✓	
World																	✗	✗	

Based on the **Project()** condition and **Projects** ACL, the rule is evaluated, as shown in the flowchart.

> Note:
>
> The **In Project** condition can specify a specific project. However, in this example the value is null so the object is evaluated for membership in any project.

If the object is assigned to a project, the **Projects** ACL is applied. The **Projects** ACL defines privileges for a three accessors, **Project Teams**, **Owning User**, and **World**.



Configuring security to protect competitive data when multiple suppliers are members of a common project

To prevent a supplier group from accessing the data of another supplier when both suppliers work on the same project, you can define the following rules.

- **In Project (“***project-ID*”) **–> default_project_acl**
  - **Owning Group Has Security (“External”) –> supplier_exclusive_acl**

The **default_project_acl** grants read privileges to all members of the specified project. The **supplier_exclusive_acl** ACL has the following definition.

Owning Group		✓	✓								
Groups with Security	Internal	✓									
Groups with Security	External	✗									

The **supplier_exclusive_acl** ACL grants read privileges to the data of the supplier group with an **Internal** security designation and denies read privileges to groups with an **External** security designation.

Implementation considerations for project-level security

Placement of rules in the Access Manager rule tree

The **In Project()** access rule is typically placed above the **Owning Group()** rule or **Owning Group Has Security()** rule in the Access Manager rule tree. This places a higher level of precedence on access based on the project than on access based data ownership.

Set security precedence

You can embed type-level security rules under project-level security rules to give the type-level security rules higher precedence than the project-level security rules. For example, the project administrator can add a subbranch under the **Has Class (Form)** rule entry to control access to certain form types that contain sensitive data. The rule for the form type is written as follows.

Has Class(Form)

Has Type(Finance) –> finance_acl

If your site requires that project-level security rules take precedence over type-level security rules, you must embed project-level security rules under the type-level security rules. However, Siemens Digital Industries Software does not recommend this practice.

Program security examples

About the program-level security examples

The examples in this section are based on the following rules, which are presented in the hierarchical order in which they appear in the rule tree:

In Current Program(false) –> Not Current Program

Is Program Member(false) –> Not Program Member

In Project() –> Projects

Example 1 — Grant read access to only team members of object's assigned projects

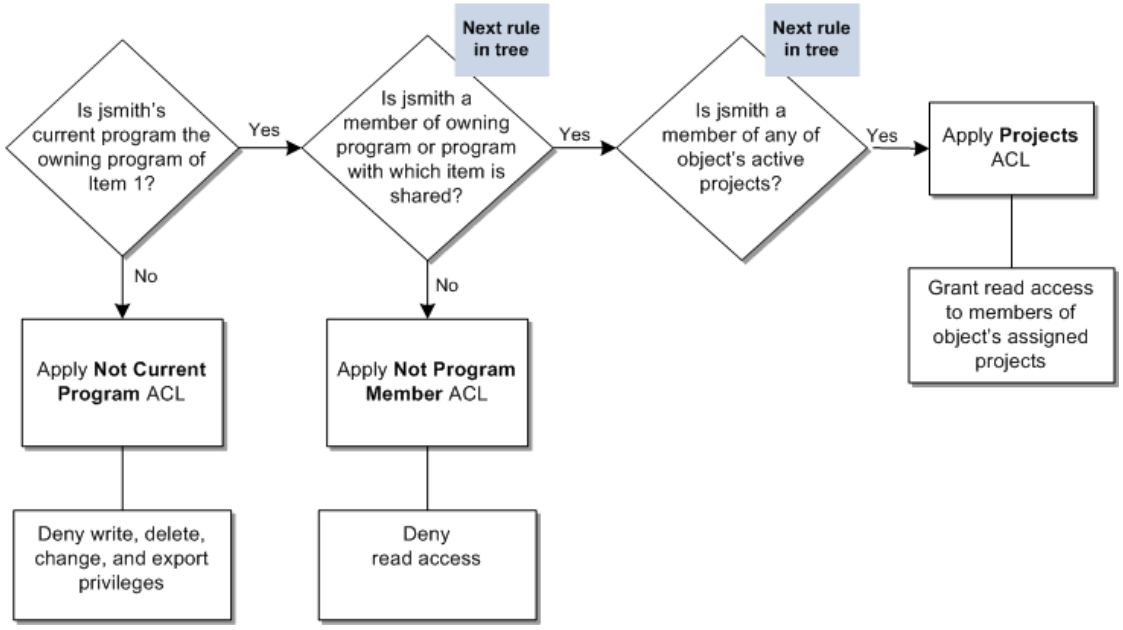
The following assumptions apply to this example:

- Item 1 is being accessed by jsmith, who:
  - Owns Item 1.
  - Is a member of Program 1 and Program 2.
- Item 1 is owned by Program 1 and is not shared with any other projects.
- jsmith's current program in the Teamcenter session is Program 1.

The **In Current Program(false) –> Not Current Program** rule is placed higher in the rule tree than the **Is Program Member(false) –> Not Program Member** rule; therefore, it is evaluated first, as shown in the flowchart.

Because jsmith's current program is the owning program of Item 1, the **Is Program Member(false) –> Not Program Member** rule is evaluated. The **Not Program Member** ACL denies read access to any user who is not a member of the project.

Because jsmith is a member of the owning program, the **In Project() –> Projects** rule is evaluated. The **Projects** ACL grants read access to members of any project to which the object is assigned.



Example 2 — Grant read access to all team members

The following assumptions apply to this example:

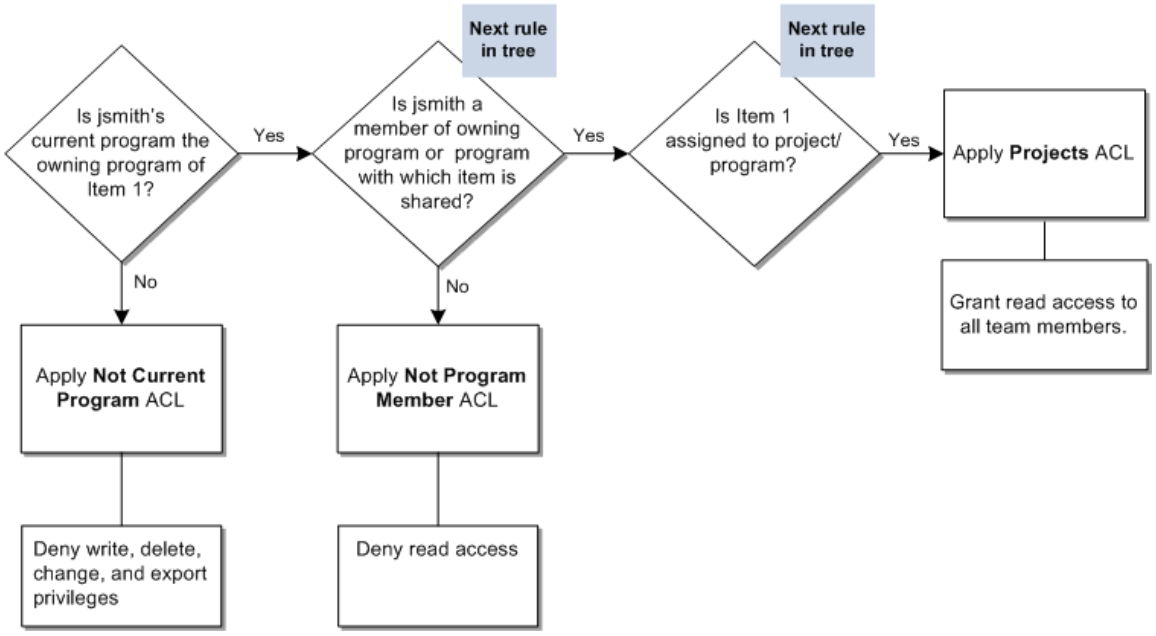
- Item 1 is being accessed by jsmith, who:
  - Owns Item 1.
  - Is a member of Program 1 and Program 2.
- jsmith's current program in the Teamcenter session is Program 2.
- Item 1 is owned by Program 1 and is not shared with any other programs.

The **In Current Program(false) –> Not Current Program** rule is placed higher in the rule tree than the **Is Program Member(false) –> Not Program Member** rule; therefore, it is evaluated first, as shown in the flowchart. jsmith's current program is not the owning program of Item 1; therefore, jsmith is denied write, delete, change, and export privileges.

The **Is Program Member(false) –> Not Program Member** rule is evaluated next. Because jsmith is a member, read access to Item 1 is granted and the next rule in the tree **In Project() –> Projects** is evaluated.

Because Item 1 is assigned to a project/program, the **Projects** ACL is applied. The **Projects** ACL grants read access to the **Project Teams** accessor. The **Project Teams** accessor gives all team members read privileges to the data in a project. For example, if the Design, Validation, and Documentation groups are selected as a project team, the **Project Teams** accessor grants privileges to all members of each group;

therefore it is not necessary to use the **Group** accessor to grant privileges to each group individually. If not, read access is denied.



Example 3 — Grant read access to item to all team members to all data in project

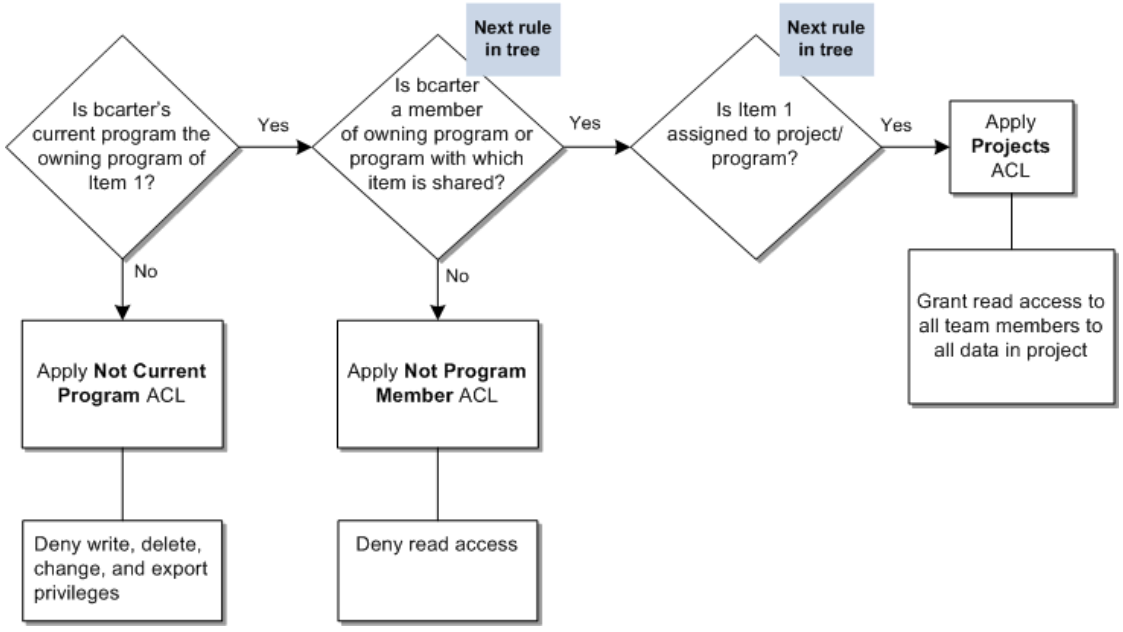
The following assumptions apply to this example:

- Item 1 is being accessed by bcarter, who:
  - Does not own Item 1.
  - Is a member of Program 1.
- bcarter's current program in the Teamcenter session is Program 1.
- Item 1 is owned by Program 1 and is not shared with any other programs.

The **In Current Program(false) –> Not Current Program** rule is placed higher in the rule tree than the **Is Program Member(false) –> Not Program Member** rule; therefore, it is evaluated first, as shown in the flowchart.

bcarter's current program is the owning program of Item 1; therefore, the **Is Program Member(false) –> Not Program Member** rule is evaluated.

bcarter is a member of the owning program; therefore, the **In Project() –> Projects** rule is evaluated and bcarter is granted read access to Item 1.



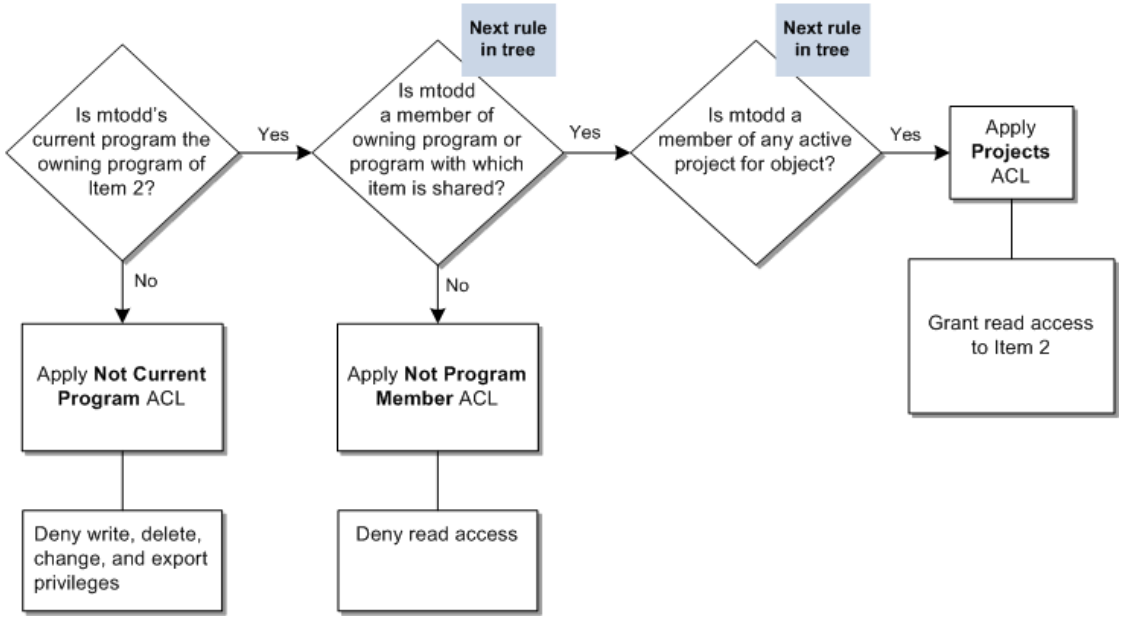
Example 4 — Grant access to single item

The following assumptions apply to this example:

- Item 2 is being accessed by mtodd, who:
  - Does not own Item 2.
  - Is a member of Program 1.
- mtodd's current program in the Teamcenter session is Program 1.
- Item 2 is owned by Program 1 and is shared with Program 2.

The **In Current Program(false) –> Not Current Program** rule is placed higher in the rule tree than the **Is Program Member(false) –> Not Program Member** rule; therefore, it is evaluated first, as shown in the flowchart below. mtodd's current program is not the owning program of Item 2; therefore, mtodd is denied write, delete, change, and export privileges.

The **Is Program Member(false) –> Not Program Member** rule is then evaluated. mtodd is a member of the owning program or a program with which the item is shared; therefore, the **In Project() –> Projects** rule is evaluated and the **Projects** ACL is applied. mtodd is granted read access to Item 2.



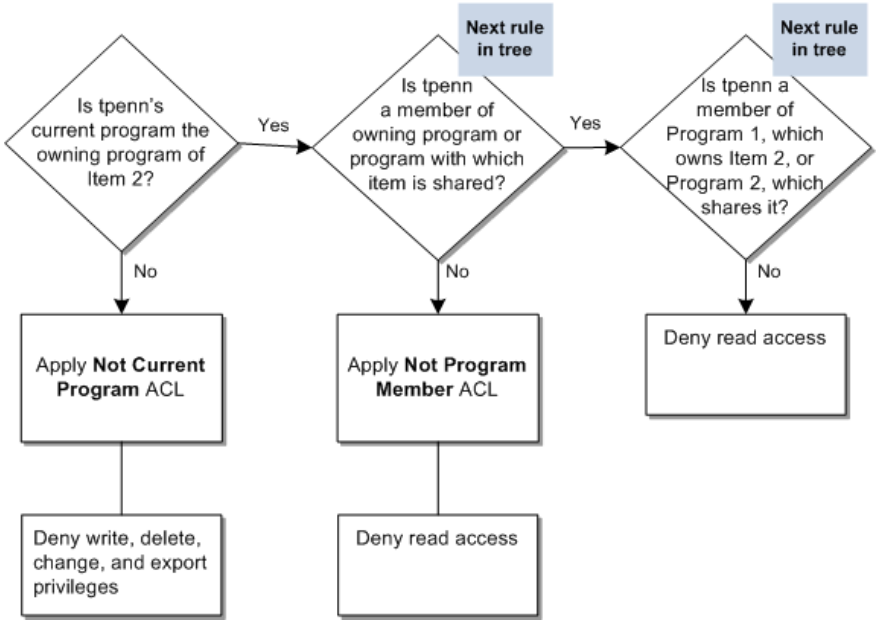
Example 5 – Deny read access

The following assumptions apply to this example:

- Item 2 is being accessed by tpenn, who:
  - Does not own Item 2.
  - Is a member of Program 3.
- tpenn's current program in the Teamcenter session is Program 3.
- Item 2 is owned by Program 1 and is shared with Program 2.

The **In Current Program(false) –> Not Current Program** rule is placed higher in the rule tree than the **Is Program Member(false) –> Not Program Member** rule; therefore, it is evaluated first, as shown in the flowchart. tpenn's current program is not the owning program of Item 2; therefore, tpenn is denied write, delete, change, and export privileges.

The **Is Program Member(false) –> Not Program Member** rule is evaluated next. tpenn is a member of the owning program or a program with which the item is shared; therefore, evaluate the next rule in the tree. tpenn is denied read privileges.



Implementation considerations for program-level security

You must consider the following points when implementing program-level security:

- Auto assignment of objects to projects.
- Program context.
- Placement of rules in the Access Manager tree.
- Interaction of program and project rules.
- Importing and exporting project data in a Multi-Site Collaboration environment.

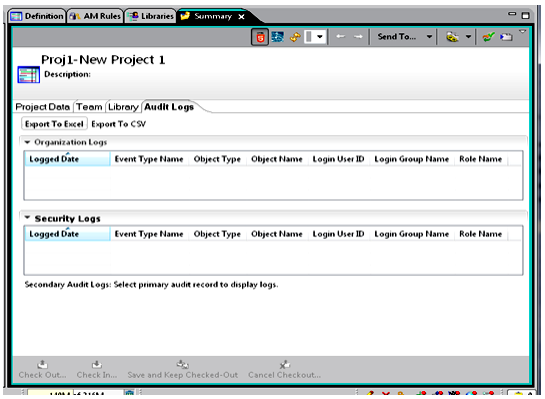
Using Security Logs to track events

You can use the **Security Logs** table in Audit Manager to see audit log events against a specific project. You can also generate reports using this data and export the audit events to Microsoft Excel and CSV (comma-separated values) format. For example, you can use the **Security Logs** table to track the following events:

- Assign/remove users to a project or program.
- Assign/remove groups of users to a project or program.
- Assign/remove all users in a role to a project role.

- Select team administrators.
- Set/unset project administrators.
- Set/unset privileged users.
- Set a default project.

As an administrator, you can check the audit log in the project's **Audit Logs Summary** tab to see the date, time, user, project name, action performed, and who performed the action. You can also run a saved query to search the audit log and view the information or generate a report that contains the information.



1. Enable audit logging using the Business Modeler IDE.
2. To define audit logs, create an audit definition using the Business Modeler IDE.
3. You can access audit information in several ways. For example, you can go to the **Summary** of the following Teamcenter applications, which shows audit logs in the **Audit Logs** tab.
   - My Teamcenter
   - ADA License
   - Structure Manager
   - Multi-Structure Manager
   - Manufacturing Process Planner
   - Schedule Manager
   - Workflow Viewer

- Organization

In addition to using the **Summary** view, you can:

- Run predefined audit reports or create new reports, using the Report Builder application.
- Create custom saved queries, using the Query Builder application.
- Run predefined audit queries, using the Teamcenter advanced search functionality.

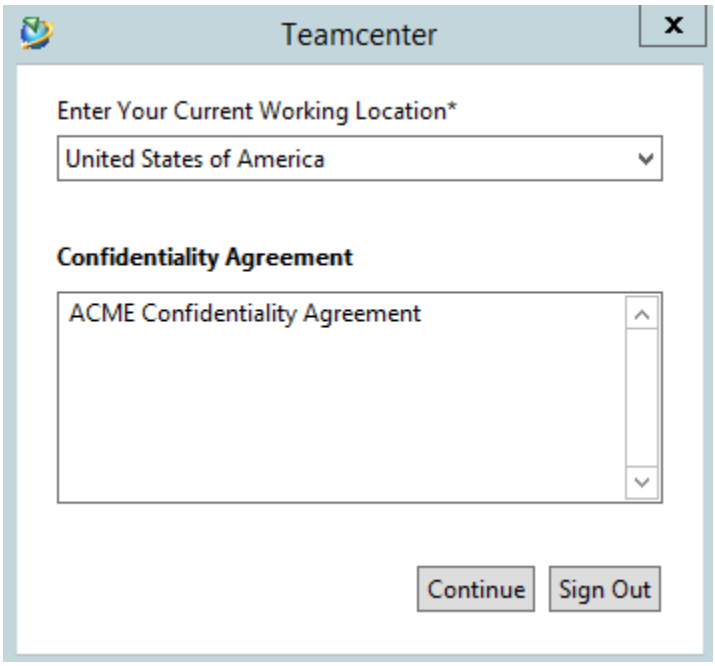
10. Configuring access based on geography

About geography access

Geography access allows you to configure both a geography entry and a custom confidentiality agreement prior to users logging on to a Teamcenter session.

For example, in the following Teamcenter session, users must first select the country in which they are currently located before the home page is displayed. If the user does not select a country, the only other option is to logoff.

If you require your users to agree to a confidentiality agreement, for example, for authorized data access (ADA) requirements, you can configure a custom confidentiality agreement statement to be displayed following the selection of their current working location. The **Continue** button is unavailable until a valid country is selected in the drop-down list.



You can run a report, **License Login Report**, that displays the logon information. This report is displayed in My Teamcenter by choosing **Tools→Reports→Report Builder Reports→License Login Report**.

Configure geography access

1. Update the site geography.

   You can assign geography to a site using the **site_util** utility.

> Note:
>
> You can also assign site geography using the Organization application.

2. Configure the geography list using the Business Modeler IDE.

   By default, Teamcenter attaches the **Fnd0CountryCodes** list of values (LOV) on the **User.Geography** attribute.

   > Note:
   >
   > If you add a custom LOV to the **User.Geography** attribute, you must remove it before starting an upgrade.

3. Update the user geography.

   You can assign geography to a single user using the **-Geography** argument of the **make_user** utility. To change the geography for all users at the same time, you can perform a batch mode change using the **-allUserDeclaredGeography** argument and the two-character ISO 3166 country code; for example, to set geography to Germany (DE) for all users, enter:

   ```
   -allUserDeclaredGeography=DE
   ```

   > Note:
   >
   > You can also assign user geography using the Organization application.

4. Configure preferences for logon entry of geography:

   - **LoginCountry_selection_enabled**

     Enables the **Country Selection** dialog box for users to select the country from which they are logging in.

True	Displays the **Country Selection** dialog box.
False	Allows the logon process to continue and display the user's home page.

   - **ShowUserDeclaredGeography**

     Controls the visibility of the **User Declared Geography** dialog box on the **User Settings** dialog **Login** panel.

True	Displays the **User Declared Geography** field on the **Login** panel of the **User Settings** dialog.
False	Hides the **User Declared Geography** field on the **Login** panel of the **User Settings** dialog.

- **LoginCountry_save_previous_selection**

  Allows/denies the ability to save the previous country selection in the **Country Selection** dialog box. If users are logging on from the same site each time, you can configure it so the user does not have to make the country selection each time.

  > Note:
  >
  > This preference is ignored when **LoginCountry_selection_enabled** is set to **False**.

True	Downloads the previously selected country and fills in the combination box in the **Country Selection** dialog box with the value stored on the **User.Geography** attribute. The user selects **Agree** to accept the previously entered country.
False	Causes the **Country Selection** dialog box to not save the previous geography entry. This forces all users who log on to enter a new country when logging on. The initial value in the selection box is blank and the **Agree** button is unavailable until the user selects a country.

5. After the geography access is enabled for users, you can generate the **License Login Report**. This report, which is also helpful for audit purposes, displays the following data:

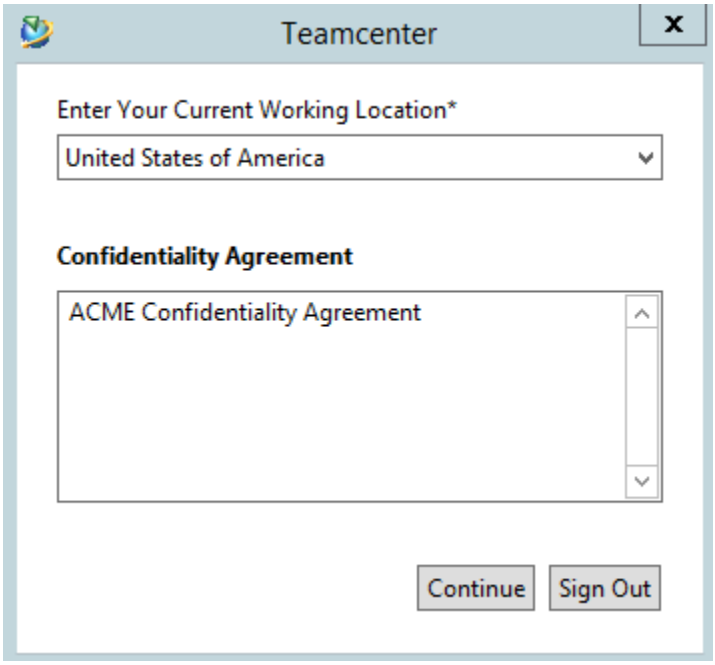
   - User ID
   - Month
   - Year
   - Geography
   - Intellectual property (IP)

   This report documenting logon information of users can be stored and used for future reference.

Configure a custom confidentiality agreement

> Note:
>
> By default, there is no confidentiality agreement configured.

You can configure a custom confidentiality agreement statement to be displayed following the user's selection of their current working location. The **Continue** box is unavailable until a valid country is selected in the dropdown list.



1. Create a custom textserver directory, for example, **customLocales**. Add the subdirectory structure for your language locales, for example, **en_US**.

   Tip:

   Create this directory in a location other than *TC_ROOT* or *TC_DATA* to prevent its loss during migrating or patching. A typical custom structure might be:

   ```
   C:\
   |---customLocales
      |---en_US
      |---de_DE
      |---fr_FR
      |---no_translation
   ```

2. Populate your custom textserver directory.

   a. Browse to your custom directory location, for example, **C:\ customLocales\en_US** and create a blank XML file using a naming convention, for example, *custom-name*_**text_locale.xml**. In this example, the file name is **ACME_Confidentiality_Agreement_text_locale.xml**.

   b. Browse to your custom directory location, for example, **C:\ customLocales\no_translation** and create a blank XML file using a naming convention, for example, *custom-name*_**text.xml**. In this example, the file name is **ACME_Confidentiality_Agreement_text.xml**.

Example:

With a confidentiality agreement for each locale, the directory structure now resembles the following:

```
C:\
|---customLocales
   |---en_US
   |      ACME_Confidentiality_Agreement_text_locale.xml
   |
   |---de_DE
   |      ACME_Confidentiality_Agreement_text_locale.xml
   |
   |---fr_FR
   |      ACME_Confidentiality_Agreement_text_locale.xml
   |
   |---no_translation
   |      ACME_Confidentiality_Agreement_text.xml
```

3. Modify your new *custom-name*_**text_locale.xml** file by configuring it with the following values for the custom textserver location.

```
<?xml version="1.0" encoding="Language_Encoding" standalone="yes"?>
<textsrv filename="custom-name_text_locale.xml">
<key id="LoginCountry_confidentiality_statement">Confidentiality
Agreement
     Statement</key>
</textsrv>
```

Note:

The double quotation marks (") need to remain. They do not encapsulate the Confidentiality Agreement Statement.

a. Set **Language_Encoding** to the language encoding, such as **us-ascii**.

   Retrieve this information by navigating to *TC_ROOT***lang\textserver***language-locale** and opening the **versionlocal_locale.xml** file. The encoding is listed on the first line of this XML file.

b. Set **textsrv filename** to the *custom-name*_**text_locale.xml**.

c. Set **Confidentiality Agreement Statement** to the confidentiality agreement you want to use.

   Your **ACME_Confidentiality_Agreement_text_locale** file should now resemble the following:

```
<?xml version="1.0" encoding="us-ascii" standalone="yes"?>
<textsrv filename="ACME_Confidentiality_Atreement_text_locale">
<key id="LoginCountry_confidentiality_statement">ACME
```

```
Confidentiality
     Agreement</key>
</textsrv>
```

4. Create a non-translatable resource file, *custom-name*_**text.xml** file (for example, **ACME_Confidentiality_ Agreement_text.xml**), in the **no_translation** directory.

   Edit the **ACME_Confidentiality_ Agreement_text.xml** file and add the following lines:

```
<textsrv filename="ACME_Confidentiality_Agreement_text.xml"
     xmlns:xi="http://www.w3.org/2001/XInclude">
<xi:include href="ACME_Confidentiality Agreement_text_locale.xml"
     parse="xml"/>
```

5. Modify the *TC_DATA***tc_profilevars.bat** file by adding the following environment variable, where **C:\customLocales** is the custom textserver directory you created.

```
set TC_USER_MSG_DIR=C:\customLocales
```

   Save and close this file.

6. Set the following site preferences:

   - **LoginCountry_selection_enabled=true**
   - **LoginCountry_save_previous_selection=true**

   Close the rich client, stop Teamcenter Services, clear the server cache, and restart Teamcenter services.

7. Verify the Confidentiality Agreement by logging on to the rich client.

Tip:

Following are tips for creating a confidentiality statement:

- Your confidentiality text file has different contents in each language directory. For example:
  - English (**en_US**) file:

```
<?xml version="1.0" encoding="us-ascii" standalone="yes"?>
<textsrv filename="ACME_Confidentiality_Agreement_text_locale.xml">
```

```
 <key id="LoginCountry_confidentiality_statement">ACME Confidentiality
     Agreement in English</key>
</textsrv>
```

- German (**de_DE**) file:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<textsrv filename="ACME_Confidentiality_Agreement_text_locale.xml">
 <key id="LoginCountry_confidentiality_statement">ACME Confidentiality
     Agreement in German</key>
</textsrv>
```

- French (**fr_FR**) file:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<textsrv filename="ACME_Confidentiality_Agreement_text_locale.xml">
 <key id="LoginCountry_confidentiality_statement">ACME Confidentiality
     Agreement in French</key>
</textsrv>
```

- Make certain your environment variable is set correctly:

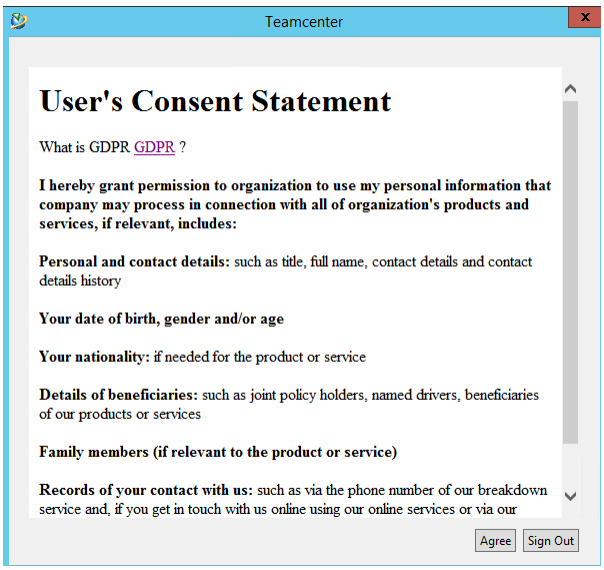
```
set TC_USER_MSG_DIR=C:\customLocales
```

11. Managing user consent

About GDPR

The General Data Protection Regulation (GDPR) is a set of rules designed to give citizens more control over their personal information. It mandates that business are compliant in how they handle personal information in connection with products and services.

Following is a sample consent statement.



Steps to create and configure a user consent GDPR statement

To create and configure a user consent GDPR statement to display as a prelogon page for both the rich client and Active Workspace, you must:

1. Create a GDPR directory and create your htm/html file with your GDPR consent statement. You will need a consent statement for each of your supported non-English locales.

2. Using the rich client, configure the consent statement dataset and attach the English locale consent statement by uploading it from your GDPR directory.

    a. Search for the **Consent Statement** workspace object using query.

    b. Check out the **ConsentStatement** dataset and select **Named References** to display the **Fnd0ConsentDataset** reference statement.

    c. Click **Upload**, browse to the location of the **consent_statement.hml** file, select a consent statement file that you created in Step 1 and click **Upload**.

d. Check in the **ConsentStatement** dataset.

e. Set the TCM Release status for the **ConsentStatement** revision and the **ConsentStatement** dataset by selecting the consent statement revision: **File** → **New** → **Workflow Process** and select **TCM Release Process** as the process template.

Repeat this step to create a separate dataset for each of your non-English locale statements and upload the appropriate statement for each non-English locale.

3. Set the GDPR-related preferences for your required scenario.

Of the six scenarios, select the appropriate scenario. For example, if you require your users to accept the consent notice every time they logon to the system, you must set all three GDPR-related preferences to **True**.

Three preferences control the display of the GDPR consent page:

- **TC_Org_AllowRevoke**
- **TC_Org_GDPRUserConsent**
- **TC_Org_ShowUserConsentOnEveryLogin**

> Note:
>
> To display the GDPR consent statement post-logon window to display in Active Workspace, you must set the **AWC_PostLoginStages** preference to **GDPRConsent**.

	Consent notice appears	Option to revoke consent appears
Scenario 1	During first logon/ every logon	Yes
Scenario 2	Never	No
Scenario 3	During first logon only	Yes
Scenario 4	During first logon/ every logon	No
Scenario 5	During first logon only	No
Scenario 6	Never	No

	Scenario					
	1	2	3	4	5	6
TC_Org_AllowRevoke	True	True	True	False	False	False

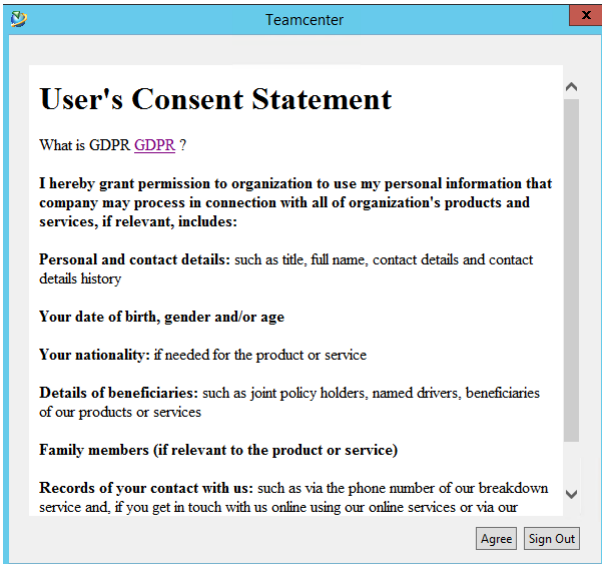
TC_Org_GDPRUserConsent	True	False	True	True	True	False
TC_Org_ShowUserConsentOnEveryLogin	True	True	False	True	False	False

The typical usage of the GDPR statement includes the following scenarios:

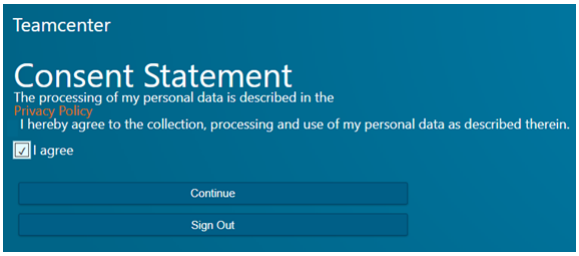
- If you configured your system to display the GDPR consent statement, the consent page, along with the check box to accept the consent, displays when the user attempts to log on with either the rich client or Active Workspace.
- The user is unable to logon until he or she accepts the consent.
- When you configure your system to display the GDPR consent statement for *consent during first logon*, the user is not presented the consent statement again unless a new revision is created and the **TCM Released** status is set for the new revision.
- If you configure the consent statement, the user can revoke the consent statement through the rich client. However, once revoked, the user is logged out from Teamcenter and will not able to log on again until the consent statement is accepted.
- Once the user accepts the consent statement (in either the rich client or Active Workspace), it applies to all clients.

Following are examples of the consent statement for the rich client and Active Workspace and how to revoke consent:

- Consent statement in the rich client:

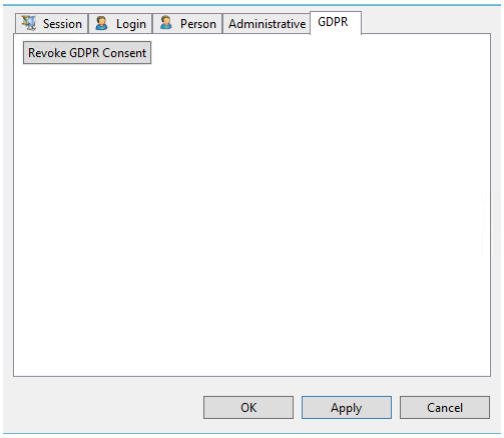


- Consent statement in the Active Workspace:

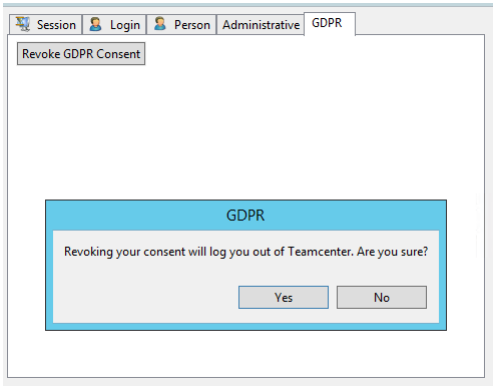


- Revoke consent statement in the rich client:

  a. To revoke consent from within My Teamcenter, select **Edit** → **User Setting** → **GDPR**.

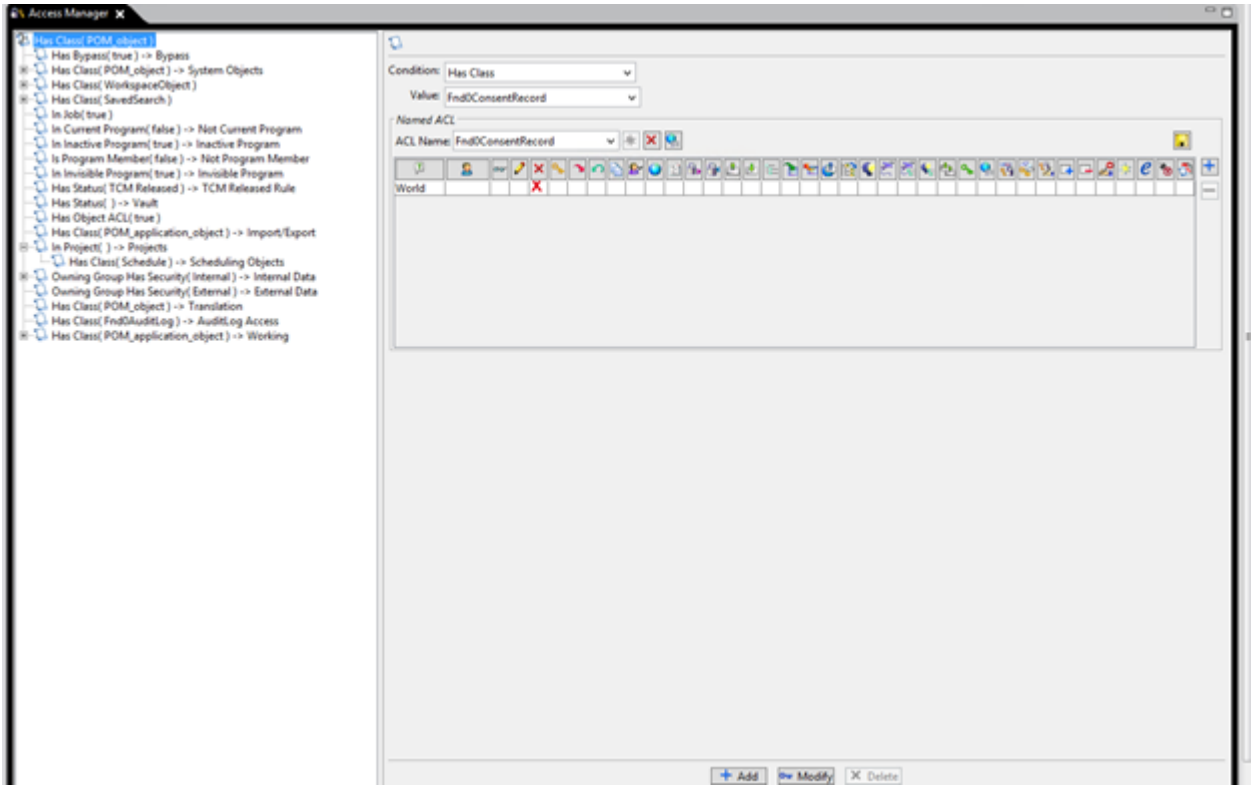


  b. From the **GDPR** tab, select **Revoke GDPR Consent**.



  c. Select **Yes** to log out of Teamcenter.

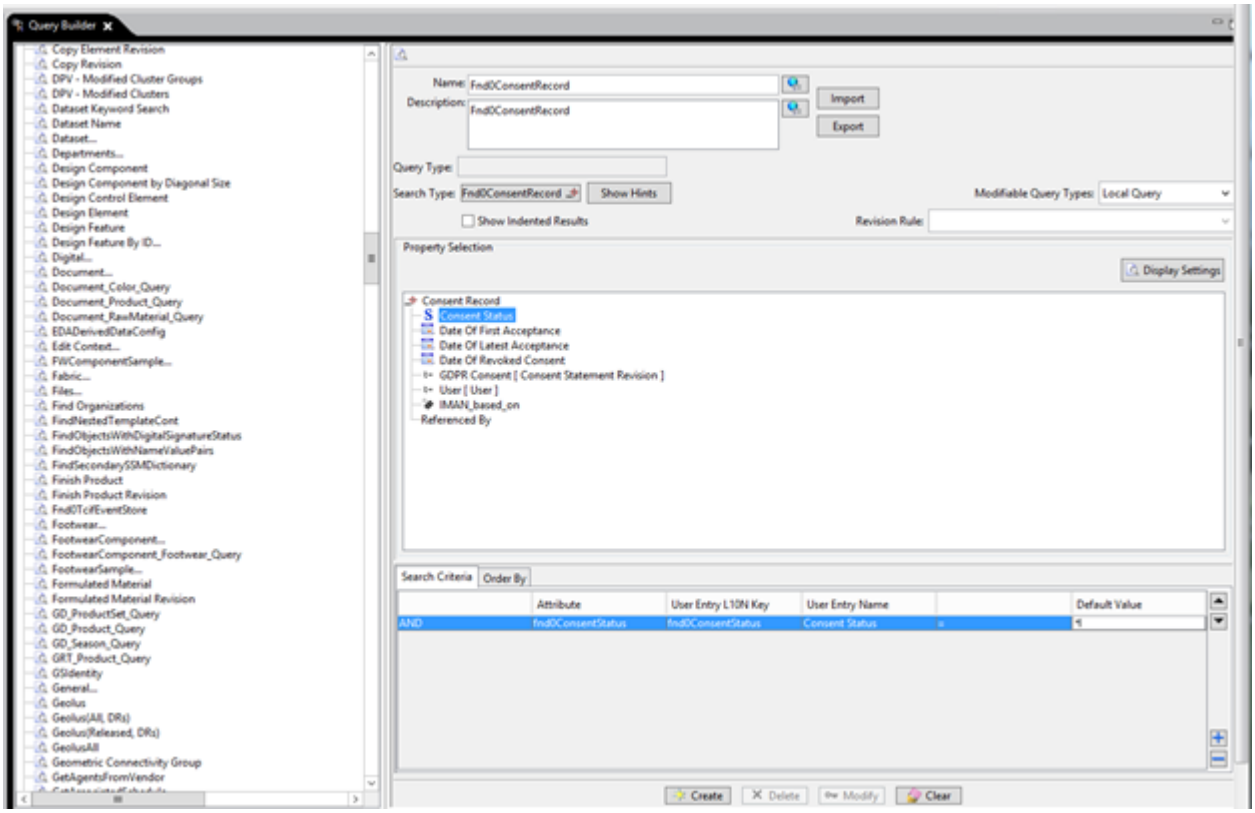
4. Using Access Manager, set the **Fnd0ConsentRecord** ACL to prevent a user from deleting consent record objects.



5. Using Query Builder, create a saved search, **Fnd0ConsentRecord**, to search for consent records.

   a. Using advanced search, search for **Fnd0ConsentRecord**.

   b. Create a saved query to search for the **Consent Status**.

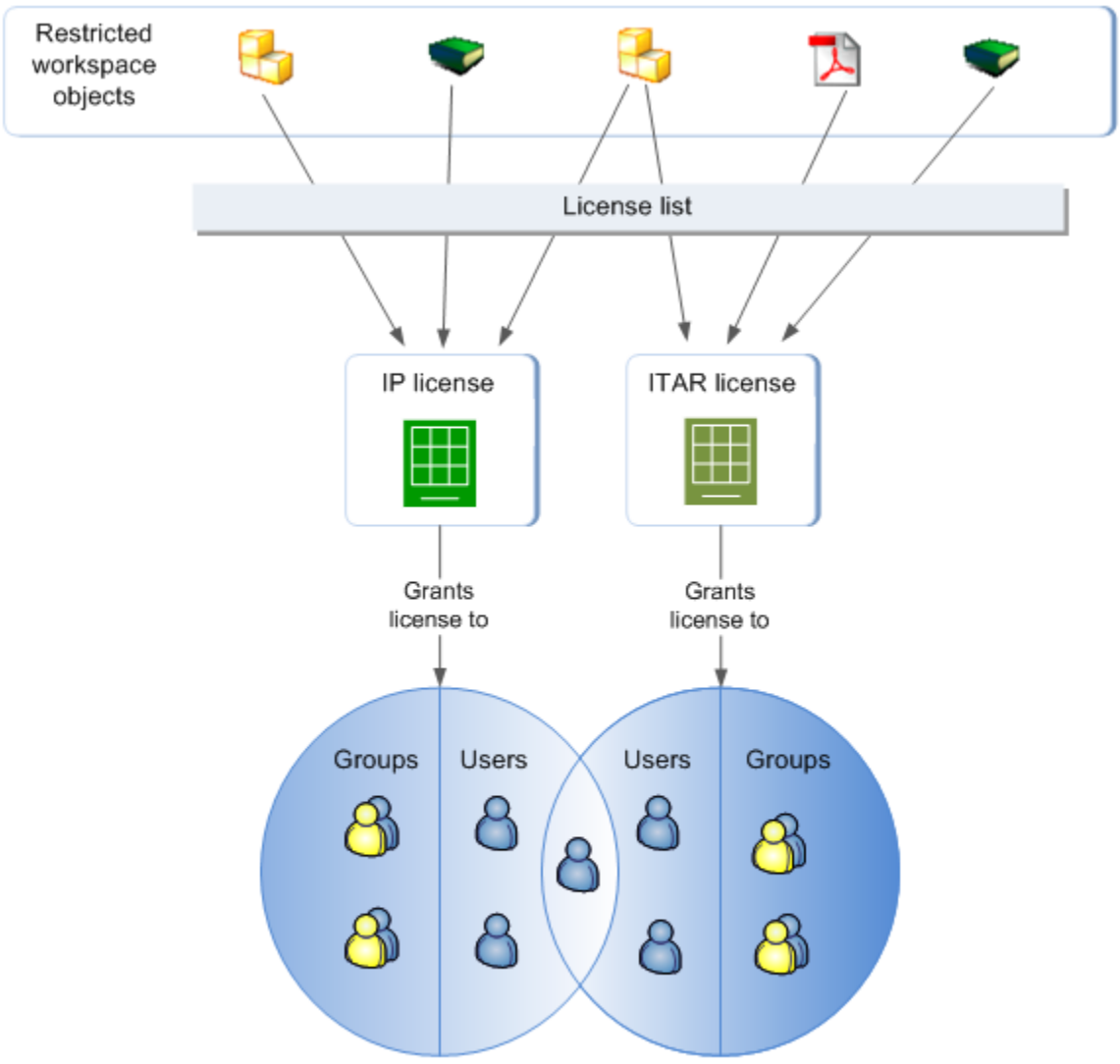


12. Configuring ADA License

Getting started with ADA License

What is ADA License?

ADA License provides support to enforce the International Traffic in Arms Regulations (ITAR) and intellectual property (IP) policies using authorized data access (ADA) licenses. ADA licenses provide discretionary (time-limited) grants or denials of access to users who do not have access to classified data based on their clearance level. ITAR licenses are the authorizing documents in Teamcenter that represent an effective Technical Assistance Agreement (TAA). The third license type, exclude license, is a mechanism for denying users access to data for a specific period of time.



ADA functionality provides the ability to:

- Classify data per ITAR or IP policies.
- Specify IP or government clearance levels for users.
- Attach or detach ADA licenses to and from workspace objects.
- Control access to classified data through Access Manager conditions and rules.
- Create and maintain audit records for actions performed on ITAR or IP licenses.

ADA licenses can be managed by non-database administrators. In addition, you can define users whose role is only to classify workspace objects.

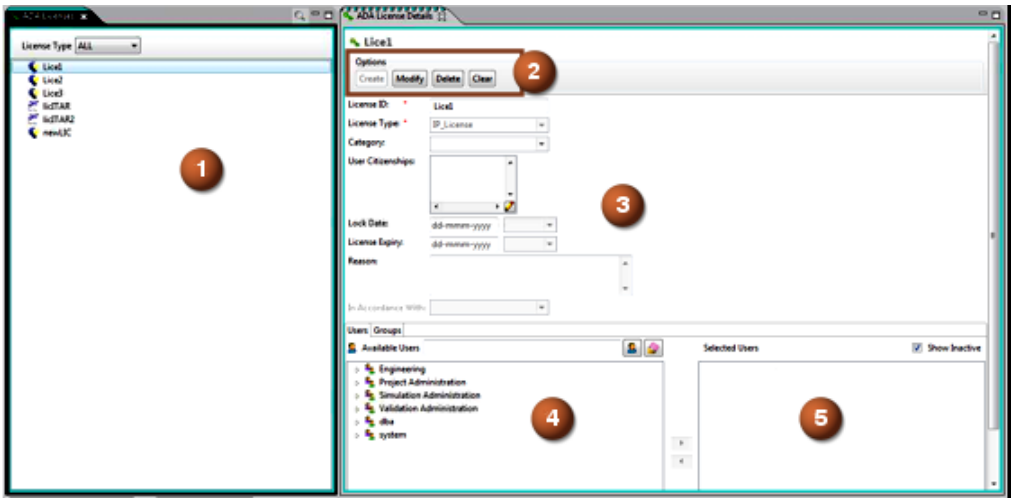
Before you begin

Prerequisites	To use ADA License to create, edit, or delete licenses, you must log on as a user with an ADA administrator role (**IP Admin** or **ITAR Admin**). Otherwise, you are limited to read-only access in the application. The required privilege is controlled by the **ADA_license_administration_privilege** preference. You must also configure the groups and roles to allow users write access to ADA License. Use Authorization for this configuration. Note: Access to ITAR license-controlled workspace objects is controlled on an object basis, and you may be unable to view specific objects if you do not have the proper ITAR privileges.
Enable ADA License	ADA License, a security application that complements other Teamcenter security features, is enabled by default when you install Teamcenter. To disable ADA License, set the **ADA_enabled** preference to **false**.
Configure ADA License	Configure who can manage licenses and other configurations to ADA License.
Start ADA License	Click **ADA License** in the navigation pane. Note: You can log on to Teamcenter only once. If you try to log on to more than one workstation at a time, you see an error message.

ADA License interface

Orientation to the ADA License interface

Some menu commands and toolbar buttons are specific to ADA License.



1	**ADA Licenses** tree	Displays the available authorized data access (ADA) licenses filtered by the **License Type** list selection.
2	License options	Provides buttons for the actions that you can perform on a license. If you are not authorized, these are disabled.
3	**ADA License Details**	Displays information about the licenses selected in the licenses tree and allows you to select and or enter license properties.
4	**Available Users** and **Available Groups**	Displays a list of available users in the system on one tab and available groups on another. Allows you to select the users or groups that can access objects with the license attached.
5	**Selected Users** and **Selected Groups**	Displays a list of users in the system on one tab and groups on another that currently have access to objects with the license.

ADA License menus

ADA License provides commands through shortcut menus that you access with the right mouse button after selecting an object. The shortcut menus are dynamic and provide the following commands when appropriate for the selected object.

Shortcut menu command	Purpose
License→Attach	Allows you to attach licenses to the selected workspace object.
License→Detach	Allows you to detach licenses from the selected workspace object.

Teamcenter also provides the rich client **View→Audit→View Audit Logs** command to allow you to view the ADA License audit logs, if authorized.

ADA License buttons

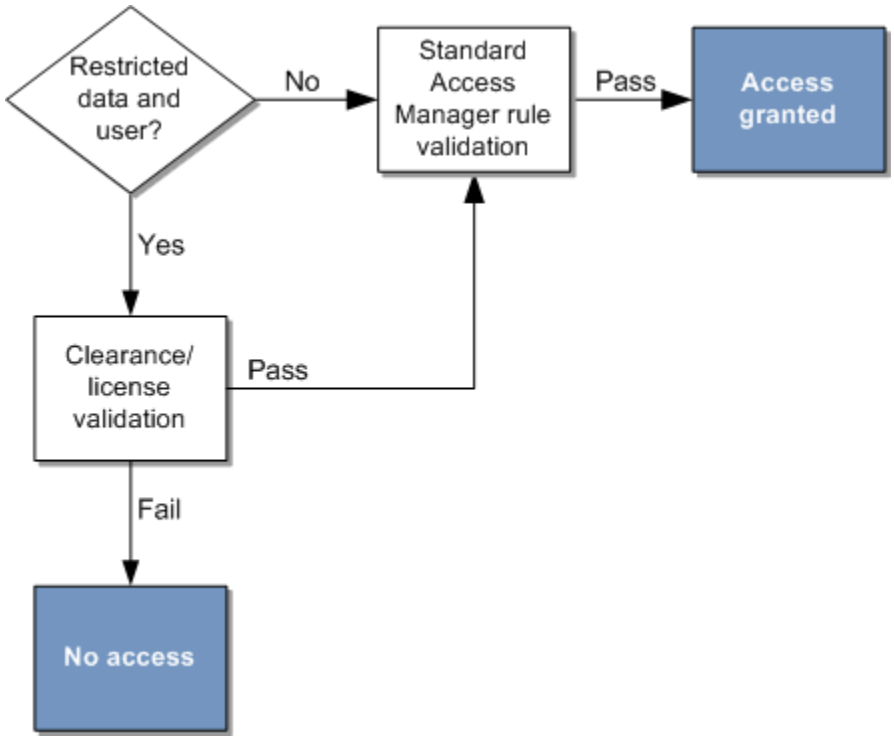
Button	Description
Create	Creates a license.
Clear	Clears the boxes in the **ADA License Details** pane.
Delete	Deletes the selected license.
Modify	Updates the ADA license with changes you have made in the **ADA License Details** pane.
View Audit Log	Displays the **audit log** that shows events logged for the selected license.

ADA License tabs

Tab	Description
Users	Displays available Teamcenter users you can select to allow access to objects with the license attached and shows users who have been previously selected.
Groups	Displays available Teamcenter groups you can select to allow access to objects with the license attached and shows groups that have been previously selected. Note: Use the **ADA_enable_subgroups** preference to control the display and selection of subgroups in the **Groups** pane. By default, this is set to **false**.

Authorized data access licensing workflow

ADA License controls sensitive data through the use of data classification, user clearance, and authorizing documents. When users attempt to access classified data in Teamcenter, their clearance level is evaluated against the classification of the object. Access is based on Access Manager rules, and if the clearance level is equal to or greater than the classification on the object, access is granted. Limited-time exceptions can be granted through an intellectual property license (**IP_License**) or International Traffic in Arms Regulations license (**ITAR_License**) and be denied through exclude licenses (**Exclude_License**).



Types of Authorized Data Access (ADA) licenses

ADA License has three types of licenses that provide limited access or exclusion:

- **IP_License**

  Grants discretionary access to specific users or groups to workspace objects that have intellectual property (IP) classification. It grants the access for a specified period of time.

- **ITAR_License**

  Grants discretionary access to specific users or groups to workspace objects with International Traffic in Arms Regulations (ITAR) classifications for a specified period of time. Typically it is used to grant access for a specific time period to citizens of other countries, United States (U.S.) citizens physically

located outside the U.S., or organizations that are named in an effective Technical Assistance Agreement (TAA) through an ITAR license.

ITAR licenses are used to control access to data that is deemed military in nature. For example, technical information may be subject to ITAR policies and, if so, must be protected so that only citizens of the U.S. have open access to the data. Viewing this classified data outside the U.S. is considered equivalent to performing an ADA export of it, which requires that rights are granted by license. Information marked as non-technical is, for the purposes of ITAR, available to all users.

The provisions for granting limited access to foreign nationals involve a Technical Assistance Agreement (TAA). These agreements are written for a specific entity or country, cover different types of information, and expire on different dates. Teamcenter-controlled information that is not technical is made available to all authorized U.S. citizens and users from foreign countries.

- **Exclude_License**

  Denies specific users or groups access to the attached workspace objects for a period of time.

Workspace objects that are under license control cannot be viewed by users who do not have specific user or group attribute values, such as:

- Nationality
- Geography
- Training
- IP or government clearance

Access based on these attributes is controlled through Access Manager (AM) rules, access control lists (ACLs), and accessors. Licenses can also be locked and you can set dates on which they will expire.

Basic tasks using ADA License

Perform the following basic tasks to manage ADA licenses and classify workspace objects:

Administering security

- **Define the ADA licensing security strategy**.
- **Define license categories**, if needed, using the Business Modeler IDE.
- Propagate ADA licenses using propagation rules in the Business Modeler IDE.

Managing ADA licenses

- **Create** and **delete** licenses to control access to objects.
- **Add In Accordance With (CFR) data** to a license.
- **Expire licenses**.
- **Lock and unlock licenses**.
- **Change the users or groups** allowed to access objects with attached licenses.

Managing ADA licenses on workspace objects

- **Attach** licenses to workspace objects.
- **Detach** licenses from workspace objects.
- **View licenses attached to workspace objects**.

Managing ADA license audit logs

- **Configure the Audit Manager application** that you want to use.
- **View and export the license audit data**.

About configuring ADA License

ADA License is a Teamcenter security application regarding authorized data access (ADA) that complements other Teamcenter security features, such as Access Manager rules and access control lists (ACLs). This application controls sensitive data through the use of data classification, user clearance, and authorizing documents. When users or groups attempt to access classified data in Teamcenter, their clearance level is evaluated against the classification of the object based on Access Manager rules. If the user or group clearance level is equal to or greater than the classification on the object, access is granted.

There are three types of licenses for authorized data access:

- IP license

  Grants discretionary access to data for a specific user for a specific period of time.

  You can configure the rule tree to check for a valid IP license associated with an object and user. If found, other access checks are bypassed.

- Exclude license

  A mechanism for denying users access to data for a specific period of time.

  You can configure the rule tree to check for a valid execution license associated with an object and user. If found, other access checks are bypassed.

- ITAR (International Traffic in Arms) license

  Grants discretionary access to specific users or groups to workspace objects with International Traffic in Arms Regulations (ITAR) classifications for a specified period of time. Typically it is used to grant access for a specific time period to citizens of other countries, United States (U.S.) citizens physically located outside the U.S., or organizations that are named in an effective Technical Assistance Agreement (TAA) through an ITAR license.

The ADA concepts described in the previous paragraphs assume that data is stored in and accessed from within the Teamcenter environment. You can also configure logging and menu suppression (blocking) to be applied when classified data is loaded in Teamcenter Integration for NX. Logging provides an audit of actions performed on exporting data, and blocking suppresses NX menus to prevent geometric data from being exported outside of the NX/Teamcenter environment.

> Note:
>
> In this context, *export* refers to performing an operation, such as creating a copy or printing data, that moves the data outside of the Teamcenter environment.

As a user with an ADA administrator role (**IP Admin** or **ITAR Admin**), you use the ADA License application to create and maintain licenses. Once created, access is either granted or denied to users and groups by associating the license directly with the data object.

Configuring ADA access

Overview of configuring ADA access

You can configure the following:

- Access to **authorized data** (workspace objects).
- Access to the **ADA License** application.
- Access to **view and apply licenses**.
- **Subtypes or categories of ADA licenses**.
- Access to **licenses based on citizenship**.

- Control of **security propagation across multiple sites**.
- Access to the **ADA License audit log**.

Controlling access to authorized data

Access Manager rules are used to control access to workspace objects with attached International Traffic in Arms Regulations (ITAR), intellectual property (IP), and exclude licenses. You control access by:

- **Configuring a user's access to authorized data** using authorized data access (ADA) and ITAR attributes, which allow or restrict access to data based on clearance levels and data classification.
- (Optional) Setting the **ADA_enable_subgroups** preference (**false** by default) to control access to subgroups.

Controlling access to the ADA License application

Access to the ADA License application depends on your group or role and the privileges set.

Teamcenter provides roles to perform specific functions in managing ADA functionality. These roles are used to assign users as part of groups, and also as accessors in writing Access Manager rules to grant or revoke access to classified workspace objects and management of ADA licenses. The table below provides details on the typical privileges associated with these roles.

Comparison of roles

Role	Description
ITAR Admin	Specifies government classification on workspace objects, manages ITAR licenses, and attaches or deletes ITAR licenses to and from workspace objects.
IP Admin	Specifies IP classification on workspace objects, manages IP or exclude licenses, and attaches or deletes IP or exclude licenses to or from workspace objects.
ADA Site Admin	Manages ITAR, IP, and exclude licenses.
ITAR Classifier	Specifies government classification on workspace objects.
IP Classifier	Specifies IP classification on workspace objects.

You control access to activities performed on licenses in ADA License by:

- Configuring the group and role combination for users who can access the ADA License application through the Authorization application.
- Setting the privilege for access in the **ADA_license_administration_privilege** preference (**ITAR_ADMIN** privilege by default).

- Assigning the appropriate privilege to users of ADA License functions, such as create, modify, or delete, as defined by the **ADA_license_administration_privilege** preference using the Organization application.

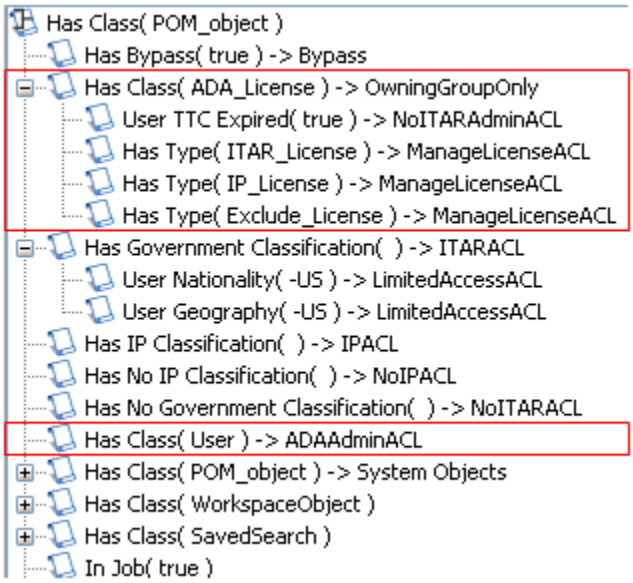
  Note:

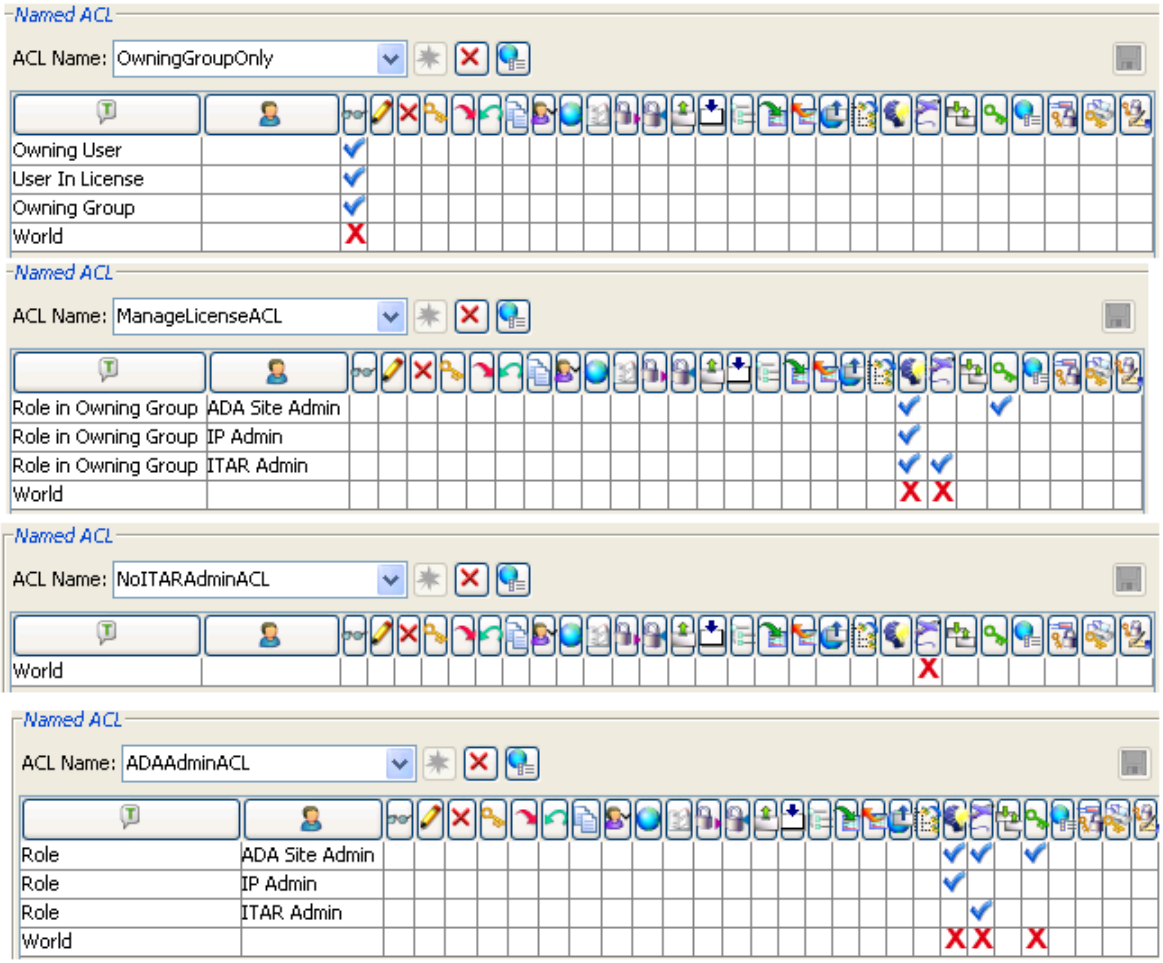
  If you have access to ADA License through the privileges **ITAR_ADMIN** and **IP_ADMIN**, you can also set International Traffic in Arms Regulations (ITAR) and government classifications.

To use the audit logging feature of ADA License, ensure the Audit Manager application is enabled (**TC_audit_manager** preference is **ON** by default).

Recommended Access Manager rules

The following are examples of recommended rules and access control lists (ACLs) for users managing ADA licenses.





Controlling access to view and apply ADA licenses

Two accessors restrict or grant access to license objects:

- **User In License**

  Evaluates a license object to verify that the user currently logged on is listed on the license. If the user is listed on the license either through the user or group value, **User In License** evaluates to **true**. If the user is not listed on the license the evaluation returns **false**.

  User In License is the opposite of **User Not In License**.

- **User Not In License**

  Evaluates a license object to verify that the user currently logged on is not listed on the license. If the user is not listed on the license either through the user or group value, this accessor type evaluate to **true**. If this user is listed on the license the evaluation returns **false**.

  User Not In License is the opposite of **User In License**.

You can use these two accessor types to control the licenses that are visible to users in Teamcenter applications, such as when searching for licenses, viewing licenses in the ADA License application, attaching licenses to an object, or viewing licenses attached to an object.

By default, the Teamcenter Access Manager rule tree does not include either of these two accessor types. Therefore, every user can view (read) the licenses that are attached to an object. This could be undesirable if you do not want users to view licenses that they are not listed on. If you want to control who can view the licenses or other similar use cases, use these accessors to obtain the desired behavior.

The following examples show rule tree configuration for each type of license in Teamcenter. The configurations allow users listed on the license to view the specified type of licenses in Teamcenter, whereas all other users are not able to see them:

Rule (Condition)	Named ACL	Accessor Type	READ
Has Type (ADA_License)	ACL1	User in License	Grant
		World	Deny

Rule (Condition)	Named ACL	Accessor Type	READ
Has Type (IP_Exclude)	ACL1	User in License	Grant
		World	Deny

Rule (Condition)	Named ACL	Accessor Type	READ
Has Type (IP_License)	ACL1	User in License	Grant
		World	Deny

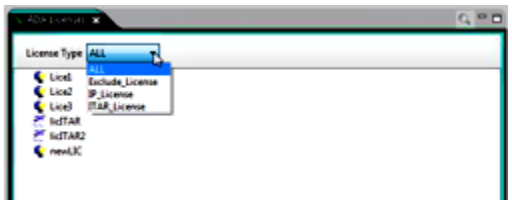
Rule (Condition)	Named ACL	Accessor Type	READ
Has Type (ITAR_License)	ACL1	User in License	Grant
		World	Deny

If this Access Manager rule tree configuration is used, the user cannot view the license object in the following areas:

- Search results (when searching for license objects using Teamcenter search).
- View of licenses that can be attached to a workspace object (when Teamcenter queries for the licenses that can be attached to an object through the **Attach License** dialog box).

- View of the license already attached to an object (for example, in the **View**, **Edit Properties**, **Summary**, **Details** tabs).

- ADA License application. The **ADA Licenses** view shown assumes the user has access to ADA License and the user can only view licenses that he or she is granted viewing access to through the rules in the rule tree.



For a license to appear in the **Attach License** dialog box, Teamcenter validates the following in addition to checking read access:

- The license is not locked or expired.

- The user has the **ITAR_ADMIN** privilege to the license if it is an ITAR license.

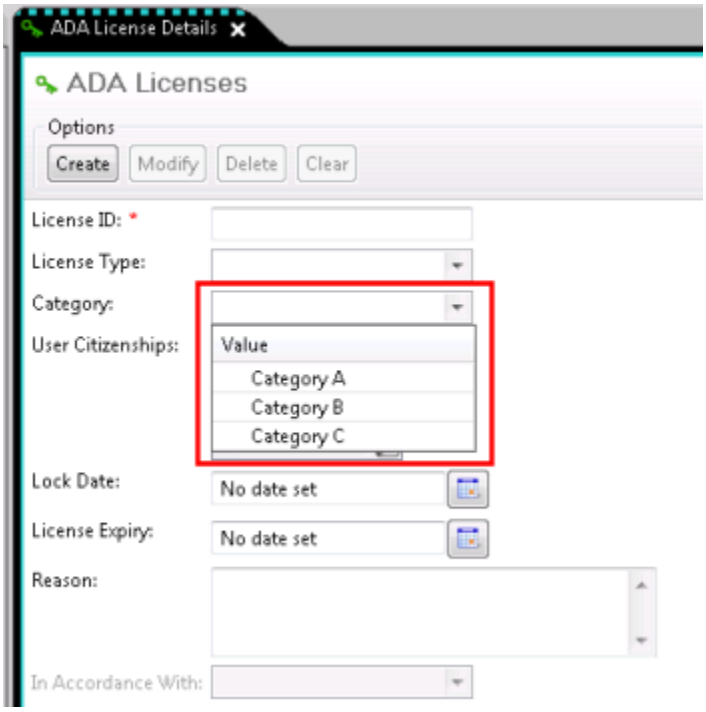
- The user has the **IP_ADMIN** privilege to the license if it is an IP or exclude license.

Defining license categories

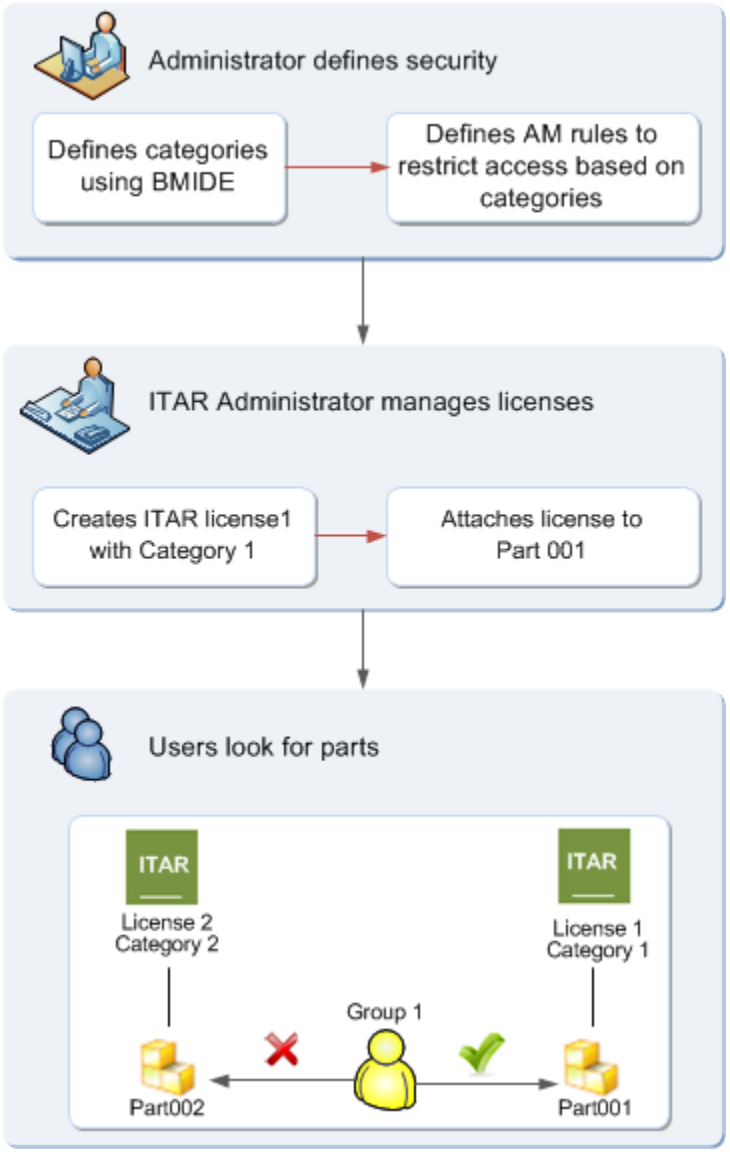
About defining license categories

ADA License provides three types of licenses (ITAR, IP, and exclude), but administrators can further refine these license types into subtypes or categories. Administrators can then write rules that restrict a user from objects based on the category of the license. For example, an administrator can create categories for different suppliers, regional groups, or departments within a company. The administrator can then restrict a group of users from accessing parts with the ITAR license of one type of category while allowing them access to an ITAR license with another category.

By default, ADA License provides three categories of licenses. They are generically named Category A through C, as shown in the ADA License Details page in ADA License. Administrators can use the Business Modeler IDE to change the default license category names to something more meaningful, use them as they are, or add more. Users of ADA License can then select license categories when creating and modifying licenses.



The workflow for defining and using categories is shown in the following figure:



See the following to learn more:

- Defining categories using the Business Modeler IDE.
- **Defining Access Manager rules to restrict access based on license categories, including an example**.
- **Creating licenses with categories**.

About defining Access Manager rules to restrict access based on license categories

Access control by licenses can be configured based on the license type to vary access at a high level or based on the license name to vary the access at a granular level. Categories offer a way to control access

by licenses in between the high and granular levels. They provide a way to have different subtypes of licenses under each type and configure access based on each category.

You can use Access Manager rule tree conditions to help you restrict access to workspace objects based on license categories. The following restrict access based on category name:

- Has ADA License Of Category
- Has Exclude License Of Category
- Has IP License Of Category
- Has ITAR License Of Category

The next four let you restrict access to workspace objects based on whether the user in the current session is listed on the license with the specified category:

- User In Attach ADA Lic of Ctgry
- User In Attach Excl Lic of Ctgry
- User In Attach IP Lic of Ctgry
- User In Attach ITAR Lic of Ctgry

Example of defining Access Manager rules to restrict access based on license categories

The following example demonstrates how ABC Company configures access privileges that vary for a single item object based on the license category.

License category configuration in Business Modeler IDE

The administrator of ABC Company uses the Business Modeler IDE to set up the desired categories for the licenses:

- Reviews the categories defined in the **Fnd0ADALicenseCategories** list of values (LOV).
- Adds more categories, if needed.

  In this case, administrator of ABC Company adds three categories for the ITAR licenses (**Category1**, **Category2**, and **Category3**).

License application configuration in ADA License

The ITAR administrator creates the following licenses setting their categories as follows:

- **license1** with a category of **Category1**.
- **license2** with a category of **Category2**.

The ITAR administrator attaches **license1** to **Part001** and **license2** to **Part002**.

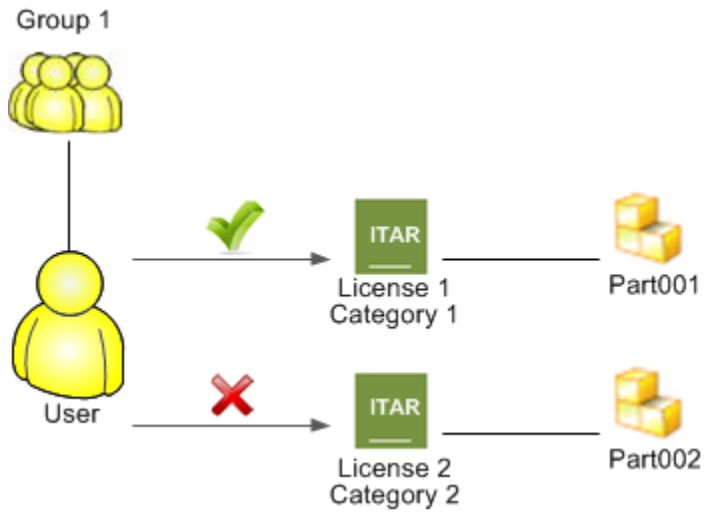
Rule configuration in Access Manager

The administrator then creates the following rules so only members of **Group1** can view objects that have an attached ITAR licenses with a **Category1** category, and only members of **Group2** can view objects with attached ITAR licenses with a **Category2**.

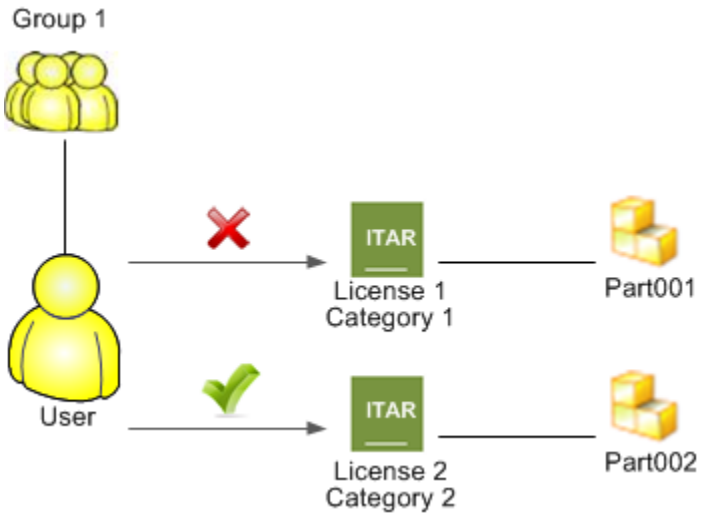
- User Is ITAR Licensed()
- Has ITAR License Of Category (“Category1”)
  - Group1 –> Read
- User Is ITAR Licensed()
- Has ITAR License Of Category (“Category2”)
  - Group2 –> Read

Rule tree evaluation

A user in **Group1** is allowed to view **Part001** because it has a license with a **Category1** attached, but the same user is not permitted to view **Part002** because it has a license with a **Category2** attached.



In a similar way, a user in **Group2** is not allowed to view **Part001** because it has a license with a **Category1** attached, but the same user in **Group2** is allowed to view **Part002** because it has a license with **category2** attached.



Example of defining Access Manager rules to restrict access based on user and license category

The following example demonstrates how ABC Company configures access privileges that vary for a single item object based on the license category and the user in the current session. ABC Company configures one set of privileges for users who are listed on a Category A IP license and configures another set of privileges for users who are listed on a Category B IP license.

License category configuration in Business Modeler IDE

The administrator of ABC Company uses the Business Modeler IDE to set up the desired categories for the licenses:

- Reviews the categories defined in the **Fnd0ADALicenseCategories** list of values (LOV).
- Adds more categories, if needed.

  In this case, ABC Company uses the two categories for the IP licenses (**Category A** and **Category B**) that are already defined in the LOV.

License application configuration in ADA License

The ITAR administrator uses to ADA License to create two IP licenses setting their categories as follows:

License type	Name	Category	Users
IP License	**IP001**	**Category A**	User1
IP License	**IP002**	**Category B**	User2

The ITAR administrator then attaches both of the licenses to a single workspace business object. For example, attaches them to the **item001**.

Rule configuration in Access Manager

The administrator then uses the Access Manager to create the following rules to check for:

- If the user is *IP Licensed*, meaning that the user currently logged on is specified on any of the IP licenses regardless of whether the license is connected to any workspace object.
- The IP license category (two rules check for this).

User Is IP Licensed()

User In Attach IP Lic of Ctgry (“Category A”)->ACL_A

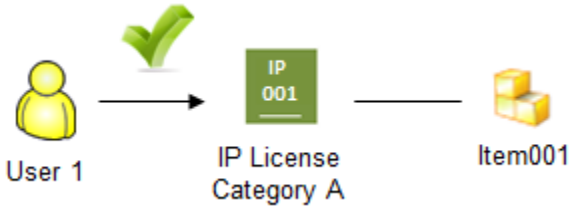
User In Attach IP Lic Of Ctgry (“Category B”) -> ACL_B

Example rule tree evaluation

The following explains how the rules are evaluated.

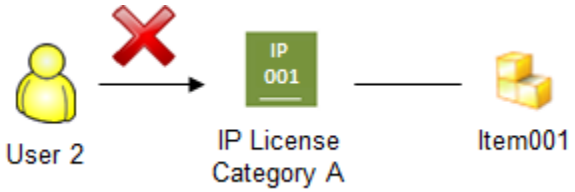
- **user1** logs on and tries to access **Item001**:
  - **User Is IP Licensed** evaluates to **true** because **user1** is listed on an IP license.

  - **User In Attach IP Lic Of Ctgry (“Category A”)** evaluates to **true** because **user1** is on an IP license, with **category=Category A**, and the license is attached to the **Item001**.

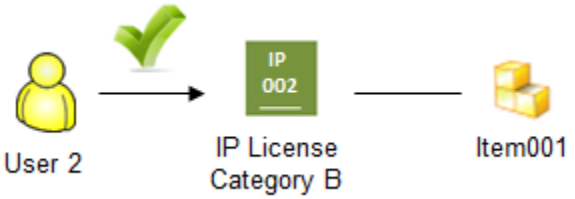


- **user2** logs on and tries to access **Item001**.
  - **User Is IP Licensed** evaluates to **true** because **user2** is listed on an IP license.

  - **User In Attach IP Lic Of Ctgry ("Category A")** evaluates to **false** because **user2** is not listed on an IP license with **category=Category A**.



  - **User In Attach IP Lic Of Ctgry ("Category B")** evaluates to **true** because **user2** is listed on an IP license with **category=Category B**, and the license is attached to the **Item001**.



Setting user citizenship

You can apply one or more citizenships to a license when creating and modifying the license in the ADA License application and when using the **ada_util** utility. This lets you support access control based on user citizenships.

For example, you can create a license with Great Britain and the United States as the allowed citizenships. Then, a user accessing the object to which the license has been applied can have either a United States or Great Britain citizenship or both.

Citizenships are two-letter country codes from ISO 3166, for example, **US** (United States) and **GB** (Great Britain).

Controlling license propagation

Controlling security propagation

ADA licenses can be propagated to a new object when the object is created using the **Save As** command. License propagation is determined by relation propagation rules and the values specified for the **ADA_saveas_propagated_license_types** preference.

The license propagation leverages the project propagation mechanism.

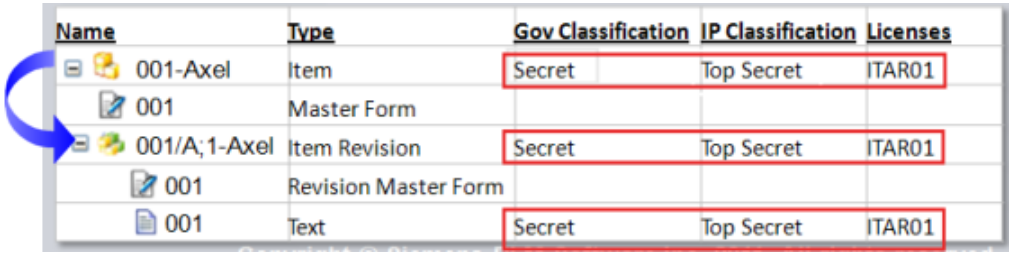
- A license attached to an item is propagated to the item revisions and associated datasets attached through the listed relations.
- Licenses attached to an item revision are propagated to associated datasets.
- Save As actions propagate the licenses and classification data from the source workspace object to the target.

  Save As actions include the commands **Save As**, **Check Out**, **Revise**, and **Baseline**.

Propagation does not cascade through product structures.

Example:

When a user sets the government and IP classifications and attaches an ITAR license to **001–Axel**, the attributes are propagated from **001–Axel** to its related item revisions and datasets:



You can use access rules to grant or deny access based on any of these attributes.

- If object Has Government Classification = Secret

    - Allow Group1 to read
- If object Has IP Classification = Top-Secret
    - Allow Group2 to read
- If User In License
    - Allow user to read

You can also **control license propagation across multiple sites**.

Controlling the propagation of security data using propagation rules

A propagation rule copies property values from one object type to another object type using relationships and property references when a certain operation occurs (for example, at checkin).

Using propagation rules, you can control the propagation of security data such as project assignments, ADA licenses, and Classification properties (for example, **ip_classification**, **gov_classification**, and **itar_classification**).

> Note:
>
> Propagation rules now replace the older preferences-configured propagation method.

Use the **Propagation Rules** editor in the Business Modeler IDE to create propagation rules.

Controlling license propagation across multiple sites

In a configuration that includes multiple sites, use the **ADA_override_on_import** preference to specify whether or not a remote site mirrors the license-related properties of the master site for a workspace object. For example, whether the Authorized Data Access (ADA) license attached to or detached from an item revision is propagated to the remote site.

Configure audit log access

To view or query for audit log information, the following must be set:

- Turn on audit log access.

  Set the **TC_audit_manager** preference to **ON**.

- Set the Audit Manager application by setting the **TC_audit_manager_version** preference to **3**.

Good ADA license practices

- **Placement of ADA rules in the rule tree**

  Rules that identify users with International Traffic in Arms Regulations (ITAR) or intellectual property (IP) clearance or licenses should be placed high in the rule tree. This ensures that they are the first rule enforced, for example, before any workflow rules.

- **Specify authorized data access rule precedence**

  When both ITAR and IP access rules are in force at a site, ITAR rules should be configured to take precedence over IP rules.

- **Specify classification and licenses at the correct object level**

  Specify the classification value or apply licenses to the correct workspace object to prevent unintended propagation and the associated access restrictions.

- **Associate users to correct groups/subgroups**

  Users should be made members of the most specific group possible to ensure the accuracy of key attributes, such as the nationality of the organization.

For more best practices for managing authorized data access, see *About configuring ADA License*.

Configuring ADA License for IP

About configuring ADA License for IP

Limited-time exceptions to allow access to data that is under authorized data access (ADA) control can be granted to specific users through an authorizing document (intellectual property (IP) license). Conversely, access to classified data can be denied to specific users for a period of time through the use of exclude licenses.

> Note:
>
> Securing data through an IP license only restricts it so it cannot be seen by users. If the data is visible, nothing prevents a user from printing or saving a copy.

Applying Access Manager concepts to IP classified data

About applying Access Manager concepts to IP classified data

The following concepts are fundamental to understanding how to apply Access Manager concepts to intellectual property (IP) data.

Concept	Description
IP clearance	IP clearance applies to a specific user and specifies the level of access the user has to sensitive (classified) information. The clearance value assigned to the user must match a value defined in the **IP_level_list_ordering** preference.
IP classification	IP classification applies to data and specifies the clearance level required for users to access the data. The classification value assigned to an object must match a value defined in the **IP_level_list_ordering** preference.
IP licenses	IP licenses grant discretionary access to data for a specific user or group for a specific period of time. The rule tree can be configured to check for a valid IP license associated with an object and user. If found, other access checks are bypassed.
Exclude licenses	Exclude licenses are a mechanism for denying users access to data for a specific period of time. The rule tree can be configured to check for a valid exclude license associated with an object and user. If found, other access checks are bypassed.
IP Admin privileges	**IP Admin** privileges must be explicitly granted and denied to prevent users from gaining unauthorized access to IP data.
IP Classifier privileges	**IP Classifier** privilege lets users classify data without having all the privileges of the **IP Admin**. It must be explicitly granted and denied to prevent users from gaining unauthorized access to IP data.

Access Manager rules for restricting access to IP

Access Manager rules allow you to establish access controls on intellectual property (IP) data. The following rule conditions and accessor types are used to configure IP data access rules.

Condition	Value	Description
User Has IP Clearance	Specific clearance values that can be prefixed by the following operators: **“>”** **“>=”** **“<”** **“<=”** **=**	Checks whether the IP clearance level of the user being evaluated is equal to, greater than, or less than the value specified in the condition. For example, **>=secret**. The operators can be used without a clearance value; the user's clearance is compared to the IP classification attribute of the object based on the specified operator. For example, **>=**. Note: If the data is not IP classified, the **User Has IP Clearance** condition is evaluated as being true,

Condition	Value	Description
		regardless of whether or not the user is assigned a clearance level.
Has IP Classification	Specific IP classification attribute values that can be prefixed by the following operators: “>” “>=” “<” “<=” =	Checks whether the IP classification of the workspace object being evaluated is equal to, greater than, or less than the value specified in the condition. The operators can be used without a clearance value; the IP classification attribute of the object is compared to the user's clearance level based on the specified operator. Note: If the object has no IP classification attribute value, this rule does not apply.
Has No IP Classification		Checks whether the workspace object does not have a value specified in the IP classification attribute.
User Is IP Licensed	**true** or **false**	Checks whether the user being evaluated is listed on an IP license attached to the workspace object. If set to **true** verifies the existence of a valid (not expired) IP license attached to the object being evaluated that names the current user or their group as a licensee.
User Is Excluded	**true** or **false**	Checks whether the user being evaluated is listed on an exclude license attached to the workspace object.
User In Attached IP License	**Any** or **All**	Checks whether the user being evaluated is listed on any or all IP licenses attached to the workspace object being evaluated. • If set to **Any**, the user is on at least one IP license. • If set to **All**, the user is on all IP licenses.
Has Named IP License	*License ID*	Checks whether a specific IP license is attached to the workspace object being evaluated.
User In Named IP License	*License ID*	Checks whether the user being evaluated is listed on an IP license of the specified name. It does not check if the license is attached to the workspace object being evaluated.
User In Attached Exclude License	**Any** or **All**	Checks whether the user being evaluated is listed on any or all of the exclude licenses attached to the workspace objects.

Condition	Value	Description
		• If set to **Any**, the user is on at least one exclude license. • If set to **All**, the user is on all exclude licenses.
Has Named Exclude License	*License ID*	Checks whether a specific exclude license is attached to the workspace object being evaluated. It does not check if a user is on the license.
User In Named Exclude License	*License ID*	Checks whether the user being evaluated is listed on an exclude license of the specified name. It does not check if the license is attached to the workspace object being evaluated.

Accessor	Description
User Excluded	User or group that is listed in a valid exclude license attached to the workspace object being evaluated.
User is IP Licensed	User is cited in a current IP license associated with the selected object either directly or by membership in a cited organization (group).
User IP Unlicensed	User is not cited in any current IP license associated with the selected workspace object.
User Under IP Clearance	User's IP clearance is below the level required by the object. This accessor is typically used to revoke access and is only applicable when the IP clearance on the user and the IP classification on the object come from a common multilevel scheme defined by the **IP_level_list_ordering** preference.
User Has IP Clearance	Checks whether the IP clearance level of the user being evaluated is equal to, greater than, or less than the value specified in the condition.
User Over IP Clearance	User's IP clearance level is over the level required by the object. This accessor is typically used to grant access and is only applicable when the IP clearance level on the user and the IP classification on the object come from a common multilevel scheme defined by the **IP_level_list_ordering** preference.

Approaches to classification and licensing

Classification and licensing can be implemented at different levels within the **object model hierarchy**. This section describes three approaches to licensing and classification for controlling access to the datasets that contain geometry data.

Classifying, licensing, and performing the rule check at the item level

Applying classification and licensing to the item object and performing the rule check at the item level allows you to control access to the item and all related information beneath that item in the object hierarchy, because licensing and classification attributes are inherited from parent items in the hierarchy.

While this method achieves the goal of protecting the geometry data, it prevents unauthorized users from viewing the metadata, such as items, item revisions, and BOM view revisions.

Classifying and licensing at the item level but performing the rule check at the dataset level

Applying classification and licensing at the item level but performing the rule check at the dataset level is a more flexible alternative. For example, assume that licensing and classification are applied at the item level and the Access Manager rule is written to check the classification and licensing when the object being access is a **UGMASTER** dataset.

This approach achieves the goal of preventing unauthorized access to the geometry data while allowing you to apply classification and licensing at a high level in the hierarchy, which is more efficient than applying classification and licensing to each dataset individually. However, this approach also allows users without clearance to view the metadata associated with the geometry.

The following example shows how the rule tree might be configured to support classification and licensing at the item level with rule checking performed at the dataset level:

- Has Class(User) –> IPAdminACL
- Has Type(UGMaster)
  - Is Ex Licensed(True) –> NoAccessACL
  - User Is IP Licensed(True) –> Consumer ACL
    - Has Class(POM_object) –> ProjRoleACL
  - User Is IP Licensed(False) –> NoAccessACL
    - User Has IP Clearance(>=) –> ProjRoleACL
- Has Type(UGPart)
  - Is Ex Licensed(True) –> NoAccessACL
  - User Is IP Licensed(True) –> Consumer ACL
    - Has Class(POM_object) –> ProjRoleACL
  - User Is IP Licensed(False) –> NoAccessACL
    - User Has IP Clearance(>=) –> ProjRoleACL

Classifying, licensing, and performing rule checks at the dataset level

Classifying and licensing at the dataset level does not allow inheritance of classification and licensing properties from parent items and item revisions; therefore, these attributes must be applied to each individual dataset. While this is a valid approach to classification and licensing, Siemens Digital Industries Software does not recommend this approach.

Scenarios — Restricting user access using rules and ADA licenses

To learn about restricting user access to data using rules and authorized data access (ADA) licenses together, see the use cases in *Scenario – Using rules to control access to data through ADA licenses*. The use cases use ITAR examples but there are corresponding rule conditions for managing access to data using IP licenses, so the concepts shown for ITAR apply to IP as well.

Propagating ADA licenses

Authorized data access (ADA) licenses can be propagated to a new object when the object is created using the **Save As** command. License propagation is determined by relation propagation rules and the values specified for the preference.

In addition, you can propagate ADA licenses using propagation rules. These rules provide a codeless configuration way to copy security data, such as ADA licenses, by using the **Propagation Rules** editor in the Business Modeler IDE.

Scenario – Implementing ADA for IP based on roles and projects

About the Implementing ADA for IP based on roles and projects scenario

In this scenario, Access Manager rules, data classification, and intellectual property (IP) licensing are used to control access to IP data based on roles and projects. In addition, use cases illustrate how the rule tree is evaluated for different users with varying clearance levels, roles, project membership, and IP licensing.

Roles and projects in the Implementing ADA for IP based on roles and projects scenario

The examples in this scenario assume that the ABC Part Company uses roles and projects to control access to data. Roles represent functional areas to which users are assigned, and projects organize data by work program.

ABC Part Company uses the following Teamcenter groups: **Design**, **Engineering**, and **Test Engineering**.

Work is divided into the following projects:

- **Project1**

  A design project that contains some sensitive data.

- **Project2**

  An engineering project that does not contain sensitive data; however, some members of **Project2** require access to specific sensitive files in **Project1** and **Project3**.

- **Project3**

  A test engineering project that does not contain sensitive data; however, some members of **Project3** require access to specific sensitive files in **Project1** and **Project2**.

The following table maps data requirements for these groups and projects.

	Design group	**Engineering group**	**Test Engineering group**
Project1	All design data is assigned to **Project1**, and all users in the **Design** group are members of **Project1**. Access can be granted based on project membership or by using IP licenses.	A subset of engineering data is assigned to **Project1**. Specific users in the **Engineering** group are granted access to specific data in **Project1** using IP licenses.	A subset of test data is assigned to **Project1**. Specific users in the **Test Engineering** group are granted access to specific data in **Project1** using IP licenses.
Project2	A subset of design data is assigned to **Project2**. Specific users in the **Design** group are granted access to specific data using IP licenses.	All engineering data is assigned to **Project2**, and all users in the **Engineering** group are members of **Project2**. Access can be granted based on project membership or by using IP licenses.	A subset of test data is assigned to **Project2**. Specific users in the **Test Engineering** groups are granted access to specific data in **Project2** using IP licenses.
Project3	A subset of design data is assigned to **Project3**. Specific users in the **Design** group are granted access to specific data using IP licenses.	A subset of engineering data is assigned to **Project3**. Specific users in the **Engineering** group are granted access to specific data using IP licenses.	All test data is assigned to **Project3**, and all users in the **Test Engineering** group are members of **Project3**. Access can be granted based on project membership or by using IP licenses.

In addition to the group and project structure employed by ABC Part Company, the following assumptions are made in the examples in this section:

- The value of the **AM_PROJECT_MODE** preference is **false**.

  > Note:
  >
  > If this preference is not set, or if it is set to **false**, only the current active project of the logged on user is evaluated. If the preference is set to **true**, all of the logged on user's active projects are evaluated.

- Data is associated with one group and assigned to one project.

- Sensitive data is assigned a classification value. No assumptions are made about at what level in the data model hierarchy the data is classified or the IP license is attached.

  Siemens Digital Industries Software recommends classifying and licensing at the item level.

- Classified data is contained in **.prt** files associated with **UGMASTER** datasets.

- Teamcenter metadata associated with the classified data must be available to project members even if they do not have clearance to access the geometry data or an IP license that grants them access. Membership in the project is sufficient to allow access to the metadata.

- Users requiring access to classified data are assigned a clearance level that equals or exceeds the classification value of the data that they need to access. Users must also be a member of at least one of the projects with which the data is associated; otherwise, they must be granted access by IP licensing.

Rule tree and ACLs in the Implementing ADA for IP based on roles and projects scenario

The Teamcenter administrator at ABC Part Company has created the **IPAdminACL** ACL, **ConsumerACL** ACL, the **NoAccessACL** ACL, and the **ProjRoleACL** ACL as part of their security implementation.

The **IPAdminACL** ACL used in the rule tree is defined, as follows.

Role	IP Admin	✓
World		✗

The **ConsumerACL** ACL used in the rule tree is defined, as follows.

														Other privileges
Role	Consumer	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	

The **NoAccessACL** ACL used in the rule tree is defined, as follows.

														Other privileges
World	Temp Part Author	✗												

The **ProjRoleACL** ACL used in the rule tree is defined, as follows.

														Other privileges
Role in Projects of Object	Team Author	✓	✓	✓	✓	✓			✓			✓	✓	
Role in Projects of Object	Temp Part Author	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	

Using the **IPAdminACL** ACL, **ConsumerACL** ACL, **NoAccessACL** ACL, and the **ProjRoleACL** ACL, the ABC Part Company has implemented the following rules in the Access Manager rule tree:

- Has Bypass(true) –> ByPassACL
- Has Class(User) –> IPAdminACL
- Has Type(UGMaster)
  - User Is IP Licensed(True) –> ConsumerACL
    - Has Class(Pom_object) –> ProjRoleACL
  - User Is IP Licensed(False) –> NoAccessACL
    - User Has IP Clearance(>=) –> ProjRoleACL

    <subsequent-rules>

Evaluation of the rule tree in the Implementing ADA for IP based on roles and projects scenario

When a user attempts to access data in Teamcenter, the rules in the tree are evaluated as follows:

1. If the user has **By Pass** set, the **ByPassACL** ACL determines access privileges to the object.

   This is a high-level rule that grants system administrator privileges.

2. The **Has Type(UGMaster)** condition is evaluated.

   If the data object type is **UGMaster**, the evaluation proceeds down the rule tree, as follows:

   - The **User Is IP Licensed(true)** condition is evaluated to determine if the user is cited in a valid IP license for the data object.

     If so, the **Has Class(POM_object) –> ProjRoleACL** subbranch is evaluated. If the **Has Class(POM_object) –> ProjRoleACL** rule applies, the **ProjRoleACL** is applied.

If the **Has Class(POM_object) –> ProjRoleACL** rule does not apply, the **ConsumerACL** ACL from the parent rule, **User Is IP Licensed(True) –> ConsumerACL**, is applied.

- If the **User Is IP Licensed(True) –> ConsumerACL** rule does not apply, the evaluation continues to the **User Is IP Licensed(False) –> NoAccessACL** branch.

  If the user, or their group, is not cited in a valid IP license, the **User Has IP Clearance(>=) –> ProjRoleACL** subbranch is evaluated.

  If the user's clearance is greater than or equal to the classification level of the object, the **ProjRoleACL** ACL is applied. If the user's clearance level is less than the classification level of the object, the **NoAccessACL** ACL from the parent rule, **User Is IP Licensed(False) –> NoAccessACL**, is applied.

Use case 1 — Attempted access with privileges

In this use case, the rule tree is evaluated when Robert Smith attempts to access **Part1** in the database. Robert requires read, write, and import privileges to complete his task.

Robert Smith has the following clearance level, role, and project membership.

User	Clearance level	Role	Project	Current project
smithr	2	Team author in Project1	Project1	Project1

In addition, **Part1** is a **UGMaster** dataset. The classification level of **Part1** is **1**, and there are no valid IP licenses associated with **Part1** that cite Robert Smith as an authorized user.

Given this information, the rule tree is evaluated as follows:

1. The **Has Type(UGMaster)** condition is evaluated, and the evaluation proceeds down the tree.

2. The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated.

   There is no valid IP license on **Part1** citing Robert Smith as an authorized user; therefore, the evaluation proceeds down the tree.

   > Note:
   >
   > Because the **User Is IP Licensed(true) –> ConsumerACL** is evaluated as **false**, the **Has Class(Pom_object) –> ProjRoleACL** subbranch is not evaluated.

3. The **User Is IP Licensed(False) –> NoAccessACL** rule is evaluated.

Robert Smith is not IP licensed; therefore, the **User Has IP Clearance(>=) –> ProjRoleACL** subbranch is evaluated. The rule evaluates to **true**, because Robert Smith's clearance level is **2** and the classification level of **Part1** is **1**; therefore, the **ProjRoleACL** ACL is applied.

At this point, the rule tree evaluation has determined that the user has sufficient privileges to perform the requested operation and no further rule tree processing is required.

Use case 2 — Unauthorized export

In this use case, the rule tree is evaluated when Jane Davis attempts an unauthorized export operation on **Part1**.

Jane Davis has the following clearance level, role, and project membership.

User	Clearance level	Role	Role in project	Project	Current project
davisj	1	Consumer	Team Author in Project1	Project1	Project1

In addition, **Part1** is a **UGMaster** dataset. The classification level of **Part1** is **2**, and Jane Davis is cited by a valid IP licenses associated with **Part1**.

Given this information, the rule tree is evaluated as follows:

1. The **Has Type(UGMaster)** condition is evaluated, and the evaluation proceeds down the tree.

2. The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated.

   There is a valid IP license on **Part1** citing Jane Davis as an authorized user; therefore, the **Has Class(Pom_object) –> ProjRoleACL** subbranch is evaluated.

3. The **Has Class(Pom_object) –> ProjRoleACL** rule evaluates to **true**, and the **ProjRoleACL** ACL is applied.

   However, Jane Davis does not fill a role in **Project2** to which **Part1** is assigned; therefore, none of the privileges of the **ProjRoleACL** ACL are applied and the evaluation proceeds down the rule tree.

4. The **User Is IP Licensed(False) –> NoAccessACL** is evaluated.

   Jane Davis is IP licensed; therefore, the **NoAccessACL** ACL is not applied and the subbranch is not evaluated.

   At this point, the rule tree evaluation has determined that the user does not have sufficient privileges to perform the export operation and no further rule tree processing is required.

Use case 3 — Attempted access without IP license

In this use case, the rule tree is evaluated when Doug Abbott, who has a role in the project, attempts to access **Part1** without appropriate clearance or a valid IP license.

Doug Abbott has the following clearance level, role, and project membership.

User	Clearance level	Role in project	Project
abbottd	1	Temp Part Author in Project2	Project2

In addition, **Part1** is a **UGMaster** dataset. The classification level of **Part1** is **2**, and it is assigned to **Project2**. Doug Abbott is not cited by a valid IP licenses associated with **Part1**.

Given this information, the rule tree is evaluated as follows:

1. The **Has Type(UGMaster)** condition is evaluated, and the evaluation proceeds down the tree.

2. The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated.

   There is not a valid IP license on **Part1** citing Doug Abbott as an authorized user; therefore, the **ConsumerACL** ACL is not applied, and the **Has Class(Pom_object) –> ProjRoleACL** subbranch is not evaluated.

3. The **User Is IP Licensed(false) –> NoAccessACL** rule is evaluated.

   This condition is true; therefore, the **User Has IP Clearance(>=) –> ProjRoleACL** subbranch is evaluated.

   Doug Abbott's clearance level is **1**, and the classification of the part is **2**; therefore, the **NoAccessACL** ACL from the parent branch, **User Is IP Licensed(false) –> NoAccessACL**, is applied and access is denied.

Scenario – Implementing ADA for IP based on groups

About the Implementing ADA for IP based on groups scenario

In this scenario, Access Manager rules, data classification, and intellectual property (IP) licensing are used to control access to IP data based on groups. In addition, use cases illustrate how the rule tree is evaluated for different users with varying clearance levels, roles, project membership, and IP licensing.

Groups used in the Implementing ADA for IP based on groups scenario

The examples in this section assume that the ABC Part Company uses groups, which organize users into functional clusters, to control access to data. ABC Part Company uses the following Teamcenter groups:

- **Design**

  Owns both sensitive data and nonsensitive data. Only users who should be allowed access to the sensitive data are granted membership in the **Design** group. Other users who require access are granted limited-time access privileges through an IP license.

- **Engineering**

  Owns both sensitive and nonsensitive data. Only users who should be allowed access to the sensitive data are granted membership in the **Engineering** group. Other users who require access are granted limited-time access privileges through an IP license.

- **Test Engineering**

  Owns both sensitive data and nonsensitive data. Only users who should be allowed access to the sensitive data are granted membership in the **Test Engineering** group. Other users who require access are granted limited-time access privileges through an IP license.

Other assumptions:

- Classification values are applied to sensitive data, and no assumption is made about where in the class hierarchy the information is classified. However, Siemens Digital Industries Software recommends classifying data at the item level of the object model hierarchy.

- Classified data is contained in a **UGMaster** dataset.

- Teamcenter metadata associated with sensitive data must be available to users in the owning group, even if they do not have the clearance required to access the classified data or an IP licensing granting access to the data. Membership in the owning group must be sufficient to establish to right to view metadata.

- Users who require access to classified data are either members of the owning group or are assigned an IP clearance level greater than or equal to the classification of the data to which they require access. Alternatively, users can be granted access using an IP license.

Rule tree and ACLs in the Implementing ADA for IP based on groups scenario

The Teamcenter administrator at ABC Part Company has created the **IPAdminACL** ACL, **ConsumerACL** ACL, the **GroupRoleACL** ACL, and the **NoAccessACL** ACL as part of their security implementation.

The **IPAdminACL** ACL used in the rule tree is defined, as follows.

Role	IP Admin	✓
World		✗

The **ConsumerACL** ACL used in the rule tree is defined, as follows.

														Other privileges
Role	Consumer	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	

The **NoAccessACL** ACL used in the rule tree is defined, as follows.

														Other privileges
World	Temp Part Author	✗												

The **GroupRoleACL** ACL used in the rule tree is defined, as follows.

														Other privileges
Role in Owning Group	Team Author	✓	✓	✓	✓	✓			✓			✓	✓	
Role in Owning Group	Temp Part Author	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	

Using the **IPAdminACL** ACL, **ConsumerACL** ACL, **NoAccessACL** ACL, and the **GroupRoleACL** ACL, the ABC Part Company has implemented the following rules in the Access Manager rule tree:

- Has Bypass(true) –> ByPassACL
- Has Class(User) –> IPAdminACL
- Has Type(UGMaster)
  - User Is IP Licensed(True) –> ConsumerACL
    - Has Class(Pom_object) –> GroupRoleACL
  - User Is IP Licensed(False) –> NoAccessACL

User Has IP Clearance(>=) –> ProjRoleACL

<subsequent-rules>

Evaluation of the rule tree in the Implementing ADA for IP based on groups scenario

When a user attempts to access data in Teamcenter, the rules in the tree are evaluated as follows:

1. If the user has **By Pass** set, the **ByPassACL** named ACL determines access privileges to the object.

   This is a high-level rule that grants system administrator privileges.

2. The **Has Type(UGMaster)** condition is evaluated.

   If the data object type is **UGMaster**, the evaluation proceeds down the rule tree, as follows:

   - The **User Is IP Licensed(true)** condition is evaluated to determine if the user is cited in a valid IP license for the data object.

     If so, the **Has Class(POM_object) –> GroupRoleACL** subbranch is evaluated and if valid, the **GroupRoleACL** ACL is applied. If the subbranch is not valid, the **ConsumerACL** ACL from the parent rule, **User Is IP Licensed(true) –> ConsumerACL**, is applied.

   - If the **User Is IP Licensed(true)** condition is not met, the **User Is IP Licensed(false)** condition is evaluated, and the **User Has IP Clearance(>=) –> GroupRoleACL** subbranch is evaluated.

     If the user's clearance is greater than or equal to the classification level of the object, the **GroupRoleACL** ACL is applied. If the subbranch is not valid, the **NoAccessACL** ACL from the parent branch, **User Is IP Licensed(false) –> NoAccessACL**, is applied.

Use case 1 — Attempted access with privileges

In this use case, the rule tree is evaluated when Robert Smith attempts to access **Part1** in the database. Robert requires read, write, and import privileges to complete his task.

Robert Smith has the following clearance level, role, and group membership.

User	Clearance level	Role in Group	Group
smithr	2	Team Author	Design

In addition, **Part1** is a **UGMaster** dataset that is owned by the **Design** group. The classification level of **Part1** is **1**, and there are no valid IP licenses associated with **Part1** that cite Robert Smith as an authorized user.

Given this information, the rule tree is evaluated as follows:

1. The **Has Type(UGMaster)** condition is evaluated, and the evaluation proceeds down the tree.

2. The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated.

   There is no valid IP license on **Part1** citing Robert Smith as an authorized user; therefore, the evaluation proceeds down the tree.

   > Note:
   >
   > The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated as **false**; therefore, the **Has Class(Pom_object) –> GroupRoleACL** subbranch is not evaluated.

3. The **User Is IP Licensed(False) –> NoAccessACL** rule is evaluated.

   Robert Smith is not IP licensed; therefore, the **User Has IP Clearance(>=) –> GroupRoleACL** subbranch is evaluated.

   The rule evaluates to **true**, because Robert Smith's clearance level is **2** and the classification level of **Part1** is **1**. The **GroupRoleACL** ACL grants **Read**, **Write**, and **Import** privileges to the user.

At this point, the rule tree evaluation has determined that the user has sufficient privileges to perform the requested operation and no further rule tree processing is required.

Use case 2 — Unauthorized export

In this use case, the rule tree is evaluated when Jane Davis attempts an unauthorized export operation on **Part1**.

Jane Davis has the following clearance level, role, and group membership.

User	Clearance level	Role	Role in group	Group
davisj	1	Consumer	Team Author in Design group	Design

In addition, **Part1** is a **UGMaster** dataset owned by the **Engineering** group. The classification level of **Part1** is **2**, and Jane Davis is cited by a valid IP license associated with **Part1**.

Given this information, the rule tree is evaluated as follows:

1. The **Has Type(UGMaster)** condition is evaluated, and the evaluation proceeds down the tree.

2. The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated.

   There is a valid IP license on **Part1** citing Jane Davis as an authorized user; therefore, the **HasClass(Pom_object) –> GroupRoleACL** subbranch is evaluated.

The **HasClass(Pom_object) –> GroupRoleACL** rule evaluates to **true**, and the **GroupRoleACL** ACL is applied.

However, Jane Davis does not fill a role in the **Engineering** group by which **Part1** is owned; therefore, none of the privileges of the **GroupRoleACL** ACL are applied, and the evaluation proceeds down the rule tree.

At this point, the rule tree evaluation has determined that the user does not have sufficient privileges to perform the export operation and no further rule tree processing is required.

Use case 3 — Attempted access without valid license

In this use case, the rule tree is evaluated when Doug Abbott, who has a role in the owning group, attempts to access **Part1** without appropriate clearance or a valid IP license.

Doug Abbott has the following clearance level, role, and group membership.

User	Clearance level	Role in group	Group
abbottd	1	Temp Part Author in Engineering group	Engineering

In addition, **Part1** is a **UGMaster** dataset. The classification level of **Part1** is **2**, and it is owned by the **Engineering** group. Doug Abbott is not cited by a valid IP license associated with **Part1**.

Given this information, the rule tree is evaluated as follows:

1. The **Has Type(UGMaster)** condition is evaluated, and the evaluation proceeds down the tree.

2. The **User Is IP Licensed(true) –> ConsumerACL** rule is evaluated.

   There is not a valid IP license on **Part1** citing Doug Abbott as an authorized user; therefore, the **ConsumerACL** ACL is not applied and the **Has Class(Pom_object) –> GroupRoleACL** subbranch is not evaluated.

   The evaluation continues down the tree.

3. The **User Is IP Licensed(false) –> NoAccessACL** rule is evaluated.

   This condition is true; therefore, the **User Has IP Clearance(>=) –> GroupRoleACL** subbranch is evaluated.

   Doug Abbott's clearance level is **1** and the classification of the part is **2**; therefore, the rule is not valid.

   The **NoAccessACL** ACL from the parent rule, **User Is IP Licensed(false) –> NoAccessACL**, is applied and access is denied.

Scenario – Implementing ADA for IP at the dataset level

Scenario – Implementing ADA for IP at the dataset level

The following use cases illustrate how Access Manager rules can be configured to control access to classified intellectual property (IP) data. Both examples assume that the dataset is the object that is classified and that the item or item revision and any corresponding BOM views are viewable by any user.

Use case 1 – Using levels of user clearance

In this example, the ABC Part Company is implementing rules to secure sensitive data. To do this, they have established two levels of user clearance, **secret** and **super secret**, and one level of IP (data) classification, **secret**. In addition, they have created an IP license, **lic 1** that names Robert Smith as an authorized user. They have applied these clearances and classifications as follows.

User	Clearance level
Robert Smith (smithr)	None
Jane Davis (davisj)	**secret**
Peter Taylor (taylorp)	**super secret**

Dataset ID	Classification level	License
ABC0001	None	None
ABC0002	**secret**	None
ABC0003	**secret**	**license 1**

Based on the clearance, classification, and licensing, ABC Part Company has implemented three access control lists (ACLs): **IPAdminACL** ACL, **IPACL** ACL, and **NoIPACL** ACL.

The **IPAdminACL** ACL used in the rule tree is defined, as follows.

		
Role	IP Admin	
World		

The **IPACL** ACL is defined as follows.

Owning User		✓										
User IP Licensed		✓										
User Under IP Clearance		✗										
User over clearance											✓	
Role in Owning Group	IP Admin											✓
World											✗	✗

The **NoIPACL** ACL is defined as follows.

Role in Owning Group	IP Admin											✓
World											✓	✗

The **IPACL** ACL and the **NoIPACL** ACL are used in the following rules:

- Has Class (POM_object)
  - Has Bypass (true) –> Bypass
  - Has IP Classification() –> IPACL
  - Has No IP Classification() –> NoIPACL
  - Has Class(User) -> IPAdminACL
  - Has Class(POM_object) –> System objects

In addition to the clearance levels, classification levels, ACLs, and rules, ABC Part Company can set the **TC_session_clearance** preference to provide one of three levels of monitoring: **unset**, **blocking**, or **logging**. The **Unmanage** privilege can be granted to allow users to circumvent the blocking, if applicable.

The results of the security implementation described in this section are displayed in the following table. The table lists results for each user, at each level of session clearance monitoring, for each dataset.

Session Clearance	User	Clearance	Dataset	Classification	License	Read	Write	IP Logged	User can unmanage	Unmanage
block	smithr	none	ABC0001	none	none	Y	Y	N	Y	Y
block	davisj	none	ABC0001	none	none	Y	Y	N	Y	Y
block	taylorp	none	ABC0001	none	none	Y	Y	N	Y	Y
block	smithr	none	ABC0002	secret	none	N	N/A	N/A	N/A	N/A
block	davisj	secret	ABC0002	secret	none	Y	Y	Y	N	N
block	taylorp	super secret	ABC0002	secret	none	Y	Y	Y	Y	Y
block	smithr	none	ABC0003	secret	lic 1	Y	Y	Y	N	N
block	davisj	secret	ABC0003	secret	lic 1	Y	Y	Y	N	N
block	taylorp	super secret	ABC0003	secret	lic 1	Y	Y	Y	Y	Y
log	smithr	none	ABC0001	none	none	Y	Y	N	Y	Y
log	davisj	secret	ABC0001	none	none	Y	Y	N	Y	Y
log	taylorp	super secret	ABC0001	none	none	Y	Y	N	Y	Y
log	smithr	none	ABC0002	secret	none	N	N/A	N/A	N/A	N/A
log	davisj	secret	ABC0002	secret	none	Y	Y	Y	Y	Y
log	taylorp	super secret	ABC0002	secret	none	Y	Y	Y	Y	Y
log	smithr	none	ABC0003	secret	lic 1	Y	Y	Y	Y	Y
log	davisj	secret	ABC0003	secret	lic 1	Y	Y	Y	Y	Y
log	taylorp	super secret	ABC0003	secret	lic 1	Y	Y	Y	Y	Y
unset	smithr	none	ABC0001	none	none	Y	Y	N	Y	Y

Session Clearance	User	Clearance	Dataset	Classification	License	Read	Write	IP Logged	User can unmanage	Unmanage
unset	davisj	secret	ABC0001	none	none	Y	Y	N	Y	Y
unset	taylorp	super secret	ABC0001	none	none	Y	Y	N	Y	Y
unset	smithr	none	ABC0002	secret	none	N	N/A	N/A	N/A	N/A
unset	davisj	secret	ABC0002	secret	none	Y	Y	N	Y	N
unset	taylorp	super secret	ABC0002	secret	none	Y	Y	N	Y	Y
unset	smithr	none	ABC0003	secret	lic 1	Y	Y	N	Y	Y
unset	davisj	secret	ABC0003	secret	lic 1	Y	Y	N	Y	Y
unset	taylorp	super secret	ABC0003	secret	lic 1	Y	Y	N	Y	Y

Use case 2 – Using the super-secret role

In this example, the ABC Part Company is implementing rules to secure sensitive data using a variation on the security scheme implemented in use case 1.

To do this, they have established one level of user clearance, **secret**, and they have created a new role, **super-secret**. (The **super-secret** role provides functionality equivalent to the **super secret** user clearance level in use case 1.)

Based on this clearance level and role, ABC Part Company has implemented the **IPACL** ACL, as follows.

Owning User		✓										
User IP Licensed		✓										
User Under IP Clearance		✗				✗	✗	✗	✗	✗		
Role in Owning Group	super-secret										✓	

Role in Owning Group	IP Admin											✓
World											✗	✗

The difference between the definition of the **IPACL** ACL in this use case and the ACL in use case 1 is that the **User Over Clearance** accessor is replaced with the **Role in Owning Group(super-secret)** accessor.

The benefit of use case 2 is that it leaves no possibility of a higher level of security being applied to the data. It also enables you to limit the use of the **super-secret** role to a single group or project.

In both use cases, using Access Manager to assign **unmanage** privileges enables NX to read the blocking property.

Implementation considerations

Access Manager considerations for IP

Placement of rules in the Access Manager rule tree

Access Manager implicitly grants access privileges to data unless privileges are explicitly revoked; therefore, rules to identify users with IP clearance or license should be placed high in the rule tree. For example, placing IP rules higher than the **Has Object ACL** condition prevents users from retaining privileges to something that is later classified. If neither IP-related condition applies, the rule tree is evaluated and data access privileges are determined by other factors, such as project membership or object ownership, as the rule tree is evaluated.

Authorized data access rule precedence for ITAR and IP

When both ITAR and IP access rules are in force at a site, ITAR rules must take precedence over IP rules. Therefore, you must place ITAR rules higher in the rule tree than IP rules.

NX security for classified data for ITAR and IP

When operating in Teamcenter Integration for NX mode, Teamcenter provides security metadata to NX. This metadata is computed using the user and data properties and directs how NX behavior is modified when processing secure data. Security metadata is constant in NX, even if the information changes in Teamcenter. For example, if a user is granted access to data in NX based on a license, and the license expires while the data is open in NX, the expiration is not recognized by NX until the user attempts to save the data back to Teamcenter, at which point the operation fails because the user's license has expired.

Authorized data access logging and blocking supports NX datasets as well as non-NX datasets that can be translated into a part file, including JT, Solid Edge, and CATIA data.

The following NX operations are affected when logging and blocking are implemented.

Operation	Blocked and logged	Blocked
File→Save As	X	
File→Print	X	
File→Plot	X	
File→Send to Package File	X	
File→Export→Part	X	
File→Export→Parasolid	X	
File→Export→User Defined Feature	X	
File→Export→CGM	X	
File→Export→STL	X	
File→Export→Polygon File	X	
File→Export→Author HTML	X	
File→Export→Teamcenter Visualization	X	
File→Export→VRML	X	
File→Export→PNG	X	
File→Export→JPEG	X	
File→Export→GIF	X	
File→Export→TIFF	X	
File→Export→BMP	X	
File→Export→XWD	X	
File→Export→IGES	X	
File→Export→STEP203	X	
File→Export→STEP214	X	
File→Export→DXF/DWG	X	
File→Export→2D exchange	X	
File→Export→Heal Geometry	X	
File→Export→V4 Catia	X	

Operation	Blocked and logged	Blocked
File→Export→V5 Catia	X	
File→Export drawings to Teamcenter	X	
File→Interoperate	X	
File→Collaborate	X	
Edit→Copy Display	X	
View→Visualization→High Quality Image	X	
View→Visualization→Create Animation	X	
Tools→NX Manager→Export Assembly	X	
Tools→NX Manager→Save Outside Teamcenter	X	
Tools→Part Families Create		X
Tools→Part Family Update		X
Assemblies→Components→Create New	X	
Assemblies→Components→Add Existing		X
Assemblies→Components→Substitute Component		X
Assemblies→Components→Part Family Update		X
Assemblies→Components→Create New Parent		X
Assemblies→Cloning→Create Clone Assembly	X	
Edit→Paste [Pasting of a component]		X
Dragging a drawing template onto a part		X

Multi-Site Collaboration considerations

Take care when sharing classified data with other Teamcenter sites.

Caution:

It is possible that when importing a mixture of classified and unclassified data, the user who performs the import could view data to which they do not have clearance. Teamcenter tests for privileges when importing data, but the system does not read the classification attribute until the object has been reassembled and saved at the importing site.

In addition, to ensure that classification and license attributes are retained when classified data is imported to a remote site, you should first verify that the clearance and classification values and AM rule tree entries are synchronized between the sites.

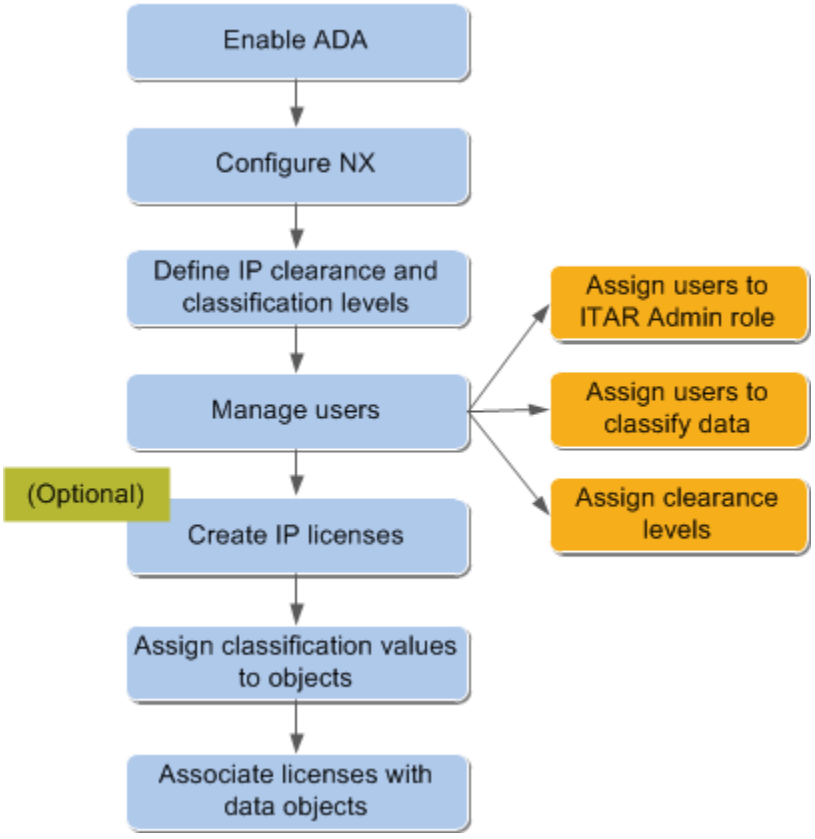
The following points should also be considered when working with classified data in a Multi-Site Collaboration environment:

- Licenses are defined independently at each site and must be matched when data is imported.
- Licenses are not automatically exported with data objects; however, references to the licenses are exported with the data object.
- The **TC_multi_site_ada_license_user_bypass** preference can be set to allow licenses to be re-attached to replicated objects upon import, even when the importing user does not have the privileges required to attach such a license.
- If a license with the same ID as that referenced by the data object exists at the importing site, the association is maintained.
- If a data object references a license ID that does not exist at the importing site, there are no IP privileges on the object when it is imported.
- Classification values assigned to data are preserved when the data is imported.
- The **ADA_override_on_import** preference can be set to keep IP and government classification values on a workspace object in sync between the master and replica sites.

Basic tasks for configuring and administering ADA for IP data

About the tasks for configuring and administering authorized data access for IP data

The following basic tasks must be performed when configuring and administering authorized data access for intellectual property (IP) data:



Enabling authorized data access

Authorized data access (ADA) features are enabled by default when you install Teamcenter. To disable ADA, set the **ADA_enabled** preference to **false**.

Configuring logging and blocking in NX

The **TC_session_clearance** preference, in conjunction with NX runtime properties, enables you to establish indirect access controls in NX using logging or menu suppression (blocking) to control classified data that is loaded in a Teamcenter Integration for NX session.

Logging provides auditable evidence of the use of various NX commands on classified data. This is implemented by NX internal mechanisms that are beyond the scope of this document. For more information, see the NX Help Library.

Blocking suppresses NX menus that, if used on classified data, could result in exporting geometric data outside the NX/Teamcenter managed environment. The blocking feature also provides menu action logging.

While allowing data out of the managed environment creates a security vulnerability, you may at times want to grant a user permission to use NX menus that involve exporting data out of the environment. You can use the **Unmanage** privilege in Access Manager to grant users the ability to access these restricted features.

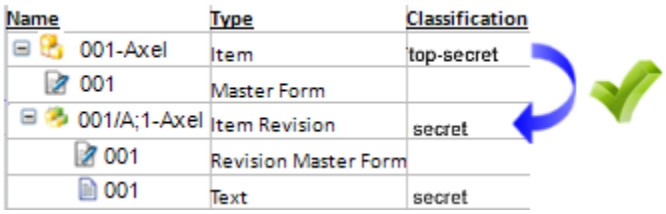
Configuring logging and blocking is optional.

Defining IP clearance and classification levels

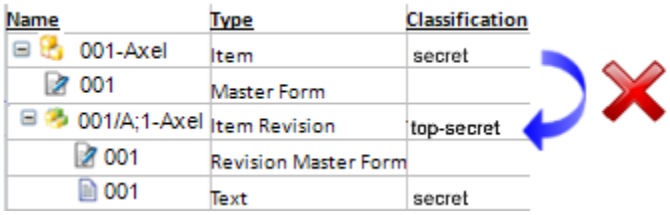
Each site defines a list of intellectual property (IP) classification values and clearance levels that are assigned to data objects and users for IP access evaluation. The list is maintained in the **IP_level_list_ordering** preference. For example, you define a list of the **secret**, **top-secret**, and **super-secret** access levels. The order in which the levels appear in **IP_level_list_ordering** defines their restrictiveness. The first row is the lowest classification, and the last row is the highest. Access Manager compares these values to determine user access rights to an object.

The IP classification values are propagated to related objects according to the propagation rules defined in Teamcenter (when setting the value on an item, all revisions and their attached datasets gets the same value). This only applies, however, when setting a value that is *more restrictive* than the current value. If you set a less restrictive value on an object, the value is not propagated to the related objects.

For example, if you set a classification value of **top-secret** on the **001–Axel** part, it is propagated to the related objects because its classification is more restrictive than the IP classification of the related objects (**secret**):

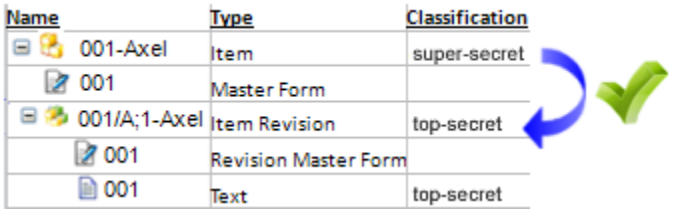


In the following example, however, if you set an IP classification value of **secret** on the **001–Axel** part, it is *not* propagated from **001–Axel** to the related objects because it is less restrictive (at a lower level) than the IP classification value set on the related objects (**top-secret**):



You can replace an IP classification value with another of equal value. For example, in the following example, **top-secret** is replaced with **super-secret** because it is at the same level.

> Note:
>
> Classification values that are of equal value are separated by commas in the **IP_level_list_ordering** preference.



Assigning IP Admin role and grant IP Admin privileges

The **IP Admin** privilege is required to specify IP classification on data objects. The privilege to administer IP licenses is defined by the value specified for the **ADA_license_administration_privilege** preference (**ITAR_ADMIN** by default). The same privilege is required to add and remove users from licenses and set license expiration dates, typically by either being the owning user or being assigned to the **IP Admin** role.

Applying licenses to or removing licenses from objects requires that the user have **IP Admin** privileges on both the license and the object to which the license is being applied.

Users can be assigned to the **IP Admin** role by a Teamcenter administrator using the Organization application.

> Note:
>
> License creation is not controlled by Access Manager rules.

Assigning users to classify data

Use the Organization application to assign users to the **IP Classifier** role to allow them to classify IP information without granting all of the privileges of the **IP_ADMIN**. Users of these roles must be granted the **IP_Classifier** privilege by the customer rule tree modifications.

Comparison of roles

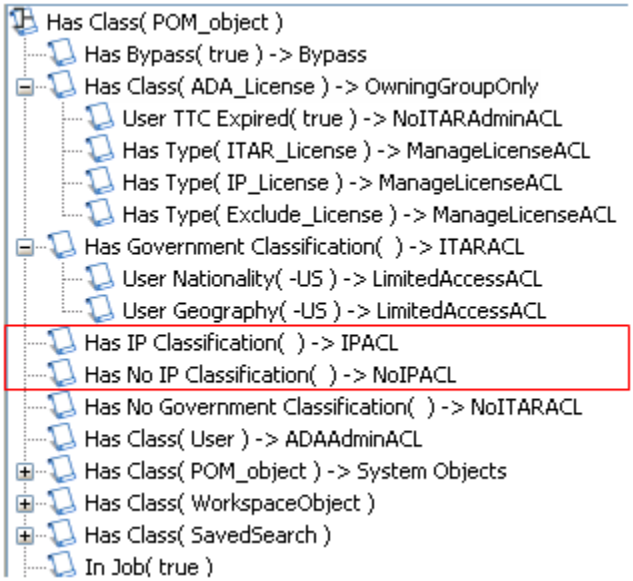
The following table provides information at a high level about the actions that users with **ITAR_ADMIN**, **IP_ADMIN**, **ITAR_Classifier**, and **IP_Classifier** privileges can perform (there are minor exceptions for attaching or detaching licenses):

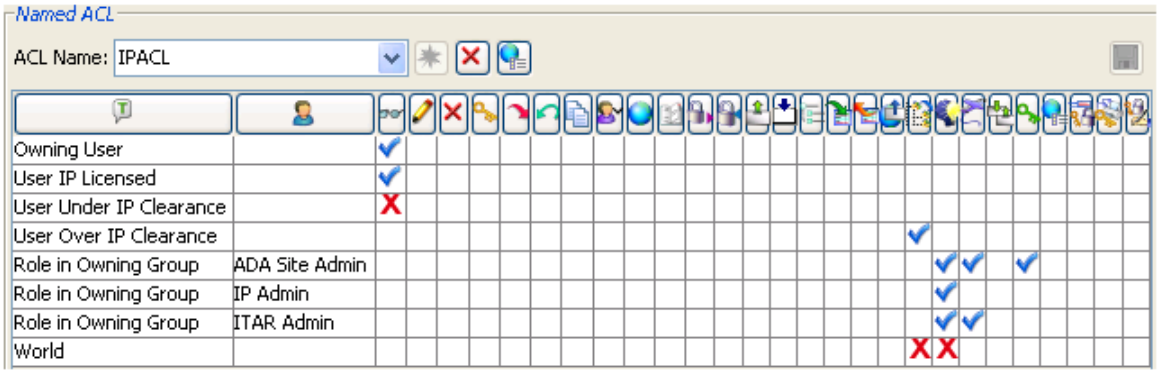
Action	ITAR_ADMIN	IP_ADMIN	ITAR_Classifier	IP_Classifier
Create ITAR, IP, or exclude license	Yes, if preference **ADA_license_ administration_**	Yes, if preference **ADA_license_ administration_**	No	No

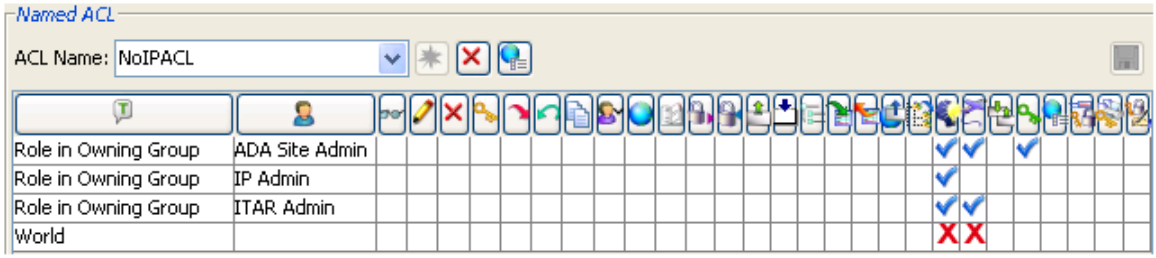
Action	ITAR_ADMIN	IP_ADMIN	ITAR_Classifier	IP_Classifier
	privilege is set to **ITAR_ADMIN**	**privilege** is set to **IP_ADMIN**		
Modify ITAR, IP, or exclude license	Yes, if preference **ADA_license_ administration_ privilege** is set to **ITAR_ADMIN**	Yes, if preference **ADA_license_ administration_ privilege** is set to **IP_ADMIN**	No	No
Delete ITAR, IP, or exclude license	Yes, if preference **ADA_license_ administration_ privilege** is set to **ITAR_ADMIN**	Yes, if preference **ADA_license_ administration_ privilege** is set to **IP_ADMIN**	No	No
Attach ITAR license to a workspace object	Yes	No	No	No
Attach IP or exclude license of workspace object	No	Yes	No	No
Detach ITAR license from workspace object	Yes	No	No	No
Detach IP or exclude license from workspace object	No	Yes	No	No
Set or modify government classification on workspace object	Yes	No	Yes	No
Set or modify IP classification on workspace object	No	Yes	No	Yes
Set or modify IP clearance for a user	No	Yes	No	No

Recommended rules for intellectual property (IP) classification of workspace objects

The following are examples of recommended rules and access control lists (ACLs) for users performing IP classification of workspace objects.







Assigning clearance levels to users

Use the Organization application to assign IP clearance levels to users. The clearance value assigned to the user must match a value defined in the **IP_level_list_ordering** preference.

Creating IP licenses

Users who have the same privilege as the value set for the **ADA_license_administration_privilege** preference can create and maintain intellectual property (IP) licenses in the ADA License application. Once created, privileges are either granted or denied to users by associating the license directly with the data object.

Assigning classification values to data objects

Overview of assigning classification values to data objects

Access to classified data is determined by an evaluation of the user's clearance level and the classification level that is applied to the object. The **IP Classification** property specifies the classification level of an individual object and can be viewed and modified in the same manner as other object properties. Valid values for this property are derived from the **IP_level_list_ordering** preference. Users must have **IP Admin** or **IP Classifier** privileges to classify objects in Teamcenter.

> Note:
>
> The **Classification**, **Classified**, and **Classified in** properties apply to the Teamcenter Classification application, which is used to categorize objects for reuse. They do not apply to classification for the purpose of authorizing data access.

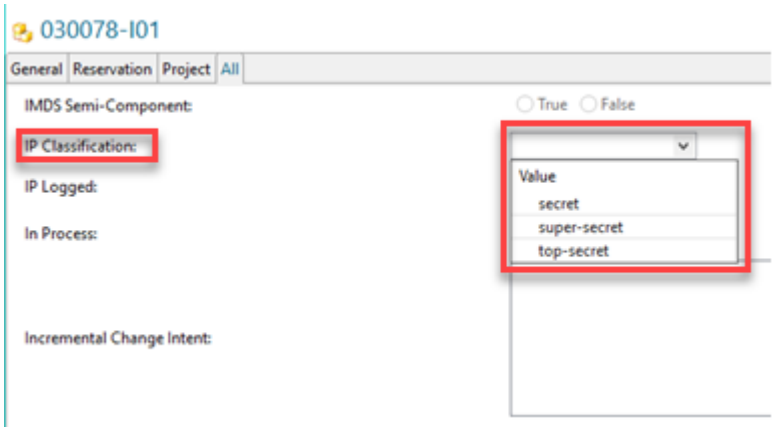
Assign an IP classification value to an object

> Tip:
>
> You can also display and modify object properties in the **Details** pane by selecting the object and choosing **Details** from the list in the upper-right corner of the Teamcenter window. This method allows you to assign a classification value without performing the extra steps required to display the **Properties** dialog box.

1. In the tree display in My Teamcenter, right-click the object to be classified.
2. From the shortcut menu, choose **Edit Properties**.

   Teamcenter checks the object out of the database and opens it for edit.
3. Click the **All** tab to display all of the object properties.
4. Scroll to locate the **IP Classification** property.
5. Select the IP classification value from the drop-down list.

   The drop-down list values are specified in the **IP_level_list_ordering** preference.



6. Check in the object.

Associating licenses with data objects

Introduction to associating licenses with data objects

IP licenses grant named users discretionary, limited-time access to classified data. Users (IP administrators) who are assigned to the **IP Admin** role maintain them. They have **IP Admin** privileges to both the object and the license. IP administrators also attach licenses to data objects.

> Note:
>
> - You must have **IP Admin** privileges on both the object and the license to attach or remove licenses.
> - You may want to set the **ADA_enable_subgroups** preference to specify whether or not subgroup members of the top-level groups can be authorized access to workspace objects with attached licenses when the top-level group is not authorized access.

Attach IP licenses to a data object

1. In the My Teamcenter tree pane, right-click the object to which you want to attach the license.
2. From the shortcut menu, choose **License→Attach**.
3. In the **Assign an Object to Licenses** dialog box, select the license that you want to attach to the object.
4. Click **OK**.

Configuring ADA for ITAR support

About configuring ADA for ITAR support

Teamcenter allows you to control access to data that is deemed military in nature. For example, technical information may be subject to International Traffic in Arms Regulations (ITAR) policies and, if so, must be protected so that only citizens of the United States have open access to the data. Viewing this classified data outside the U.S. is considered equivalent to performing an ADA (authorized data access) export of it, which requires that rights are granted by license. Teamcenter can grant access rights for a specific time period to citizens of other countries, U.S. citizens physically located outside the United States, or organizations that are named in an effective Technical Assistance Agreement (TAA). Information marked as *nontechnical* is, for the purposes of ITAR, available to all users.

> Note:
>
> Do not confuse an ADA export with other Teamcenter export actions. For example, a Multi-Site Collaboration export from a site within the United States to another U.S. site is not an ADA export.

If you implement ITAR controls at your site, an ITAR administrator can use Access Manager to control each user's rights to read or modify the status of technical data or ADA export it, based on ITAR markings, TAA licenses, and the user's citizenship. You can mark any object in the system (including parts, design files, and documents) as *government classified*, thus enabling ITAR access control on it. ITAR markings are internationally recognized codes including the USML (United States Munitions List) and the ECCN (Export Control Classification Number). The mechanism of assigning codes to data is determined by your company's business practices and is not controlled by Teamcenter.

You can define the following attributes for each user:

- A nationality, citizenship (one or more), and geography (physical location). Teamcenter uses these attributes to determine if the user is a foreign (non-U.S.) national for ITAR purposes or a U.S. national located outside the U.S. In either of these cases, if the user views classified material, it is considered an ADA export and requires a license granting rights to export.

- Government clearance status and technology transfer certification (TTC) date to track the user's level of clearance for viewing data marked as government classified. Teamcenter revokes the user's access rights after the TTC date expires unless renewed. The administrator may manually cancel a TTC at any time.

In the same way, Teamcenter tracks the geography of each site in the system and can consequently determine if a site that requests data is domestic or foreign. It uses this information with the user attributes to determine if the requesting user is a foreign national or a domestic national.

The ITAR administrator creates each TAA as a separate license that is assigned to a named group (a corporation or other organization). Teamcenter tracks all groups that are issued with TAAs and their expiry dates.

For details of how to maintain groups, geography, and licenses, see *Organization Management Using Groups, Roles, and Users*.

You can also configure logging and menu suppression (blocking) to be applied when classified data is loaded in Teamcenter Integration for NX. Logging provides an audit of actions taken on exported data, and blocking suppresses NX menus to prevent geometric data from being exported outside of the NX/Teamcenter environment.

Applying Access Manager concepts to technical data subject to ITAR

Basic Access Manager concepts and terms related to ITAR

The following concepts are key to understanding how to apply Access Manager concepts to data subject to ITAR.

Concept	Description
Government classification	Government classification is an object attribute that can be used in conjunction with the user's government clearance level to validate their access privileges. This requires that the government clearance on the user and the government classification on the object come from a common multilevel scheme defined by the **ITAR_level_list_ordering** preference. The government classification attribute can also be used to describe the object. Descriptive classification values are typically United States Munitions List (USML) codes or Export Control Classification Number (ECCN) codes. When used in this way, the value of the **ITAR_level_list_ordering** preference is null.
Government clearance	Government clearance is a user attribute that specifies the level of clearance that users have to classified data. Note: Valid classification and clearance values are derived from the values set for the **ITAR_level_list_ordering** preference.
Nationality and citizenship	Nationality and citizenship are attributes that specify the nationality and citizenship of a user or organization (group) and enables access rules to determine whether the user or organization is domestic or whether an ITAR license is required. Appropriate values for this field are two-character ISO 3166 codes. These attributes are evaluated when the Access Manager rule tree includes rules that use the **User Nationality**, **Group Nationality**, **User Citizenship**, or **User Citizenship Or Nationality** condition.
Geography	Geography is an attribute that specifics the geographic location of the user or site. International Traffic in Arms Regulations specify that U.S. nationals located outside the United States must be named in an effective Technical Assistance Agreement

Concept	Description
	(TAA) to access classified data. Appropriate values for this field are two-character ISO 3166 codes. This attribute is evaluated when the Access Manager rule tree includes rules that use the **User Geography** or **Site Geography** condition.
Organization	An *organization* in the authorized data access context is a Teamcenter group that models a legal corporation or company whose members can be granted access to classified data. An organization has the following attributes that are derived from ISO 6253 standards: **Organization Name**, **Organization Legal Name**, **Organization Alternate Name**, **Organization Address**, **Organization URL**, **Organization Status**, **Organization ID**, **Organization Type**, **Nationality**. Note: Organizations are Teamcenter groups and as such are subject to hierarchical group behavior. Therefore, sub-groups inherit the properties of the parent group. Siemens Digital Industries Software recommends that users be made members of the most specific group possible to ensure the accuracy of key attributes, such as the nationality of the organization.
Technical Assistance Agreement	Authorizing document that allows the export of classified data to U.S. nationals located outside the United States or to foreign nationals. Note: In the context of authorized data access (ADA), allowing access to classified data is considered to be an *export*. The use of *export* in this context is unrelated to transferring data between Teamcenter sites.
Technology Transfer Certification Date	Date upon which the user's access to view data marked as government classified expires.
ITAR license	Grants access for U.S. nationals outside the United States or foreign nationals named by an effective Technical Assistance Agreement to access protected product data, which in the United States could contain technical information that is restricted by ITAR.
ITAR Admin privilege	ITAR Admin privileges must be explicitly granted and denied to prevent user from gaining unauthorized access to ITAR data.
ITAR Classifier privilege	The **ITAR Classifier** privilege lets users classify data without having all the privileges of the ITAR Admin. It must be explicitly granted and denied to prevent users from gaining unauthorized access to ITAR data.

Access Manager rules for protecting data subject to ITAR

Access Manager rules allow you to establish access controls on data that is subject to ITAR. The following rule conditions and accessor types are used to configure ITAR data access rules.

Condition	Value	Description
User Citizenship	Two-character ISO 3166 codes identifying a country. This condition accepts negation using a minus (–) prefix. For example, **–us** indicates any user without U.S. citizenship.	Checks whether the user has specific citizenship.
User Citizenship Or Nationality	Two-character ISO 3166 codes identifying a country. This condition accepts negation using a minus (–) prefix. For example, **–us** indicates any user without U.S. citizenship.	Checks whether the user has specific citizenship or nationality.
User Nationality	Two-character ISO 3166 codes identifying a country. This condition accepts negation using a minus (–) prefix. For example, **–us** indicates any user not from the U.S.	Specifies the nationality of a user.
Group Nationality	Two-character ISO 3166 codes. This condition accepts negation using a minus (–) prefix. For example, **–us** indicates any user belonging to a group not from the U.S.	Specifies the nationality of a group or organization.
User Geography	Two-character ISO 3166 codes. This condition accepts negation using a minus (–) prefix. For example, **–us** indicates any user not located in the U.S.	Specifies the location of the user.

Condition	Value	Description
Site Geography	Two-character ISO 3166 codes. This condition accepts negation using a minus (–) prefix. For example, **–us** indicates any user at a site not located in the U.S.	Specifies the location of the site.
User Is ITAR Licensed	**true** or **false**	If set to **true**, verifies the existence of a valid ITAR license attached to the workspace object being evaluated that names the current user or their group as a licensee.
Has Government Classification	Specific government classification attribute values that can be prefixed by the following operators: **>** **>=** **<** **<=** **=**	Validates the government classification attribute value of the object against the value specified for the condition. The operators can be used without a classification value; the government classification attribute of the object is compared to the user's clearance level based on the specified operator. Note: If the object has no government classification attribute value, this rule does not apply.
Has No Government Classification		Matches if the object has a null value for the government classification attribute.
User Has Government Clearance	Specific government clearance values that can be prefixed by the following operators: **>** **>=** **<** **<=** **=**	Validates the user's government clearance level against the value specified for the condition. The operators can be used without a clearance value; the user's clearance is compared to the government classification attribute of the object based on the specified operator.
User TTC Expired		Evaluates whether the TTC date for the user who is logged on has expired.
User In License		Specifies whether the user who is logged on is cited on the license object being evaluated.

Condition	Value	Description
User In Attached ITAR License	**Any** or **All**	Checks whether the user is on any or all ITAR licenses attached to the workspace object being evaluated. • If set to **Any**, the user is on at least one ITAR license. • If set to **All**, the user is on all ITAR licenses.
Has Named ITAR License	*License ID*	Checks whether an object has an ITAR license of the specified license ID. It does not check if a user is on the license.
User In Named ITAR License	*License ID*	Checks whether a user is on an ITAR license of the specified license ID. It does not check if the license is attached to the workspace object being evaluated.
Citizenship On Any ADA Lic	*citizenship*	Checks whether any or all of the citizenships of the user currently logged on matches any of the citizenships on the ADA licenses attached to the workspace object being evaluated.
Citizenship On Any ITAR Lic	*citizenship*	Checks whether any or all of the citizenships of the user currently logged on matches any of the citizenships on the ITAR licenses attached to the workspace object being evaluated.
Citizenship On Any IP Lic	*citizenship*	Checks whether any or all of the citizenships of the user currently logged on matches any of the citizenships on the IP licenses attached to the workspace object being evaluated.
Citizenship On Any Exclude Lic	*citizenship*	Checks whether any or all of the citizenships of the user currently logged on matches any of the citizenships on the exclude licenses attached to the workspace object being evaluated.

Accessor	Description
User Excluded	The user or their group is cited in a valid exclude license attached to the workspace object being evaluated. The exclude license is intended to revoke access on a limited time basis.
User ITAR Licensed	The user is cited in any current ITAR license associated with the selected workspace object.

Accessor	Description
User ITAR Unlicensed	The user is not cited in any current ITAR license associated with the selected workspace object.
User Under Government Clearance	The user's government clearance is below the level required by the object. This accessor is typically used to revoke access and is only applicable when the government clearance on the user and the government classification on the object come from a common multilevel scheme defined by the **ITAR_level_list_ordering** preference.
User Has Government Clearance	Compares the user's clearance with the object classification and tests whether the user has clearance above, below, or equal to that required to access the object.
User Over Government Clearance	The user's government clearance is over the level required by the object. This accessor is typically used to grant access and is only applicable when the government clearance on the user and the government classification on the object come from a common multilevel scheme defined by the **ITAR_level_list_ordering** preference.

Scenario – Using rules to control access to data through ADA licenses

About using rules to control access to data through ADA licenses scenario

The following use cases illustrate how to use Access Manager rule conditions to manage access to data using Authorized Data Access (ADA) licenses. These scenarios use International Traffic in Arms Regulations (ITAR) examples. There are also corresponding rule conditions for managing access to data using all types of licenses so these concepts pertain to any license type.

- Use case — Checking for users on named licenses
- Use case — Checking for users on a given type of license
- Use case — Checking for attached license by name

Use case — Checking for users on named licenses

The following use case illustrates how it is not necessary to attach a license to every object that you want protected. Instead, you can use the ACL rule condition **User In Named ITAR License** and the object's classification together to control access. **User In Named ITAR License** checks to see if a user is listed on a specified license. It does not check if the license is attached to the object.

> Note:
>
> There are also corresponding rule conditions for all types of licenses, so these concepts pertain to all licenses:

- **User In Named IP License**
- **User In Named Exclude License**
- **User In Named License**

Use case 1 – Restricting access based on named licenses

ABC Company builds the following Access Manager rules, allowing the **World** to have read access:

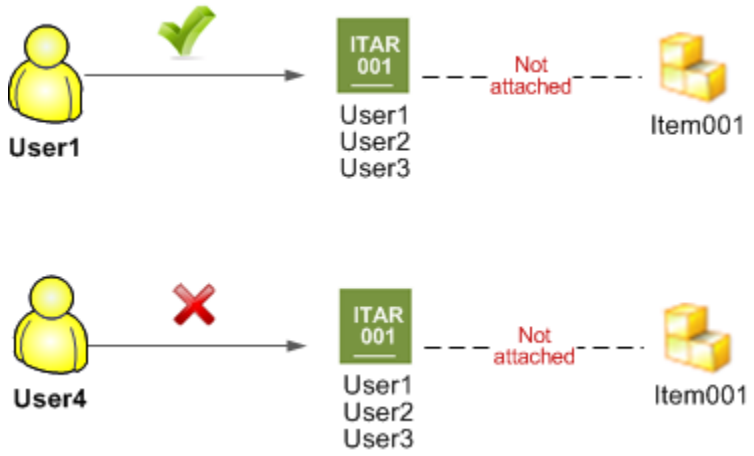
Has GovClassification = Secret

User In Named ITAR License (ITAR 001)

World –> Read

The **ITAR 001** license has three users named on it (**User1**, **User2**, and **User3**). In addition, the item trying to be accessed, **Item001**, has a **gov_classification** set to **secret**

Using the **User In Named ITAR license** rule condition, **User1** can read **Item001** because **User1** is listed on the license, while **User4** cannot read **Item001** because **User4** is not listed on the license.



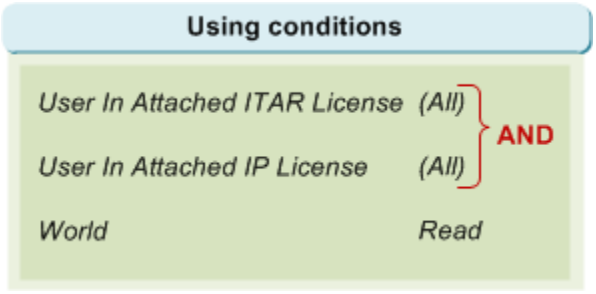
Use case — Checking for users on a given type of license

The following use cases illustrate how to check that a user is allowed access to an object if the user is named on any or all of the specified ADA licenses that are attached to the object. It uses the following rule conditions:

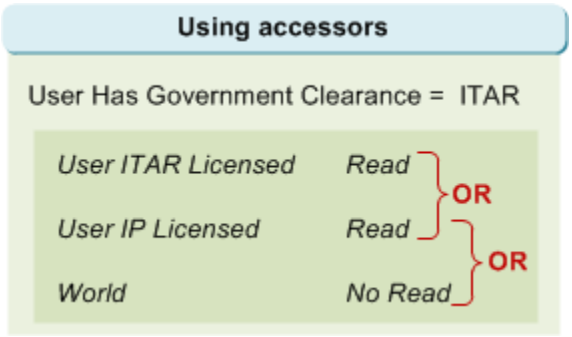
- **User In Attached ITAR License**
- **User In Attached IP License**

- **User In Attached Exclude License**
- **User In Attached License**

Because these are rule conditions, you can AND them together to build a check that determines whether the user is on all ITAR and IP licenses before being given access.



You could use accessors on the ACL, but they can only be OR'ed together. Therefore, you cannot check that the user is on both an ITAR and IP license.



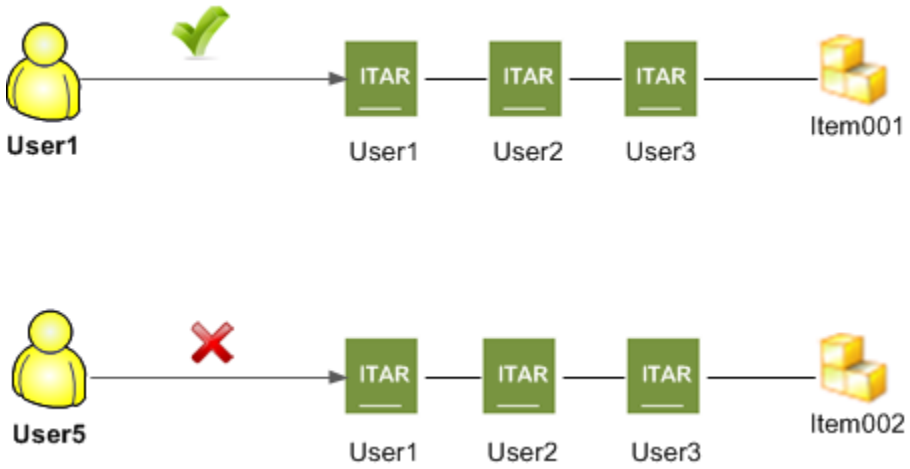
Use case 1 – User can be on any license

ABC Company builds the following Access Manager rule, which states that a user only needs to be on one or more of the ITAR licenses attached to an object to be given access to that object, with **World** having read access:

User In Attached ITAR License (Any)

World –> Read

User1 is listed on one of the licenses attached to **Item001**, as shown next. Therefore, **User1** is allowed access to **Item001**. **User5**, on the other hand, is not listed on any of the ITAR licenses attached to **item002** so **User5** is not given access to **Item002**.

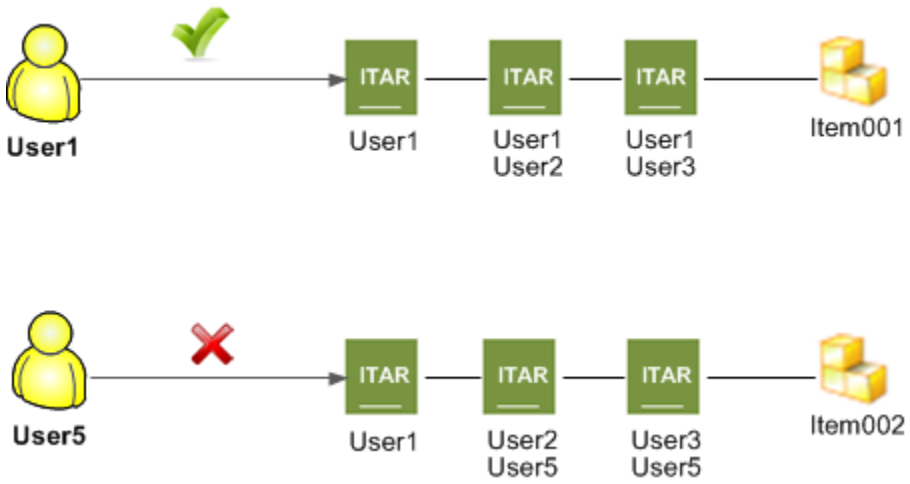


Use case 2 – User must be on all licenses

ABC Company builds the following Access Manager rule, which states that a user must be on all ITAR licenses attached to an object to be given access to that object, with **World** having read access.

- User In Attached ITAR License (All)
  - World –> Read

In this use case, **User1** is listed on all licenses attached to **Item001**, and, therefore, **User1** is allowed access to **Item001**. **User5** is denied access to **Item002** because he is only listed on two of the three ITAR licenses attached to **Item002**.



Use case — Checking for attached license by name

The following use case illustrates how you can configure access to check if an object has a license of a specific name.

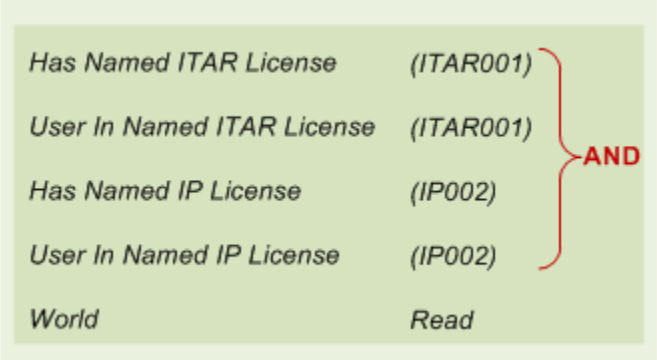
The following conditions would be used to set this type of checking:

- **Has Named ITAR License**
- **Has Named IP License**
- **Has Named Exclude License**
- **Has Named ADA License**

> Note:
>
> These conditions do not check if the user is on the license.

Using these conditions, you can configure access to check if the object has a specific ITAR license AND the user is also named on that license.

Similarly, you can check for these conditions for both ITAR and IP (AND'd together):

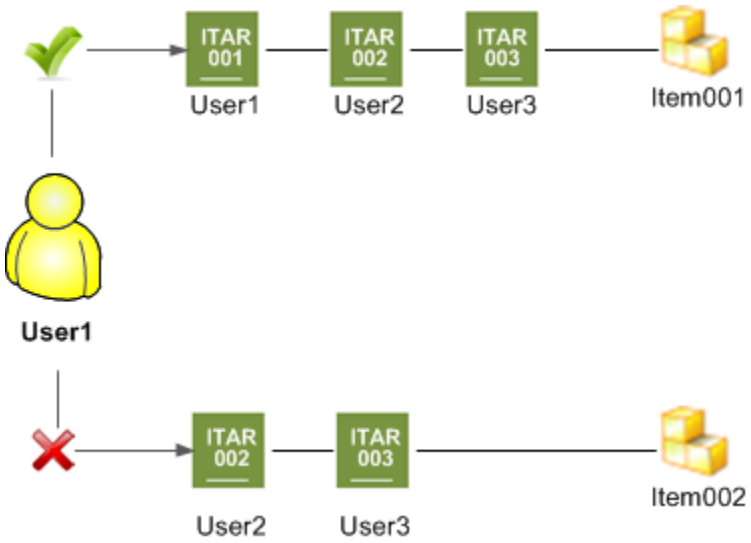


Use case 1 – Checking for attached license by name

ABC Company builds the following Access Manager rule, which states that a user is allowed access to a workspace object if there is an ITAR license by the name **ITAR001** attached to that object, with the **World** having read access:

- Has Named ITAR License (ITAR001)
  - World –> Read

User1 is allowed access because there is an ITAR license **ITAR001** attached to **Item001**, as shown next. However, **User1** is not allowed access to **Item002** because **ITAR001** license is not attached to it.



Use case — Checking user citizenships on attached licenses

The following use case illustrates how you can configure access to check if a user's citizenships are on the license's citizenship list.

The following conditions would be used to set this type of checking:

- Citizenship On Any ADA Lic
- Citizenship On Any ITAR Lic
- Citizenship On Any IP Lic

Because these are rule conditions, you can AND them together to build a check that determines whether the user citizenship is on all ITAR and IP licenses before being given access.

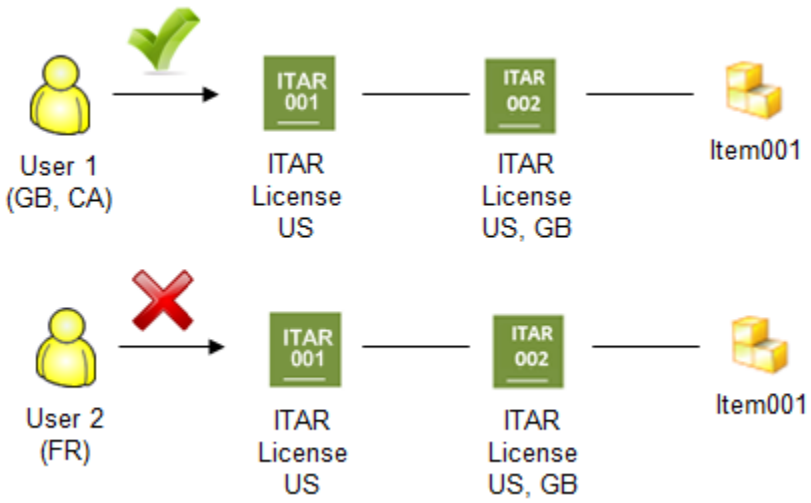
Use case 1 — Any of a user's citizenships are on the license citizenship list

ABC Company builds the following Access Manager rule, which states that only users with US or Great Britain citizenships are allowed to access certain data, with **World** having read access:

Citizenship On Any ITAR Lic ITAR (Any)

World –> Read

User1 has citizenships GB and CA and **User2** has a citizenship of FR. Two ITAR licenses are attached to **Item001**, with US listed in **ITAR001**, and US and GB listed in **ITAR002**, as shown. Therefore, **User1** is allowed access to **Item001**, but **User2**, whose citizenship FR is not listed on any of the ITAR licenses attached to **Item001**, is not given access.



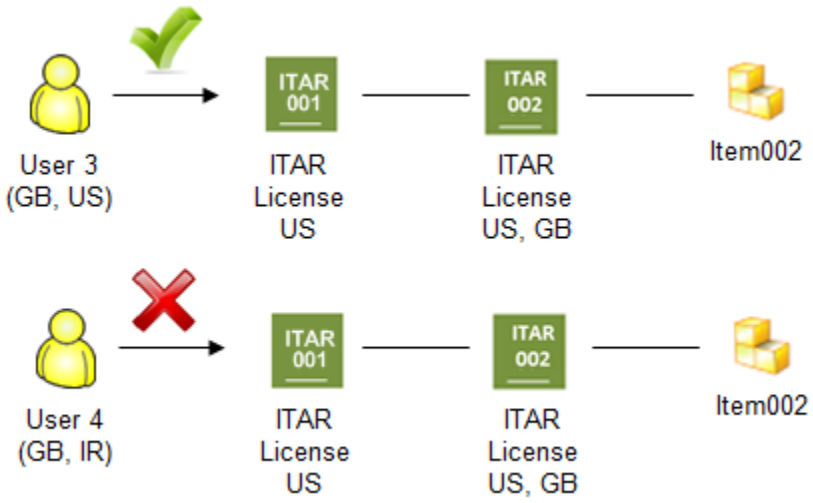
User case 2 – All citizenships of the user must be on a citizenships list of license

ABC Company builds the following Access Manager rule, which states that all user's citizenships must be on the citizenship list of ITAR licenses attached to an object to be given access to that object, with World having read access.

Citizenship On Any ITAR Lic ITAR (Any)

World –> Read

In this use case, **User3** citizenships (GB and US) are all listed on the citizenship list of the ITAR licenses attached to **Item002**. Therefore, **User3** is allowed access to **Item002**. **User4** is denied access to **Item002** because one of his two citizenships (IR) is not listed on any of the citizenship lists of the ITAR licenses attached to **Item002** even through his citizenship GB is listed.



Propagating ADA licenses

Authorized data access (ADA) licenses can be propagated to a new object when the object is created using the **Save As** command. License propagation is determined by relation propagation rules and the values specified for the preference.

In addition, you can propagate ADA licenses using propagation rules. These rules provide a codeless configuration way to copy security data, such as ADA licenses, by using the **Propagation Rules** editor in the Business Modeler IDE.

Scenario for implementing ADA for government classified data

In this scenario, Access Manager rules, government classification, user nationality, geographic location, and ITAR licenses are used to control access privileges to data.

In this example, the ABC Part Company is implementing rules to control privileges to classified and nonclassified data. To do this, they have created a single ITAR license, **license 1**, which grants access to Jane Davis. They have also assigned nationality, geographic locations, and ITAR admin privileges to users, as follows.

User	Nationality	Geography	ITAR Admin
Jane Davis	**GB**	**UK**	No
Robert Smith	**US**	**US**	No
Peter Taylor	**US**	**US**	Yes

Dataset ID	Government classification	License
ABC0001	None	None
ABC0002	**usml=XI(a)(3),eccn=2B991**	None
ABC0003	**usml=XI(a)(3),eccn=2B991**	**license 1**

ABC Part Company has also created three access control lists (ACLs): **ITARAdminACL**, **ITARACL**, **NoITARACL**, and **NoAccess**.

The ACLs are defined, as follows:

- **ITARAdminACL** ACL

Role	ITAR Admin	✓
World		✗

- **ITARACL** ACL

User ITAR Licensed		✓							
Role in Owning Group	ITAR Admin							✓	✓
World								✗	✗

- **NoITARACL** ACL

Role in Owning Group	ITAR Admin							✓	✓
World								✓	✗

- **LimitedAccess** ACL

Owning User		✓							
User ITAR Licensed		✓							
User Over Government Clearance		✓							
World		✗							

The **ITARAdminACL**, **ITARACL**, **NoITARACL**, and **LimitedAccess** ACLs are used in the following rules:

- Has Class (POM_object)
  - Has Bypass (true) –> Bypass
  - Has Government Classification() –> ITARACL
    - User Nationality("–us") –> LimitedAccess
    - User Geography("–us") –> LimitedAccess
  - Has No Government Classification() –> NoITARACL
  - Has Class(User) –> ITARAdminACL
  - Has Class(POM_object) –> System objects

In addition to these rules, ABC Part Company can set the **TC_session_clearance** preference to provide one of three levels of monitoring: **unset, blocking, or logging**. The **Unmanage** privilege can be granted to allow users to circumvent the blocking, if applicable.

Session Clearance	User	Dataset	Classification	License	Read	ITAR Logged	User can unmanage	Unmanage
block	davisj	ABC0001	none	none	Y	N	Y	Y
block	smithr	ABC0001	none	none	Y	N	Y	Y
block	taylorp	ABC0001	none	none	Y	N	Y	Y
block	davisj	ABC0002	usml=XI(a)(3) eccn=2B991	none	N	N/A	N/A	N/A
block	smithr	ABC0002	usml=XI(a)(3) eccn=2B991	none	Y	Y	N	N
block	taylorp	ABC0002	usml=XI(a)(3) eccn=2B991	none	Y	Y	Y	Y

Session Clearance	User	Dataset	Classification	License	Read	ITAR Logged	User can unmanage	Unmanage
block	davisj	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	Y	N	N
block	smithr	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	Y	N	N
block	taylorp	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	Y	Y	Y
log	davisj	ABC0001	none	none	Y	N	Y	Y
log	smithr	ABC0001	none	none	Y	N	Y	Y
log	taylorp	ABC0001	none	none	Y	N	Y	Y
log	davisj	ABC0002	usml=XI(a)(3) eccn=2B991	none	N	N/A	N/A	N/A
log	smithr	ABC0002	usml=XI(a)(3) eccn=2B991	none	Y	Y	Y	N
log	taylorp	ABC0002	usml=XI(a)(3) eccn=2B991	none	Y	Y	Y	Y
log	davisj	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	Y	Y	N
log	smithr	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	Y	Y	N
log	taylorp	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	Y	Y	Y
unset	davisj	ABC0001	none	none	Y	N	Y	Y
unset	smithr	ABC0001	none	none	Y	N	Y	Y
unset	taylorp	ABC0001	none	none	Y	N	Y	Y
unset	davisj	ABC0002	usml=XI(a)(3) eccn=2B991	none	N	N/A	N/A	N/A
unset	smithr	ABC0002	usml=XI(a)(3) eccn=2B991	none	Y	N	Y	N
unset	taylorp	ABC0002	usml=XI(a)(3) eccn=2B991	none	Y	N	Y	Y
unset	davisj	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	N	Y	N
unset	smithr	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	N	Y	N
unset	taylorp	ABC0003	usml=XI(a)(3) eccn=2B991	license 1	Y	N	Y	Y

ITAR implementation considerations

Access Manager considerations for ITAR

Placement of rules in the Access Manager rule tree for ITAR

Access Manager implicitly grants access privileges to data unless privileges are explicitly revoked; therefore, rules to identify users with ITAR clearance or license should be placed high in the rule tree. If ITAR-related conditions do not apply, the rule tree is evaluated and data access privileges are determined by other factors, such as project membership or object ownership, as the rule tree is evaluated.

Authorized data access rule precedence for ITAR and IP

When both ITAR and IP access rules are in force at a site, ITAR rules must take precedence over IP rules. Therefore, you must place ITAR rules higher in the rule tree than IP rules.

NX security for classified data for ITAR and IP

When operating in Teamcenter Integration for NX mode, Teamcenter provides security metadata to NX. This metadata is computed using the user and data properties and directs how NX behavior is modified when processing secure data. Security metadata is constant in NX, even if the information changes in Teamcenter. For example, if a user is granted access to data in NX based on a license, and the license expires while the data is open in NX, the expiration is not recognized by NX until the user attempts to save the data back to Teamcenter, at which point the operation fails because the user's license has expired.

Authorized data access logging and blocking supports NX datasets as well as non-NX datasets that can be translated into a part file, including JT, Solid Edge, and CATIA data.

The following NX operations are affected when logging and blocking are implemented.

Operation	Blocked and logged	Blocked
File→Save As	X	
File→Print	X	
File→Plot	X	
File→Send to Package File	X	
File→Export→Part	X	
File→Export→Parasolid	X	
File→Export→User Defined Feature	X	
File→Export→CGM	X	

Operation	Blocked and logged	Blocked
File→Export→STL	X	
File→Export→Polygon File	X	
File→Export→Author HTML	X	
File→Export→Teamcenter Visualization	X	
File→Export→VRML	X	
File→Export→PNG	X	
File→Export→JPEG	X	
File→Export→GIF	X	
File→Export→TIFF	X	
File→Export→BMP	X	
File→Export→XWD	X	
File→Export→IGES	X	
File→Export→STEP203	X	
File→Export→STEP214	X	
File→Export→DXF/DWG	X	
File→Export→2D exchange	X	
File→Export→Heal Geometry	X	
File→Export→V4 Catia	X	
File→Export→V5 Catia	X	
File→Export drawings to Teamcenter	X	
File→Interoperate	X	
File→Collaborate	X	
Edit→Copy Display	X	
View→Visualization→High Quality Image	X	
View→Visualization→Create Animation	X	
Tools→NX Manager→Export Assembly	X	
Tools→NX Manager→Save Outside Teamcenter	X	

Operation	Blocked and logged	Blocked
Tools→Part Families Create		X
Tools→Part Family Update		X
Assemblies→Components→Create New	X	
Assemblies→Components→Add Existing		X
Assemblies→Components→Substitute Component		X
Assemblies→Components→Part Family Update		X
Assemblies→Components→Create New Parent		X
Assemblies→Cloning→Create Clone Assembly	X	
Edit→Paste [Pasting of a component]		X
Dragging a drawing template onto a part		X

Multi-Site Collaboration considerations for ITAR

- Authorized data access (ADA) licenses are considered to be local administrative data and are maintained by individual sites.
- Take care when sharing classified data with other Teamcenter sites. It is possible that when importing a mixture of classified and unclassified data, the user who is performing the import could view data to which they do not have clearance. Teamcenter tests for privileges when importing data, but the system does not read the classification attribute until the object has been reassembled and saved at the importing site.
- Licenses are not automatically exported with data objects; however, references to the licenses are exported with the data object.
- The **TC_multi_site_ada_license_user_bypass** preference can be set to allow licenses to be attached to replicated objects upon import and to allow licenses attached to replicated objects to be imported, even when the importing user does not have the privileges required to attach such a license.
- To ensure that classification and license attributes are retained when classified data is imported to a remote site, you should first verify that the clearance and classification values and AM rule tree entries are synchronized between the sites.

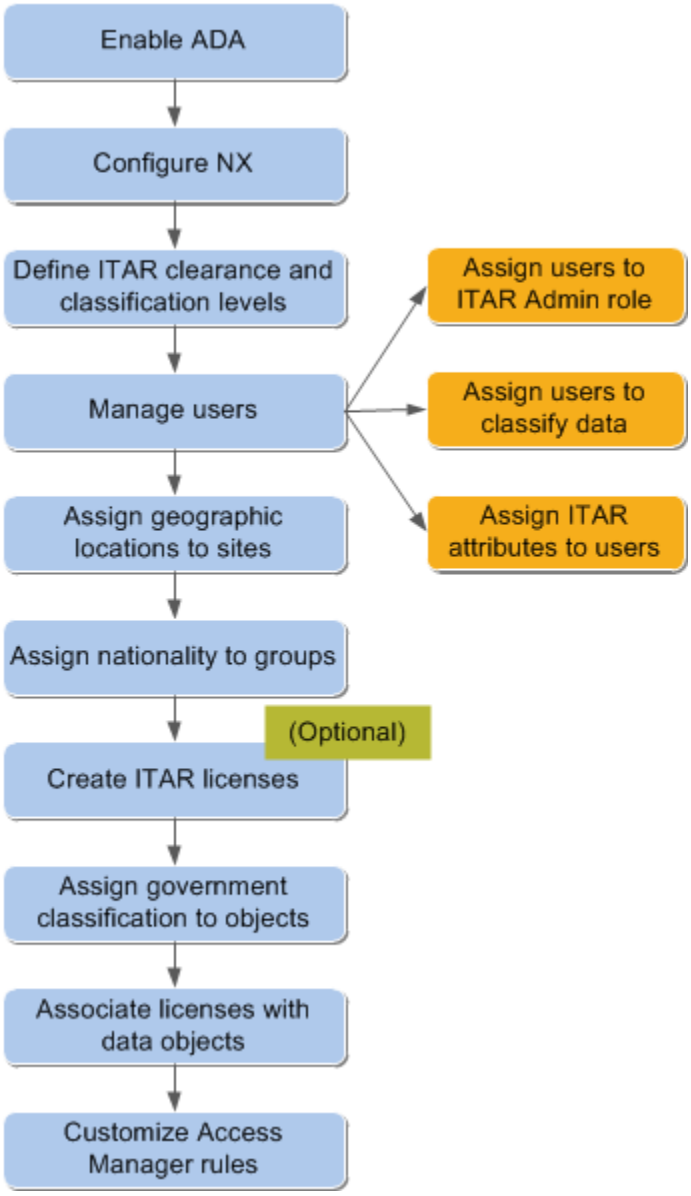
- The **Geography** property on site definitions can be used as the basis for access rules to control Multi-Site Collaboration privileges for exporting and transferring data.

- The **ADA_override_on_import** preference can be set to keep IP and government classification values on a workspace object in sync between the master and replica sites.

Basic tasks for configuring and administering authorized data access for ITAR restricted data

About the tasks for configuring and administering authorized data access for ITAR restricted data

The following basic tasks must be performed when configuring and administering authorized data access for intellectual property data:



Enabling authorized data access

Authorized data access (ADA) features are enabled by default when you install Teamcenter. To disable ADA, set the **ADA_enabled** preference to **false**.

Configuring logging and blocking in NX

The **TC_session_clearance** preference, in conjunction with NX runtime properties, enables you to establish indirect access controls in NX using logging or menu suppression (blocking) to control classified data that is loaded in a Teamcenter Integration for NX session.

Logging provides auditable evidence of the use of various NX commands on classified data. This is implemented by NX internal mechanisms that are beyond the scope of this document. For more information, see the NX Help Library.

Blocking suppresses NX menus that, if used on classified data, could result in exporting geometric data outside the NX/Teamcenter managed environment. The blocking feature also provides menu action logging.

While allowing data out of the managed environment creates a security vulnerability, you may at times want to grant a user permission to use NX menus that involve exporting data out of the environment. You can use the **Unmanage** privilege in Access Manager to grant users the ability to access these restricted features.

Configuring logging and blocking is optional.

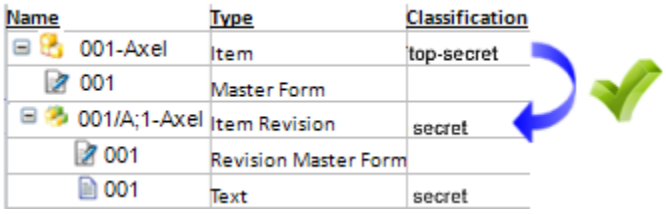
Defining ITAR clearance and classification levels

Each site can define a list of International Traffic in Arms Regulations (ITAR) (government) classification values and clearance levels that are assigned to data objects and users for ITAR access evaluation. The list is maintained in the **ITAR_level_list_ordering** preference. For example, you define a list of the **Secret**, **Confidential**, and **Top Secret** access levels. The order in which the levels appear in **ITAR_level_list_ordering** defines their restrictiveness. The first entry is the lowest classification, and the last entry is the highest. Access Manager compares these values to determine user access rights to an object.

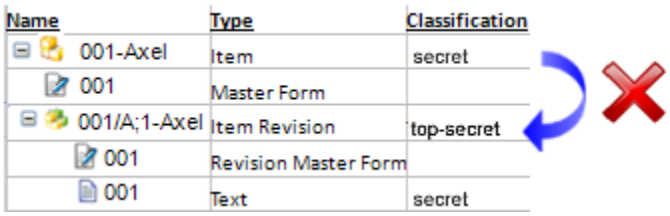
> Note:
>
> The examples in this guide do not use the classification attribute-clearance level comparison to implement authorized data access. In the examples, the classification attribute is used to provide descriptive details about the object.

The government classification values are propagated to related objects according to the propagation rules defined in Teamcenter (when setting the value on an item, all revisions and their attached datasets are assigned the same value). This only applies, however, when setting a value that is *more restrictive* than the current value. If you set a less restrictive value on an object, the value is not propagated to the related objects.

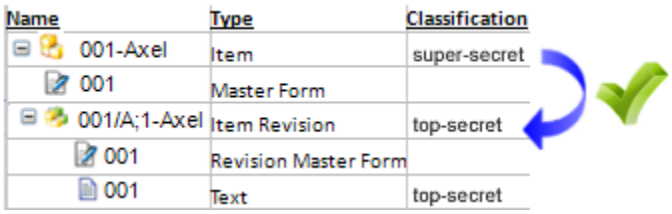
For example, if you set a classification value of **Top Secret** on the **001–Axel** part, it is propagated to the related objects because its classification is more restrictive than the government classification of the related objects (**Secret**):



In the following example, however, if you set a government classification value of **Secret** on the **001–Axel** part, it is *not* propagated from **001–Axel** to the related objects because it is less restrictive (at a lower level) than the government classification level set on the related objects (**Top Secret**):



You can replace a government classification level with another of equal value. For example, in the following example, **Confidential** is replaced with **Secret** because it is at the same level.



Note:

Classification values that are of equal value are separated by commas in the **ITAR_level_list_ordering** preference.

Assigning users to the ITAR Admin role and grant ITAR Admin privileges

The specific privilege required to administer licenses is defined by the value specified for the **ADA_license_administration_privilege** site preference (**ITAR_ADMIN** by default). The **ITAR Admin** role and the **ITAR Admin** privilege authorize users to classify data objects and administer ITAR licenses.

License modification and deletion requires the user have the privilege defined by the **ADA_license_administration_privilege** preference, either explicitly or in the context of a group or project. Adding and removing users from licenses and setting license expiration dates requires the same privilege, typically by being either the owning user or being assigned to the **ITAR Admin** role.

Applying licenses to or removing licenses from objects requires that the user have privileges specified in the **ADA_license_administration_privilege** site preference on both the license and the workspace object to which the license is being applied.

Users can be assigned to the **ITAR Admin** role by a Teamcenter administrator using the Organization application.

Note:

License creation is not controlled by Access Manager rules.

Assigning users to classify data

Use the Organization application to assign users to the **ITAR Classifier** role to allow them to classify ITAR information without granting all of the privileges of the **ITAR_ADMIN**. Users of these roles must be granted the **ITAR_Classifier** privilege.

Overview of configuring ADA access.

Comparison of roles

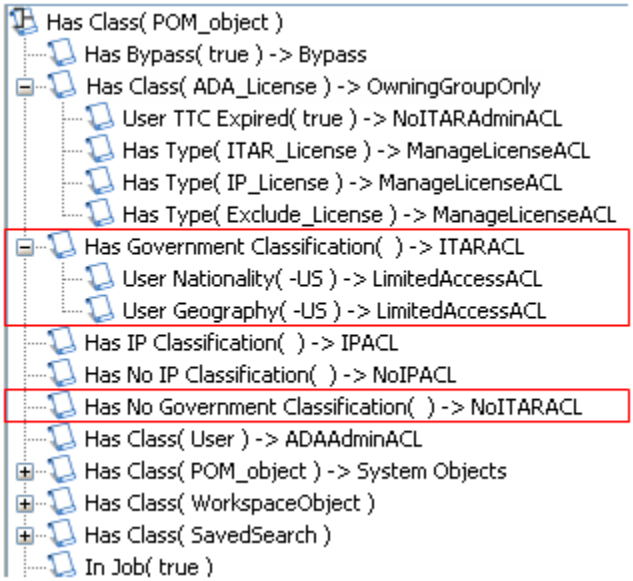
The following table provides a high level comparison of the actions that users with **ITAR_ADMIN**, **IP_ADMIN**, **ITAR_Classifier**, and **IP_Classifier** privileges can perform (there are minor exceptions for attaching or detaching licenses):

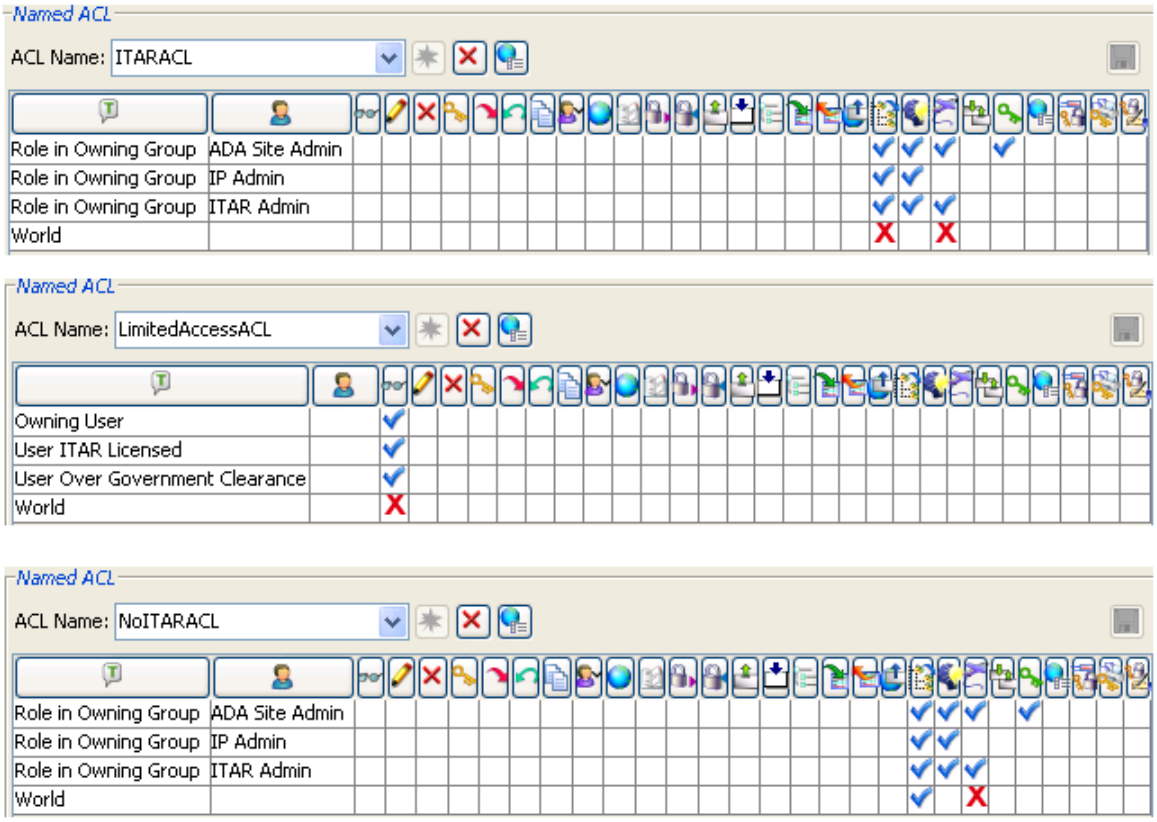
Action	ITAR_ADMIN	IP_ADMIN	ITAR_Classifier	IP_Classifier
Create ITAR, IP, or exclude license	Yes, if preference **ADA_license_ administration_ privilege** is set to **ITAR_ADMIN**	Yes, if preference **ADA_license_ administration_ privilege** is set to **IP_ADMIN**	No	No
Modify ITAR, IP, or exclude license	Yes, if preference **ADA_license_ administration_ privilege** is set to **ITAR_ADMIN**	Yes, if preference **ADA_license_ administration_ privilege** is set to **IP_ADMIN**	No	No
Delete ITAR, IP, or exclude license	Yes, if preference **ADA_license_ administration_ privilege** is set to **ITAR_ADMIN**	Yes, if preference **ADA_license_ administration_ privilege** is set to **IP_ADMIN**	No	No
Attach ITAR license to a workspace object	Yes	No	No	No
Attach IP or exclude license of workspace object	No	Yes	No	No
Detach ITAR license from workspace object	Yes, if preference **ADA_license_ administration_**	No	No	No

Action	ITAR_ADMIN	IP_ADMIN	ITAR_Classifier	IP_Classifier
	privilege is set to **ITAR_ADMIN**			
Detach IP or exclude license from workspace object	No	Yes, if preference **ADA_license_ administration_ privilege** is set to **IP_ADMIN**	No	No
Set or modify government classification on workspace object	Yes	No	Yes	No
Set or modify IP classification on workspace object	No	Yes	No	Yes
Set or modify IP clearance for a user	No	Yes	No	No
Set or modify the following values for a user: • Government clearance • TTC Date • Nationality • Geography	Yes	No	No	No

Recommended rules for international traffic in arms (ITAR) classification of workspace objects

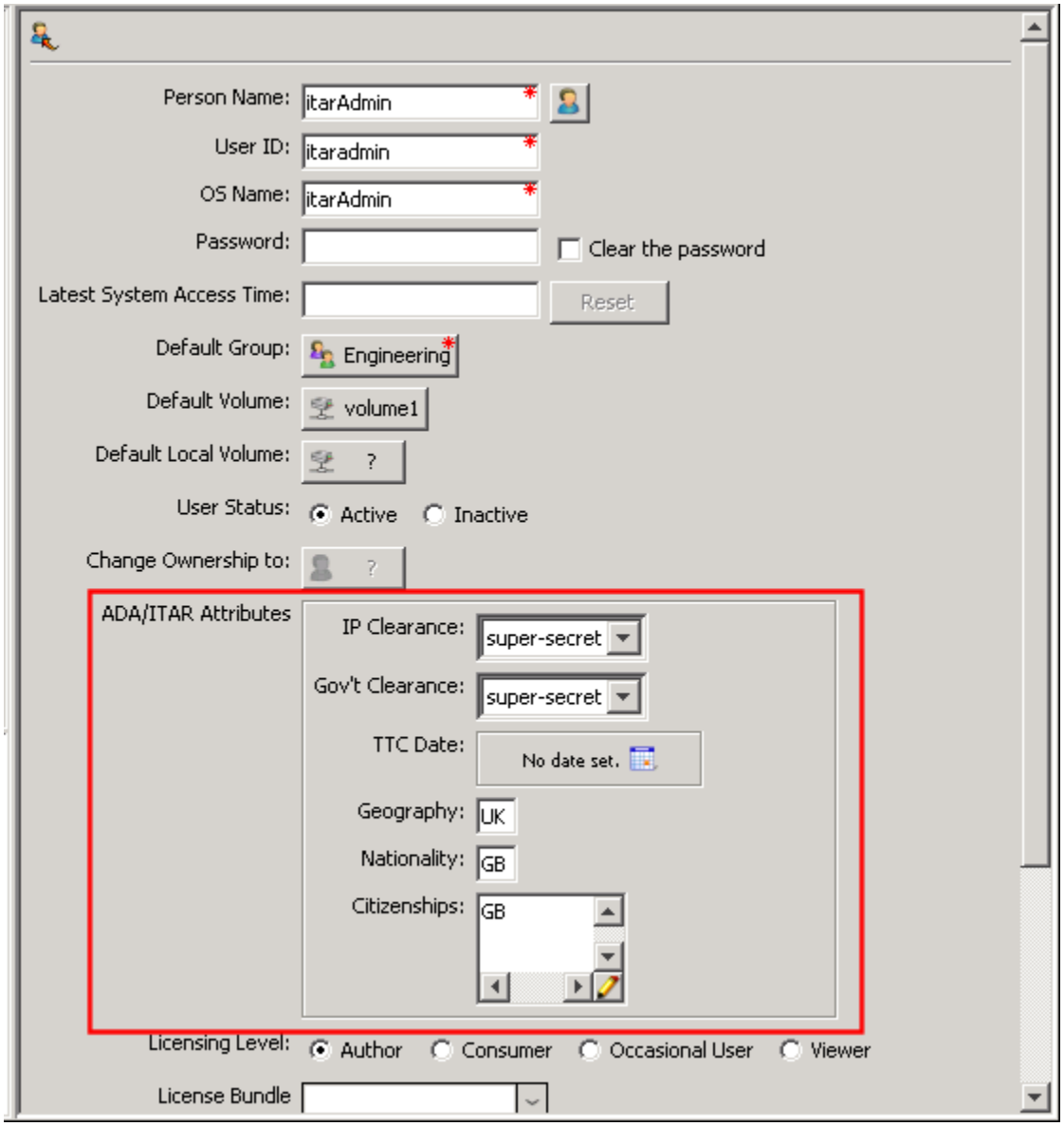
The following are examples of recommended rules and access control lists (ACLs) for users performing ITAR classification of workspace objects.





Assigning ADA ITAR attributes to users

Authorized data access uses the **government clearance**, **geography**, and **nationality**, citizenship, and **technology transfer certification date** attributes associated with a user to evaluate access rules to determine access privileges. These attributes are set in the Organization application.



Assigning geographic locations to sites

Assigning a geographic location to a site enables you to write access rules to control Multi-Site Collaboration privileges for exporting and transferring data. The **geography** attribute is set in the Organization application.

Assigning nationality to groups

You can set the nationality attribute on Teamcenter groups, which enables you to use the group as the basis for selectively granting users access to restricted data based on ITAR licensing. Groups are created and maintained using the Organization application.

Note:

Teamcenter groups are subject to hierarchical group behavior. Therefore, subgroups inherit the properties of the parent group. Siemens Digital Industries Software recommends that users be made members of the most specific group possible to ensure the accuracy of key attributes, such as the nationality of the organization.

Creating ITAR licenses (optional)

ITAR licenses provide discretionary (time limited) grants or denials of access to a user or users who do not have access to classified data based on their clearance level. ITAR licenses are the authorizing documents in Teamcenter that represent an effective Technical Assistance Agreement (TAA). The ITAR license can have one or more citizenships associated with it to restrict access.

Users who have the privilege specified in the **ADA_license_administration_privilege** preference create and maintain licenses in the ADA License application. Once created, privileges are either granted or denied to users by associating the license directly with the data object.

Assigning government classification values to data objects

Access to classified data is determined by an evaluation of the user's clearance level and the classification level that is applied to the object. The **Gov Classification** property specifies the classification level of an individual object and can be viewed and modified in the same manner as other object properties. Users must have **ITAR Admin** or **ITAR Classifier** privileges to classify objects in Teamcenter.

Note:

The **Classification**, **Classified**, and **Classified in** properties apply to the Teamcenter Classification application that is used to categorize objects for reuse. They do not apply to classification for the purpose of authorizing data access.

Assign a government classification value to an object

1. In the tree display in My Teamcenter, right-click the object you want to classify and choose **Properties**.

   Tip:

   You can also display and modify object properties in the **Details** pane by selecting the object and choosing **Details** from the list in the upper-right corner of the Teamcenter window. This method allows you to assign a classification value without performing the extra steps required to display the **Properties** dialog box.

2. In the **Properties** dialog box, click **Edit**.

   The system checks the object out of the database and opens it for edit.

3. Click the **All** link in the blue bar at the bottom of the dialog box to display all of the object properties.

4. Scroll to the bottom of the dialog box, and click the **More** link.

5. Scroll to locate the **Gov Classification** property.

6. Type the Gov classification value in the box.

7. Click **Save and Check In**.

Associating licenses with data objects

ITAR licenses grant named users discretionary, limited-time access to classified data and are created and maintained by users who assigned to the **ITAR Admin** role.

Note:

- To attach or detach International Traffic in Arms Regulations (ITAR) licenses, you must have the **ITAR_Admin** privilege for both the workspace object and the license.
- Additionally, to detach ITAR licenses, you must have the privilege specified in the **ADA_license_administration_privilege** preference (**IP_Admin**, **ITAR_Admin**, or **Administer_ADA_Licenses**) for the workplace object.
- You may want to set the **ADA_enable_subgroups** preference to specify whether or not subgroup members of the top-level groups can be authorized access to workspace objects with attached licenses when the top-level group is not authorized access.

Attach ITAR licenses to a data object

1. In the My Teamcenter tree pane, right-click the object to which you want to attach the license.

2. Choose **License→Attach** from the shortcut menu.

3. In the **Assign an Object to Licenses** dialog box, select the license that you want to attach to the object.

4. Click **OK**.

Customizing Access Manager rules

By default, authorized data access (ADA) rules are not included in the Access Manager rule tree. You can add rules to support your company's business processes.

Managing ADA licenses

Overview of managing ADA licenses and required permissions

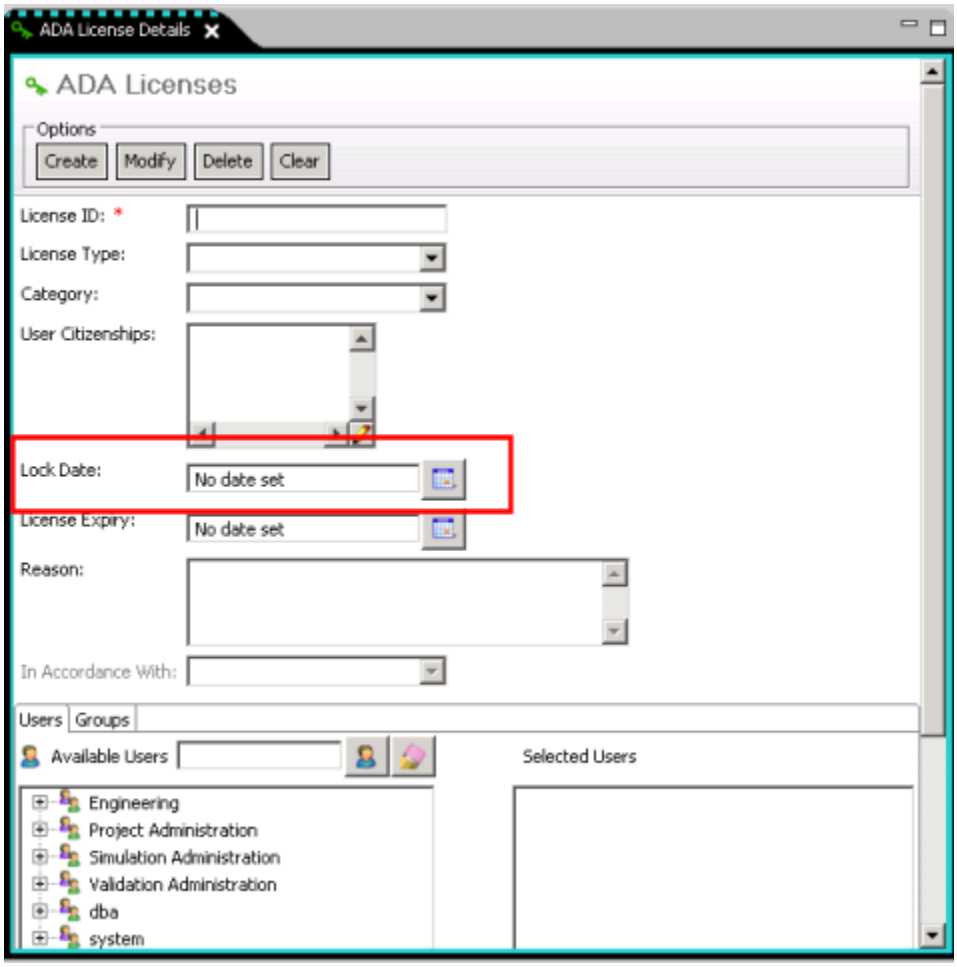
By default, you must be authorized (configured through the Authorization application) to access the ADA License application in Edit mode, and assigned the **ITAR Admin**, **IP Admin**, or **ADA Site Admin** privilege (configured through the Authorization application) to create a new ADA license. ADA licenses provide discretionary (time limited) grants or denials of access to users who do not have access to classified data based on their clearance level. ITAR licenses are the authorizing documents in Teamcenter that represent an effective Technical Assistance Agreement (TAA).

When you create or modify a license, you set:

- **License category**.
- **Applied user citizenships**.
- **Lock date**.
- **Expiry date**.
- **Users or groups** who can access an object with the license attached.

About setting a lock date

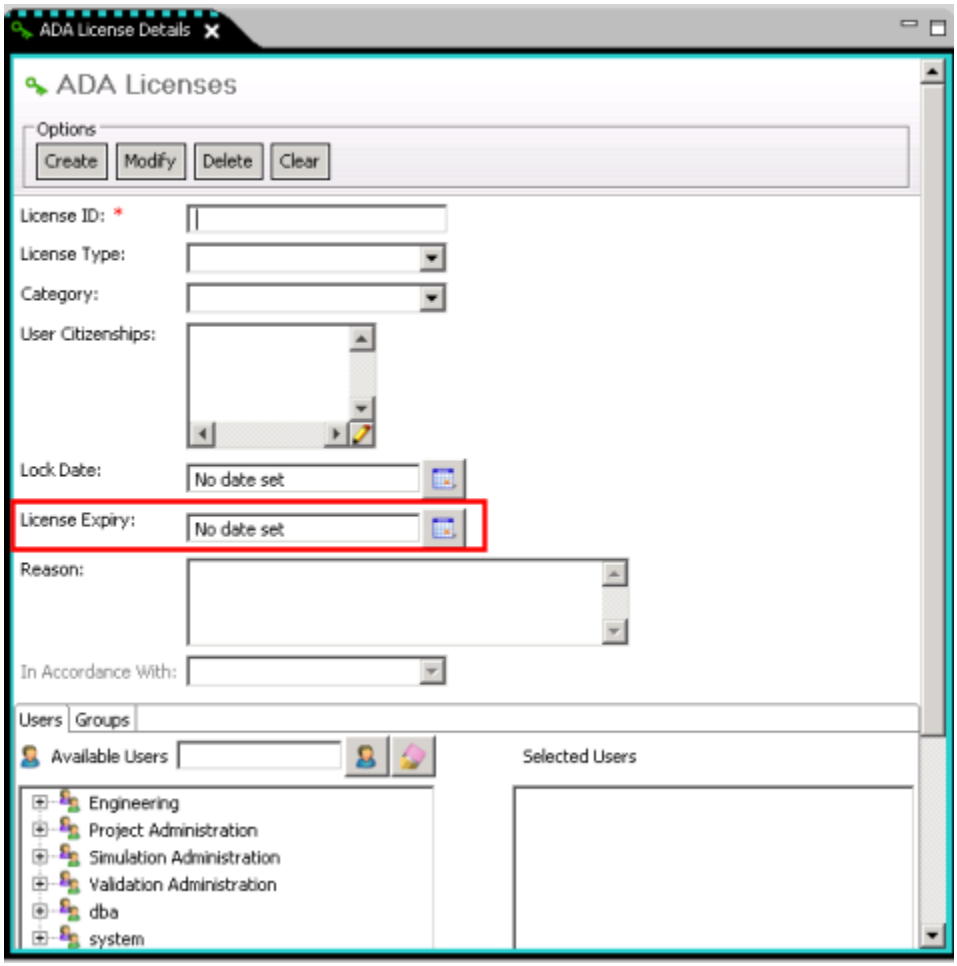
When a license is locked (reaches the lock date), Teamcenter users with appropriate privileges can query for and view the properties of the classified workspace objects associated with the license. However, the license can no longer be attached to any other workspace objects. Additional users or groups cannot be added to a locked license. However, existing users or groups specified on the license can be removed. When a license is subsequently unlocked (lock date changed to a future date or cleared), it can be attached to other workspace objects and the access can be modified.



About setting an expiry date

Once a license expires (reaches its expiration date), Teamcenter users specified on the license can no longer query for or view the classified workspace objects with the license attached. The license can no longer be attached to workspace objects.

Click **Set Date**, select a date, and if desired, a time from the **Date Control** dialog box for the **License Expiry** box, and click **OK**.



About setting user or group access

You can set the users and groups that are allowed to access the workspace objects under the license control of licenses that are not locked or have not expired.

Change the users and groups that are allowed to access objects with the licensed attached:

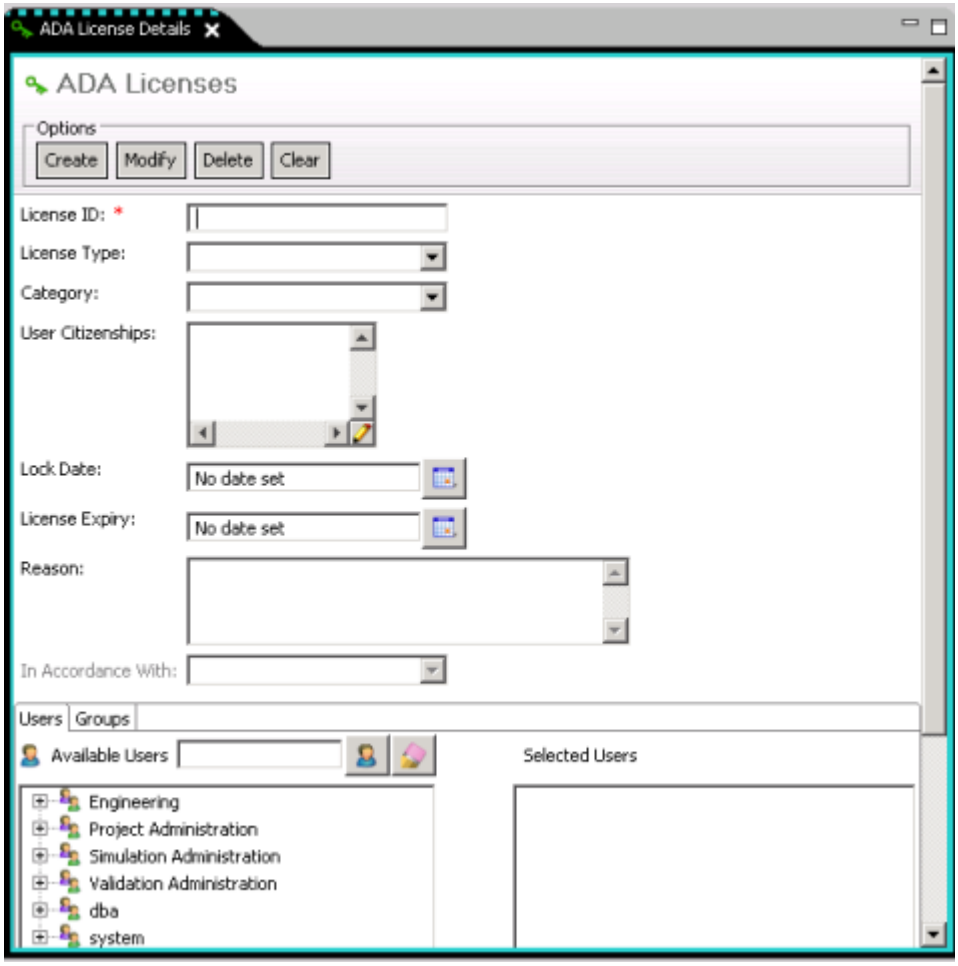
- To allow access to specific users, click the **Users** tab, select the user name in the **Available Users** list, and click ▶ to add the user to the **Selected Users** list.
- To remove access from specific users, click the **Users** tab, select the user name in the **Selected Users** list, and click ◀ to add the user to the **Available Users** list.
- To allow access to specific groups and the associated subgroups (if configured for subgroups), click the **Groups** tab, select the group name in the **Available Groups** list, and click ▶ to add the group to the **Selected Groups** list.

- To remove access to specific groups and the associated subgroups (if configured for subgroups), click the **Groups** tab, select the group name in the **Selected Groups** list, and click ◀ to add the user to the **Available Groups** list.

Create licenses

Required permissions

1. Start ADA License and select the ADA Licenses node (top node) in the left pane.



2. Type a name for the license in the **License ID** box, for example, **ITAR_USML**.
3. Select the **type of license** from the **License Type** list, for example, **ITAR_License**.
4. (Optional) Select a **category** for the license from the **Category** list, for example, select **Category A**.
5. (Optional) Click **Edit** to enter a country code and add it to the license.

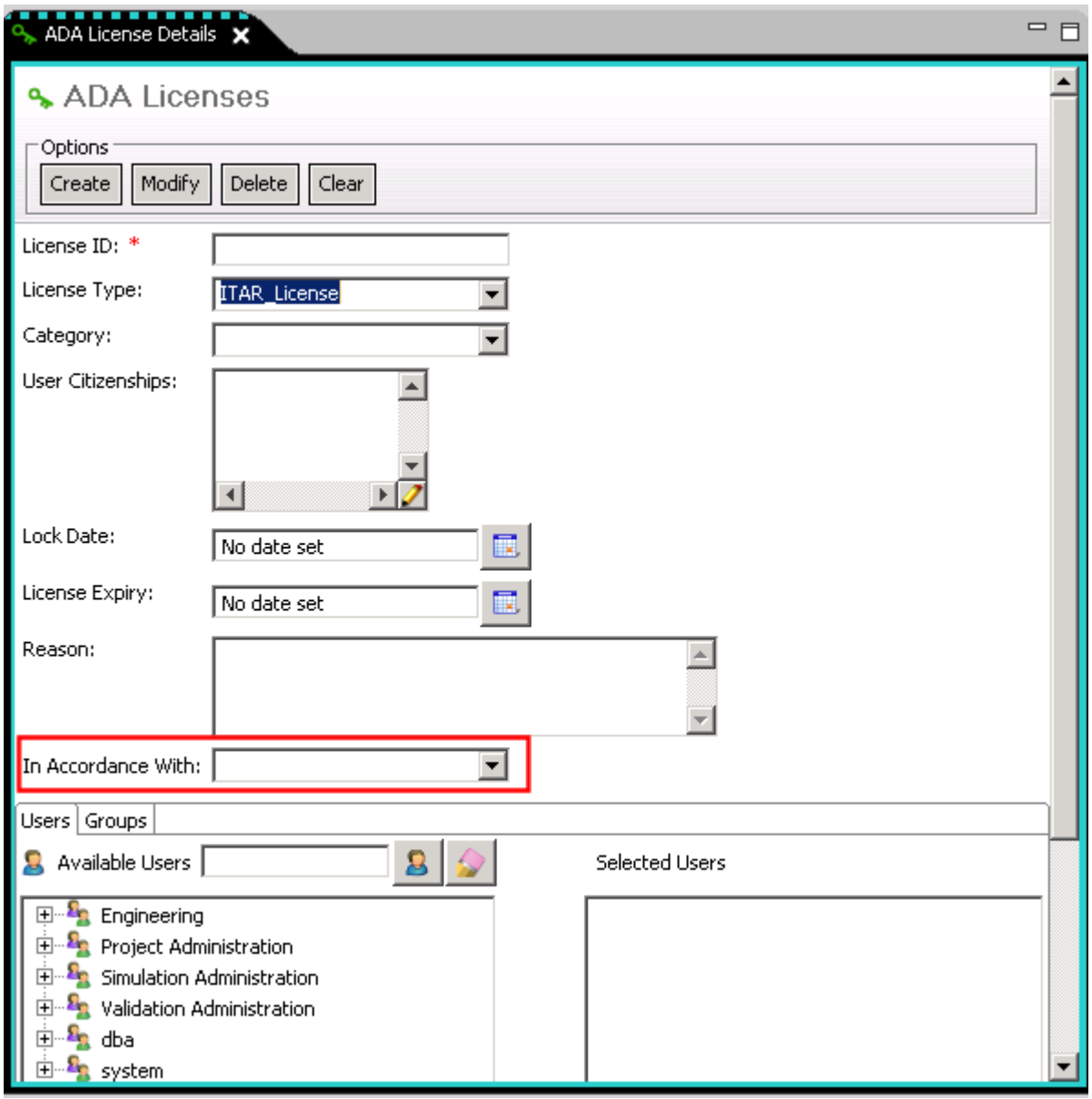
Citizenships are two-letter country code from ISO 3166. For example, **US** for the United States and **GB** for the Great Britain.

A site can define a country code list of values (LOV) and attach it to the **user_citizenships** property of ADA License. This would allow you to select a country code from the country code LOV.

6. (Optional) Using the **Lock Date** box, select a date, and if desired, a time.

7. (Optional) Next to the **License Expiry** box, select a date, and if desired, a time.

8. (Optional for ITAR licenses only) Select a qualifying Code of Federal Regulations (CFR) from the **In Accordance With** box.



A site can also define and attach a list of values (LOV) with valid values for the **In Accordance With** box using the Business Modeler IDE. This allows the user to select a value from the LOV.

9. Select the users and groups that are allowed to access objects with the license attached.

   - (Optional) To allow access to specific users, click the **Users** tab, select the user name in the **Available Users** list, and click (>) to add the user to the **Selected Users** list.

   - (Optional) To allow access to specific groups and the associated subgroups (if configured for subgroups), click the **Groups** tab, select the group name in the **Available Groups** list, and click (>) to add the user to the **Selected Groups** list.

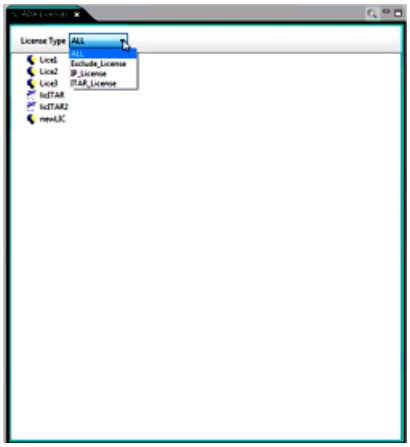
10. Click **Create**.

Modify licenses

Required permissions

> Note:
>
> You cannot change the ID of a license. If you want a different license ID, you must delete the current license and create a new one.

1. In the ADA Licenses page, from the **License Type** list, select the license type you are modifying:



2. From the **ADA Licenses** tree, select the license to modify.

3. Set or modify the desired options.

> Note:
>
> You can select a category for a license even if the license did not originally have a category defined.

4. Type a description in the **Reason** box.

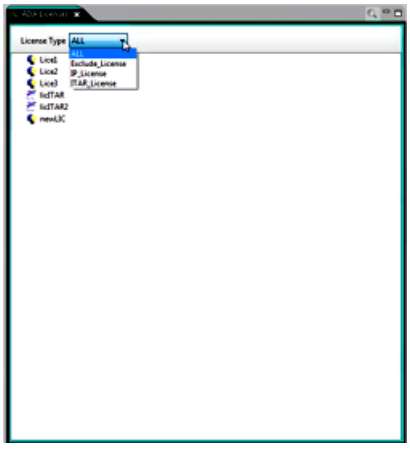
5. Click **Modify**.

Delete licenses

To delete a license, you must be assigned the **delete** privilege. You cannot delete a license that is attached to a workspace object,

1. Start ADA License.

2. From the **License Type** list, select the type of license you want to delete:



3. From the **ADA Licenses** tree, select the license to delete.

4. Click **Delete** in the **ADA License Details** pane and click **OK** in the **Confirmation** dialog box.

Managing ADA licenses on workspace objects

Overview of managing ADA licenses on workspace objects and required permissions

> Note:
>
> To attach or detach a license from a workspace object, the following privileges are needed:

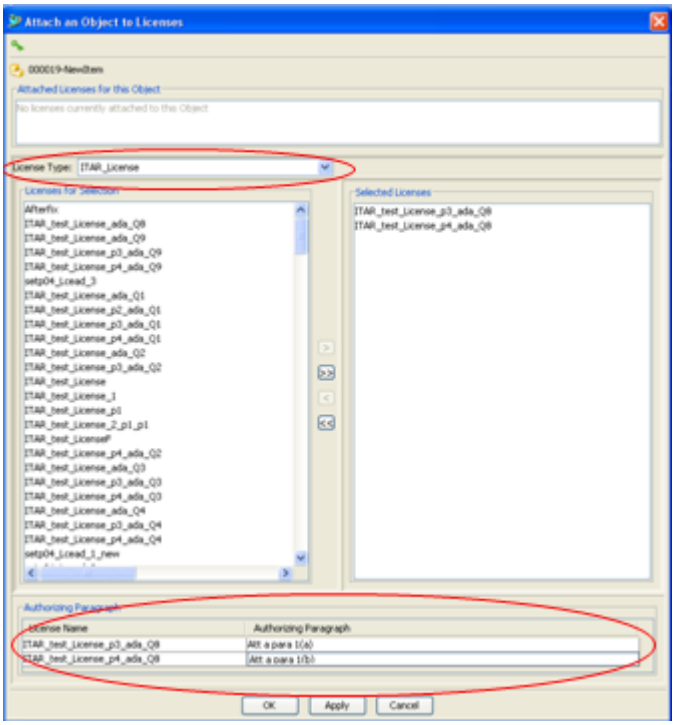
- To attach or detach intellectual property (IP) or exclude licenses, you must have the **IP_ADMIN** privilege for both the workspace object and the license.
- To attach or detach International Traffic in Arms Regulations (ITAR) licenses, you must have the **ITAR_ADMIN** privilege for both the workspace object and the license.
- Additionally, to detach, you must have the privilege specified in the **ADA_license_administration_privilege** preference (**IP_ADMIN**, **ITAR_ADMIN**, or **Administer_ADA_Licenses**) for the workspace object.

Attach licenses to workspace objects

You can attach licenses that are not locked and not expired to workspace objects. Attaching a license allows the users and groups named in the license to get read access to the workspace object. As you attach an ITAR license, you can cite the applicable authorizing paragraphs. The audit log records an entry for every license attached.

Required permissions

1. In My Teamcenter, search for the object to which you want to attach the license.
2. Right-click the object and choose **License→Attach**.
3. In the **Attach an Object to Licenses** dialog box, set **License Type** to the type of license to be attached.



> Note:
>
> Only unlocked licenses and licenses that have not reached their expiration date are listed.

4. In the **Licenses for Selection** list, select the license to be attached and click the **(>)** button.

   The selected licenses appear in the **Selected Licenses** list.

   - To move all licenses to the **Selected Licenses** list, click the **(>>)** button.
   - To remove licenses from the **Selected Licenses** list, click the **(<<)** or **(<)** button.

5. (Optional for ITAR licenses only) Type a statement in **Authorizing Paragraph** to explain the authorization.

6. Click **OK** or **Apply**.

Detach licenses from workspace objects

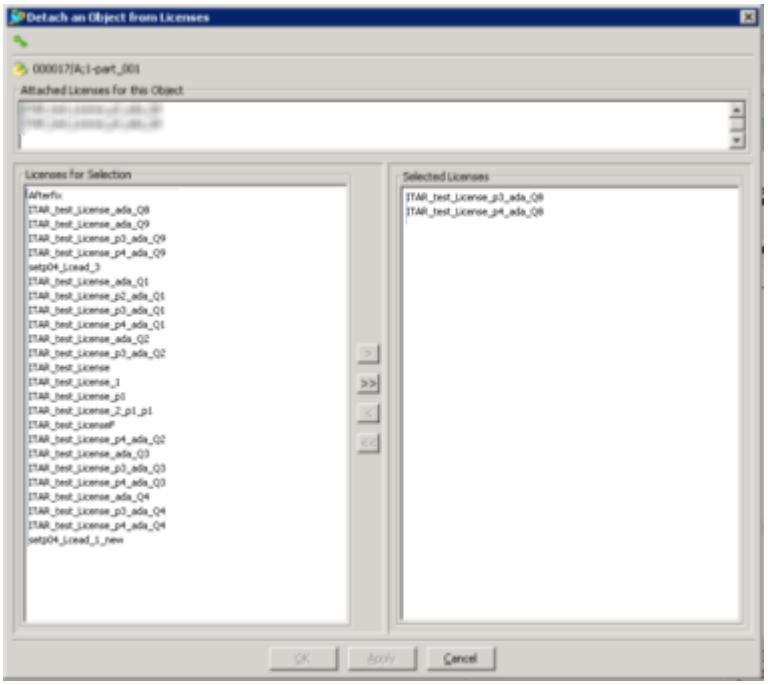
You can detach a license from a workspace object even if the license is locked or expired. When an ITAR license is detached, the associated authorizing paragraph information is deleted. The audit log records an entry for every license detached.

Required permissions

1. In My Teamcenter, search for the object from which you want to detach the license.

2. Right-click the object and choose **License→Detach**.

3. In the **Detach License** dialog box, select the license that you want to detach from the object.



4. In the **Licenses for Selection** list, select the license to be detached and click the **(>)** button.

   The selected licenses appear in the **Selected Licenses** list.

   Tip:

   - To move all licenses to the **Selected Licenses** list, click the **(>>)** button.
   - To remove licenses from the **Selected Licenses** list, click the **(<<)** or **(<)** button.

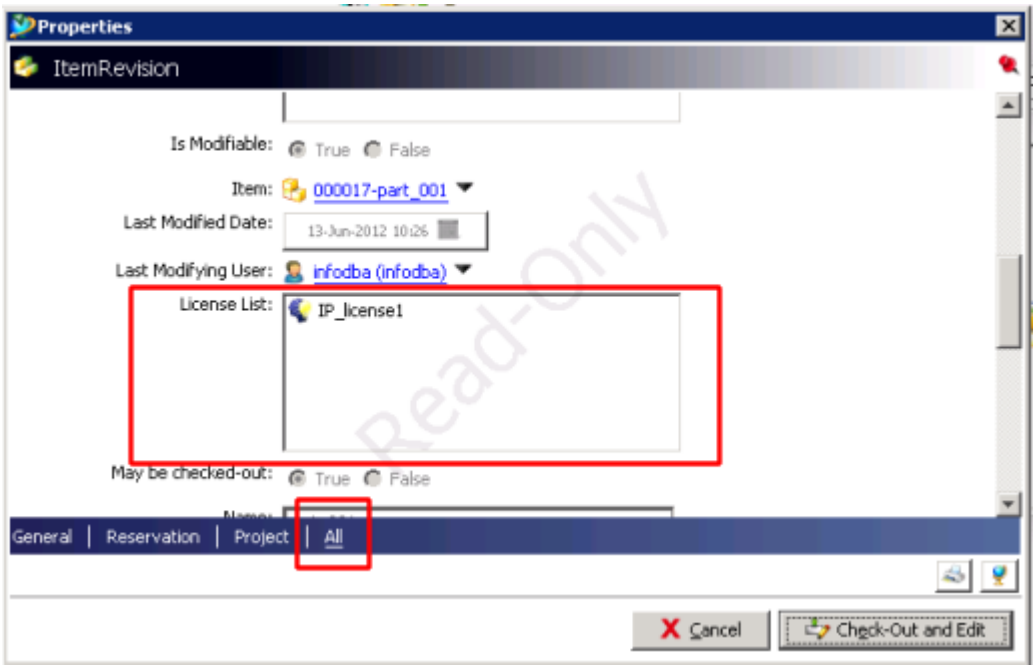
5. Click **OK** or **Apply**.

Note:

When an ITAR license is detached, the associated authorizing paragraph information is deleted.

View licenses attached to a workspace object

1. In My Teamcenter, search for the object for which you want to display the license information.
2. Right-click the object and choose **View Properties**.
3. Click **All**.
4. Scroll to the license information.



Managing ADA license audit logs

Audit log contents

Actions that you perform on an ADA license are tracked in primary audit logs to provide traceability of actions performed on licenses, including who, when, and what. By default, there are two types of log files, license change and license export logs.

License change logs provide traceability of the following actions:

- Creating or deleting a license
- Locking a license
- Modifying a lock date or unlocking a license by clearing the lock date
- Expiring a license or modifying an expiration date
- Adding or removing users or groups allowed to access workspace objects associated with a license

License export logs provide traceability of the following:

- Attaching or detaching a license to or from a workspace object
- Modifying authorizing paragraph information for an ITAR license attached to a workspace object

You can also view secondary audit logs about objects related to the license, such as the workspace object to which the license is attached. The Audit Manager provides basic secondary information by default and your administrator can define more properties to be displayed using the Business Modeler IDE.

You can export the audit data to Microsoft Excel or comma-separated values (CSV) file.

Using Audit Manager, you can view the audit log in the **Audit Logs** tab of the **Summary** view and create search queries.

View audit logs in the Summary view using the current Audit Manager

You can view audit information about the licenses (primary objects), as well as secondary objects, such as the workspace object to which the license is attached. The Audit Manager provides basic secondary information by default and your administrator can define more properties to be displayed using the Business Modeler IDE. The secondary information is in a separate, secondary audit log.

1. From the ADA License application, right-click a license, and choose **Open with→Summary**.

2. In the **Summary** view, click the **Audit Logs** tab.

3. To view the changes to the license, click **License Change Logs**.

   License change logs list the changes to a license.

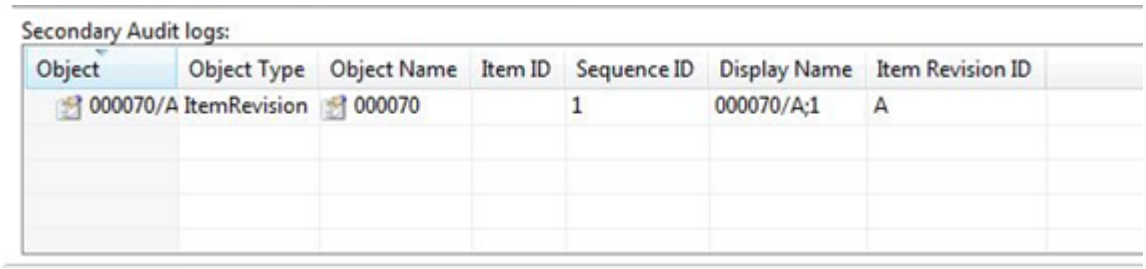
   See *Audit log contents*.

4. To view the ADA License export logs, click **License Export Logs**.

   License export logs lists the attach and detach events of the license.

   See *Audit log contents*.

5. To view audit logs for secondary objects, such as the workspace object to which the license is attached, select the primary object. The secondary object audit log appears below it.



Exporting an audit log using the current Audit Manager

You can export audit logs to Microsoft Excel or CSV formats. You can:

- Select the object whose audit log you want to export. For example, select an object with a license attached.
- Search for an audit log and export it.

> Tip:
>
> Use the following predefined audit log searches:
>
> - **Audit - License Change Logs**
> - **Audit - License Export Logs**

13. Customizing ADA security mapping

About customizing ADA security mapping

Authorized data access (ADA) helps organizations enforce access control on sensitive data through the use of data classification, user clearance, and licenses. In addition to International Traffic in Arms Regulations (ITAR) and intellectual property (IP) groupings, you can achieve compliance or regulation using Export Administration Regulations (EAR) and Export Control Classification Number (ECCN) by using classification, clearance, and licenses to control security on sensitive data.

> Note:
>
> The examples that follow use EAR security mapping.

As an administrator, you can create security mapping for any purpose, but it must follow the rules defined using the following procedure:

1. Using the Business Modeler IDE:
   - Define a custom LOV, which is attached to the custom classification and clearance properties.
   - Create a custom classification property on a workspace object.
   - Create a custom clearance property on a user object.
   - Create the custom ADA license.
2. Deploy the customization.

Creating a custom ADA security map

Step 1. Create a custom LOV for EAR levels

Use the Business Modeler IDE to create a custom LOV, for example, **EAR9_LOV**, which defines the hierarchical ordering for custom properties.

> Note:
>
> In out-of-the-box security groups, the hierarchical ordering is configured using the **IP_level_list_ordering** and **ITAR_level_list_ordering** preferences.

There are different hierarchical security levels assigned to data objects and users for access evaluation. Access Manager compares these levels to determine user access rights to the object; propagation also uses these security levels.

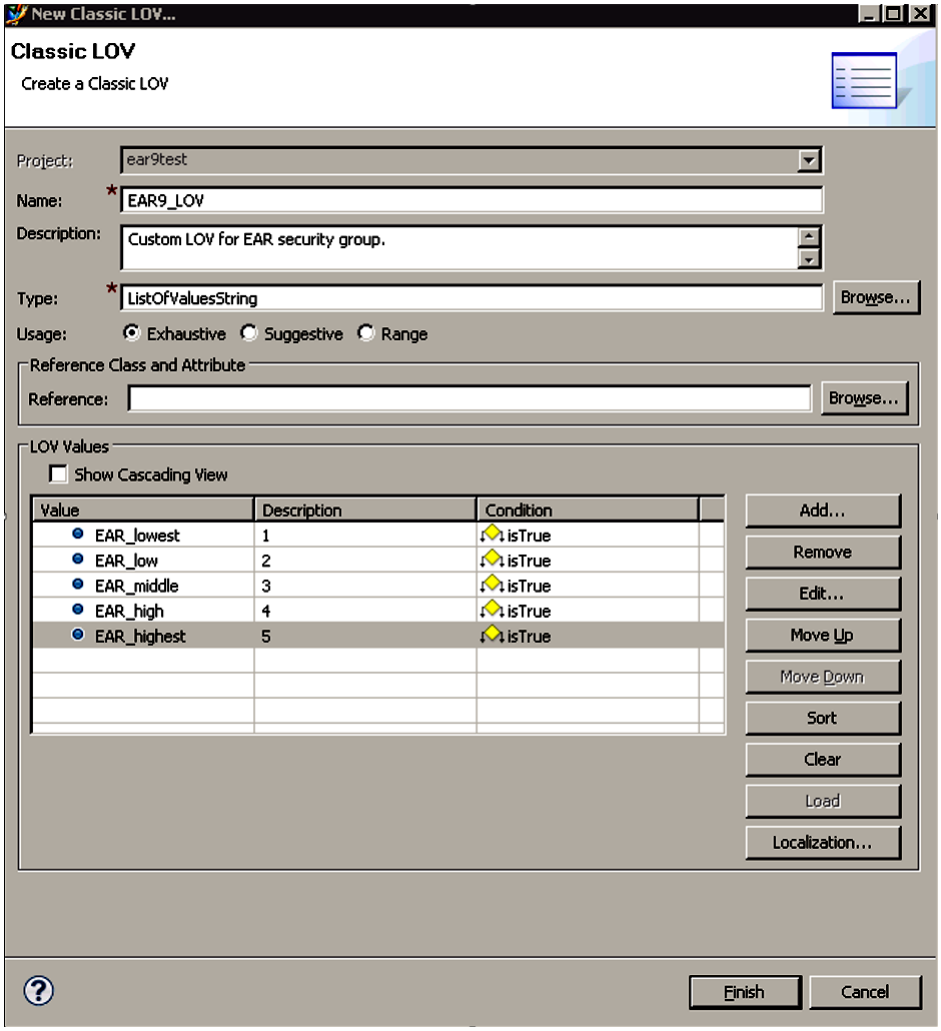
Note:

Classification order now uses the **Description** box contents of the LOV values, where **1** is the lowest level of ordering.

Each site defines a list of classification values and clearance values that are assigned to data objects. This LOV example defines five EAR levels.

Note:

- The list order appears in the list exactly as it appears in the **New Classic LOV** dialog box.
- The **Description** box is used to order the list. To put entries on the same level, enter the same number in the **Description** field.
- Select **Exhaustive** for the LOV usage type. This type is used to define all allowable entries.



Note:

You can use the **LOV_hide_desc** preference to display the LOV values. For example:

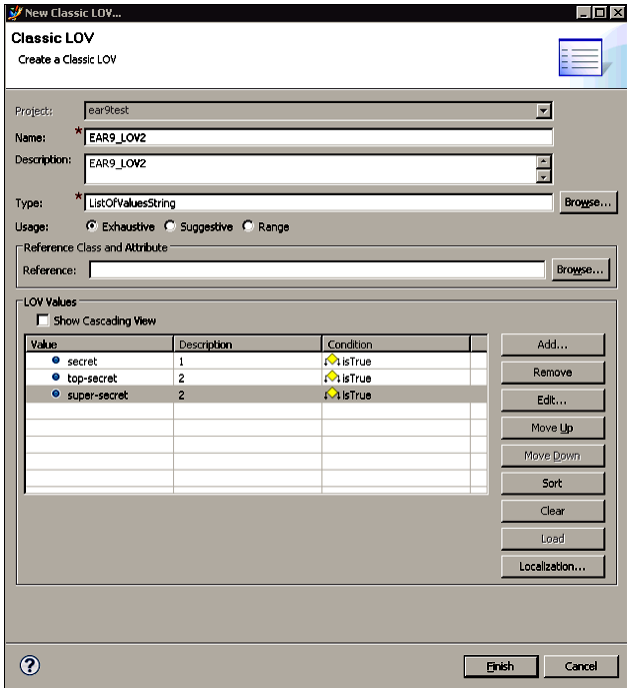
```
LOV_hide_desc=EAR9_LOV
```

Step 2. Create a custom LOV for security levels

Use the Business Modeler IDE to create a custom LOV to be used to designate security levels (**secret**, **top-secret**, and **super-secret**).

Note:

To designate entries to be on the same security level, use the same number in the **Description** box. For example, **top-secret** and **super-secret** are the same level of security because the **Description** box contains a **2** for each.

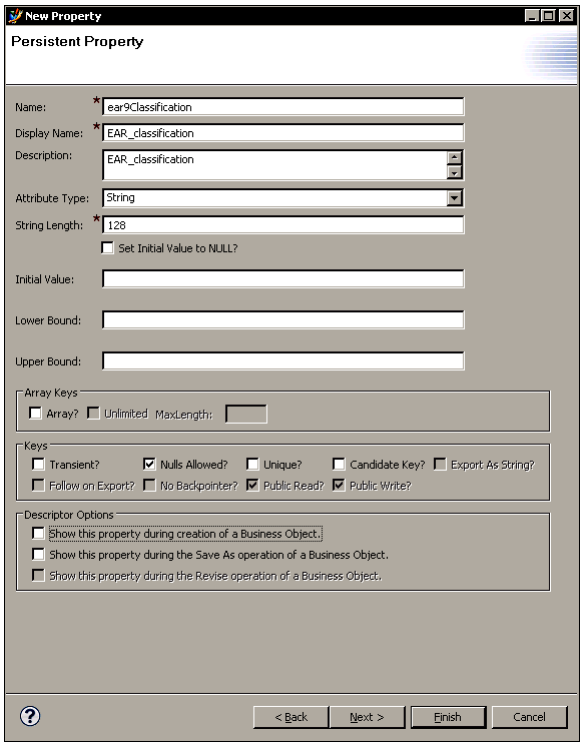


Step 3. Add a custom persistent property

In the Business Modeler IDE, locate any **WorkspaceObject** business object and add a custom classification property.

1. In the **Property Definition** dialog box, select **Persistent** and click **Next**.

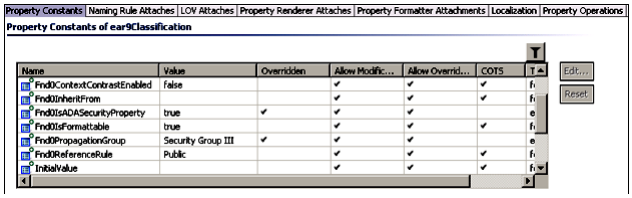
2. Using the **Persistent Property** dialog box, create the property and click **Finish**.



Step 4. Add property constants

1. Set **Fnd0IsADASecurityProperty = true** to indicate this is an ADA custom property.

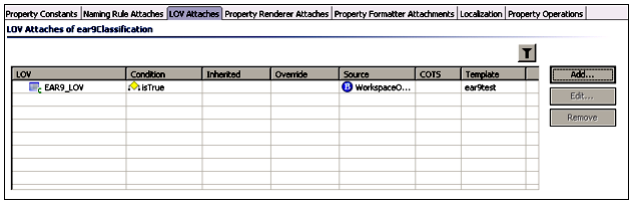
2. Set **Fnd0PropagationGroup** property constant = **Security Group III** to use hierarchical propagation, or set **Fnd0PropagationGroup** = **Security Group IV** to define your own propagation rules.



Step 5. Attach the security level LOV

1. To add the ordered list of security levels, select the security level LOV you created earlier and click the **Add** button on the **LOV Attaches** table to attach the LOV.

2. Attach the LOV to the classification property on the **WorkspaceObject** business object.

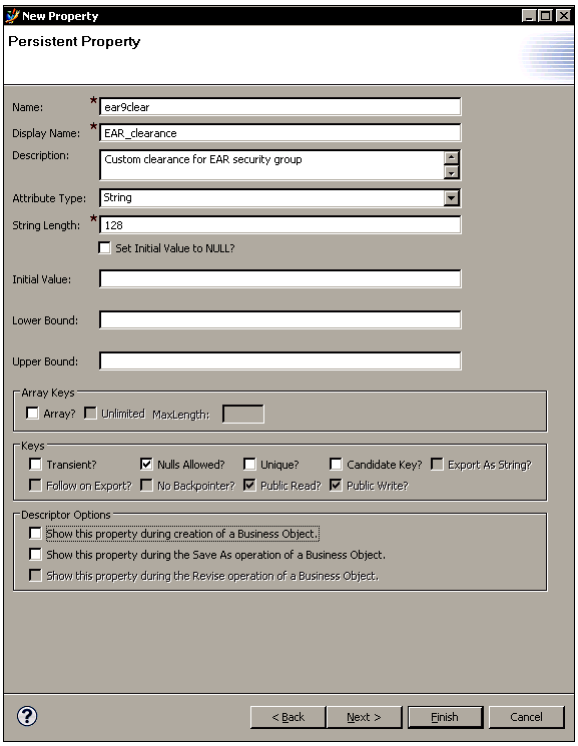


Step 6. Add a custom clearance property for the user

1. Using the Business Modeler IDE, find the **Fnd0CustomUserProfile** business object and add a property. Select **Persistent** and click **Next**.

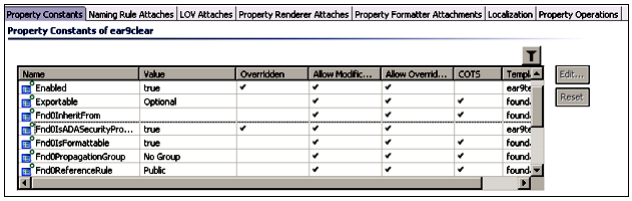
2. Using the **Persistent Property** dialog box, create the property.

> Note:
>
> The clearance name cannot exceed nine characters, for example, **ear9clear**.

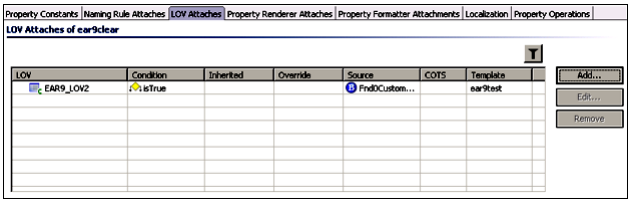


3. Update the property constant.

> Note:
>
> Set **Fnd0IsADASecurityProperty=true** to indicate this is an ADA custom property.



4. Attach the LOV created in the **first step** that contains the ordered list of security levels. If you are using Access Manager conditions and accessors to compare the classification and clearance entries, this LOV should be the same on both properties.



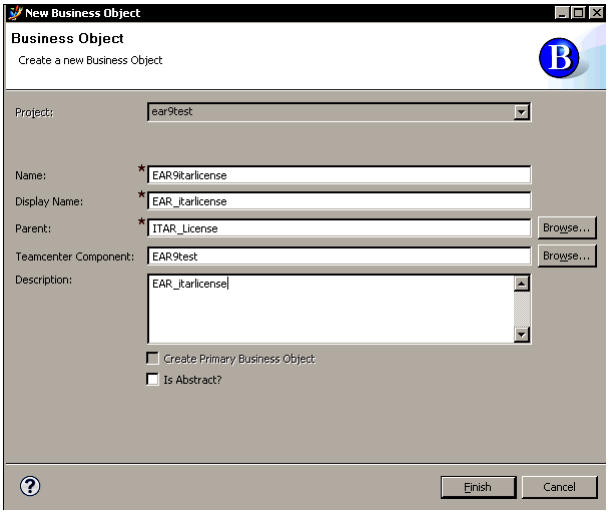
Step 7. Create a custom license subtype

1. To create a subtype of an ITAR license, right-click **ITAR_License** and choose **New Business Object**.

2. Enter values for **Name**, **Teamcenter Component**, and **Description**.

Note:

- You can also subtype **IP_License**. However, subtyping of **ADA_License** and **Exclude_License** is not supported.
- The license name cannot exceed 15 characters, for example, **EAR9itarlicense**.



Step 8. Creating ADMIN and CLASSIFIER privileges

In previous Teamcenter versions, the **ADMIN** and **CLASSIFIER** permissions for ITAR and IP were hard-coded in Access Manager. Because you can add one or many new security groups, you must perform an additional step to create an **ADMIN** and **CLASSIFIER** for your new security group.

Create a custom privilege.

1. Create the privileges by running the **install_am_privilege** utility.

```
TC_ROOT\bin\install_am_privilege -u=tcadmin -p=tcadmin -g=dba -n=EAR_ADMIN
TC_ROOT\bin\install_am_privilege -u=tcadmin -p=tcadmin -g=dba -n=EAR_Classifier
TC_ROOT\bin\install_am_privilege -u=tcadmin -p=tcadmin
     -g=dba -n=ADA_Custom_Detach
```

2. Update the **am_text.xml** file to provide readable names.

3. Update the *TC_ROOT***lang\textserver***language**\am_text_locale.xml** file to provide readable names and tooltips.

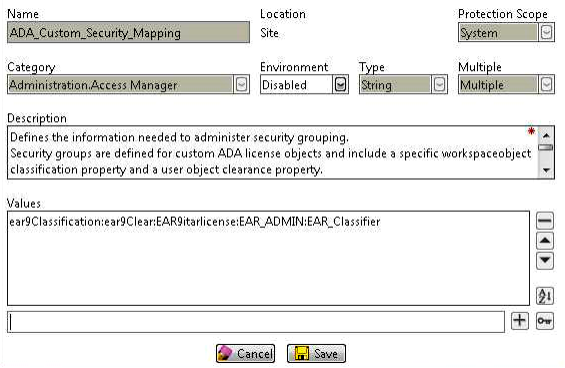
```
<key id="k_am_privilege_EAR_ADMIN">EAR ADMIN</key>
<key id="k_am_privilege_EAR_Classifier">EAR Classifier</key>
<key id="k_am_privilege_ADA_Custom_Detach">ADA Custom Detach</key>
```

Step 9. Using the ADA_Custom_Security_Mapping preference

In previous versions of Teamcenter, you used Access Manager to assign **Allow/Deny** permissions to **IP_ADMIN**, **ITAR_ADMIN**, **IP_CLASSIFIER**, and **ITAR_CLASSIFIER**. Although you still use Access Manager for this purpose, use the **ADA_Custom_Security_Mapping** preference to define which **Admin/Classifier** privileges are associated with each of your ADA custom security mappings.

The **ADA_Custom_Security_Mapping** preference is provided out of the box and empty:

- Assigns an **ADMIN** and **CLASSIFIER** for each ADA security mapping.
- Provides a link for the **WSO.Classification**, **User.Clearance**, and custom license security mapping.



The format of each entry is:

- Classification:
- Clearance:
- License:
- **ADMIN** privilege:
- **CLASSIFIER** privilege:

The **ADMIN** privilege allows you to:

- Create, modify, and delete ADA licenses.
- Attach and detach licenses to workspace objects.
- Set or modify the classification property on workspace objects.

The **CLASSIFIER** privilege allows you to set or modify the classification property on workspace objects.

> Note:
>
> The ability to update the custom clearance values for the user is *only* dependent on the user having authorization to create and modify the user. This is usually allowed if you have **DBA** privileges or given access through the Authorization application.

Step 10. Using Access Manager conditions with custom security groups

All existing Access Manager conditions work as they did before with IP and ITAR (government) classification, clearance, and licenses. However, for custom security groups:

- If there is a generic condition (**Has Named ADA License**), the value field can be overloaded to handle the custom property/license.
- If no generic condition exists, one is added.

> Note:
>
> All generic license conditions have **ADA** in the condition name.

- For the negative conditions, a variety of approaches can be used.

Clearance and classification conditions

ITAR (existing)	IP (existing)	Custom (new)
Has Government Classification	**Has IP Classification**	**Has Classification**
Has No Government Classification	**Has No IP Classification**	**Has No Classification**
User Has Government Clearance	**User Has IP Clearance**	**User Has Clearance**

The syntax for the new clearance and classification conditions follows:

- **Has Classification**(*Custom_classification*>=**custom_level1**)
- **Has No Classification**(*Custom_classification*)
- **User Has Clearance**(*Custom_clearance*>=**custom_level1**)

License conditions

ITAR (existing)	IP (existing)	Custom (new)
User Is ITAR Licensed	**User Is IP Licensed**	**User Is ADA Licensed**
User In Named ITAR License	**User In Named IP License**	**User In Named ADA License**
User In Attached ITAR License	**User In Attached IP License**	**User In Attached ADA License**
Has Named ITAR License	**Has Named IP License**	**Has Named ADA License**

ITAR (existing)	IP (existing)	Custom (new)
Has ITAR License Of Category	Has IP License Of Category	Has ADA License Of Category
Citizenship On Any ITAR Lic	Citizenship On Any IP Lic	Citizenship On Any ADA Lic
User In Attach ITAR Lic of Ctgry	User In Attach IP Lic of Ctgry	User In Attach ADA Lic of Ctgry
User-ITAR Lic Has Citizenship	User-IP Lic Has Citizenship	User-ADA Lic Has Citizenship
ITAR License Has Citizenship	IP License Has Citizenship	ADA License Has Citizenship

The syntax for the new license conditions follows:

- **User In Attached ADA License**(*Custom_License_type*:**Any** or **All**)
- **User In Named ADA License**(*Custom_License_type*:**LicenseID**)
- **User Is ADA Licensed**(*Custom_License_type*:**True** or **False**)

New Access Manager conditions

- **Has No Status**
- **User Not In Attach IP Lic Ctg**
- **User Not In Attach ITAR Lic Ctg**
- **User Not In Attach Excl Lic Ctg**
- **User In Attached ADA License**
- **User In Named ADA License**

Modified Access Manager conditions

- **Has Property**
- **Has Attribute**
- **User In Attached ITAR License**

- **User In Attached IP License**
- **User In Attached Exclude License**
- **User In Attached License**

Access Manager accessor types

Each of the following Access Manager accessor types is added to allow the custom properties to be searched.

> Note:
>
> The following clearance accessor types require you to manually add entries in the **am_text.xml** file for each custom clearance property you add.

- License accessor types
  - **User Licensed**

    Arg Values=License Types
  - **User Unlicensed**

    Arg Values=License Types

> Note:
>
> For each of the custom licenses added, two new accessors are created in Access Manager application. For example, if **EAR_ITAR_Lic** is the custom license name, then the following two entries appear in the accessor drop-down list:
>
> - **User EAR_ITAR_Lic Licenses**
> - **User EAR_ITAR_Lic Unlicenses**
>
> To see these entries, you must add the following entries in the **am_text.xml** file:
>
> ```
> <key id="k_am_accessor_key_user_ear_itar_lic_licensed">Ak</key>
>
> <key id="k_am_accessor_key_user_ear_itar_lic_unlicensed">Ak</key>
> ```
>
> In this example, **Ak** is a sample value; you must replace it with the running key ID. Also, the internal license name given in the key should be in lowercase. For example, it should *not* be **k_am_accessor_key_user_EAR_ITAR_Lic_licensed.**.

- Clearance accessor types
  - **User has Clearance** custom_clearance **Clearance**

    Arg Values=Custom Clearance properties
  - **User over** custom_clearance **Clearance**
  - **User under** custom_clearance **Clearance**

    Arg Values=Custom Clearance properties

> Note:
>
> For each custom clearance property added, three new accessors are created in Access Manager application. For example, if **EAR_Clearance** is the custom clearance property name, the following three entries appear in the accessor drop-down list:
>
> ```
> <key id="k_am_accessor_key_user_has_ear_clearance_clearance">
> <key id="k_am_accessor_key_user_under_ear_clearance_clearance">Aj</key>
> <key id="k_am_accessor_key_user_over_ear_clearance_clearance">Aj</key>
> ```

Aj is a sample value; you must replace it with the running key ID. The property name used must be lowercase.

Step 11. Custom classification propagation

To follow the hierarchical pattern established by the out-of-the box **IP Classification** and **Gov Classification** properties:

- Create the LOV and enter the desired levels in the **Description** box.
- Enter **Security Group III** in the **Fnd0PropagationGroup** property constant for each custom classification properties you create.

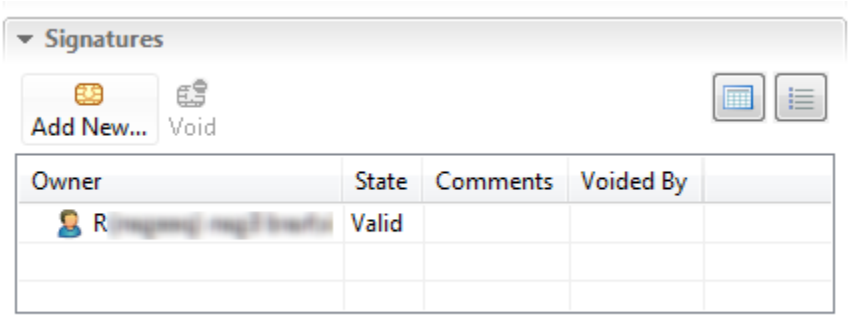
To create your own propagation patterns:

- Enter **Security Group IV** in the **Fnd0PropagationGroup** property constant for each custom classification properties you create.
- Add customer specific rules to the **Security Group IV** for your company's propagation rules.

14. Configuring PKI digital signature

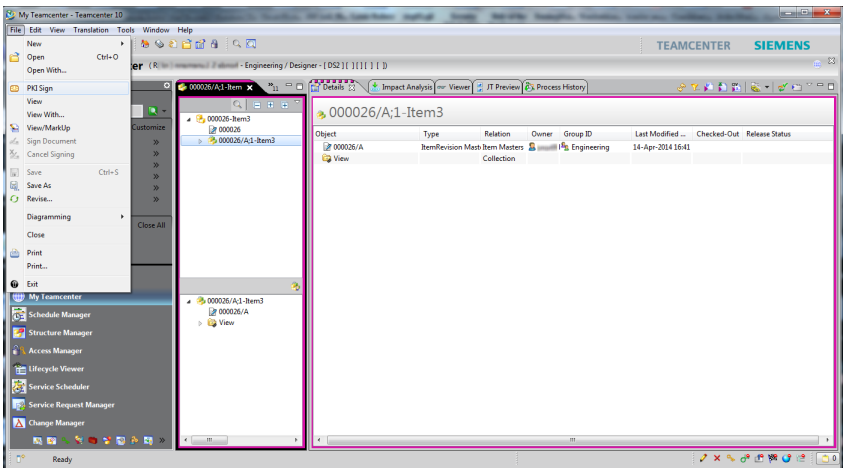
What is PKI digital signature?

A *digital signature* is a mathematical stamp on an object used to indicate if that object has been modified after the signature was applied. It also identifies who applied the digital signature. It requires public key infrastructure (PKI) authentication when applying the digital signature.

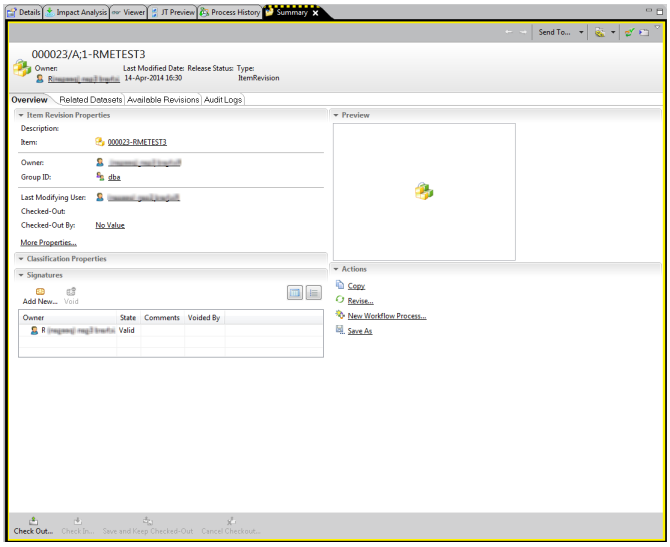


> Note:
>
> PKI *digital signature* is different than *digitally sign*. Digitally sign is a Teamcenter action associated with document management that lets you add your digital signature to an electronic document using third-party software, for example, Adobe Acrobat/Reader for an Adobe Portable Document Format (PDF) file.

After you configure your system to apply the PKI digital signature on objects, a rich client user can select an instance of a business object and choose **File→PKI Sign**.



Then, the PKI digital signature (valid) icon is applied.



Configuring your system for PKI digital signature

Install Teamcenter

> Note:
>
> The PKI digital signature feature is supported only on the Windows platform client.

Before you configure your system for PKI digital signature, you must install Teamcenter. Your current environment determines which installation type you select.

- New installation

  > Note:
  >
  > When installing this version of Teamcenter for the first time, all Access Manager rules are included. However, you must install certificate authority files during your new installation, as described in **Step 1**.

- Upgrade installation

  Upgrade your existing Teamcenter environment.

- Patch installation

  Patch your existing Teamcenter environment, if necessary.

  See **Step 1** to install certificate authority files during patch installation.

Note:

While patching using Teamcenter Environment Manager, select **Optional Configuration Enhancements**.

Step 1: Enter digital signature certificate information

For PKI digital signature, Teamcenter uses the private key from the user's PKI card to encrypt the signature so it cannot be tampered with. After the signature is applied to an object, the system reads the encrypted signature to verify the signature is valid. To verify the encrypted signature, the system must have the root and intermediate certificate authority (CA) files.

Each user certificate (PKI card) has a chain of certificate authorities behind it. These certificate authorities can be found by opening the user's certificate on Windows systems and clicking the **Certification Path** tab. The certificate authorities should already be installed on the client host so that the Web browser on that machine recognizes the user's certificate.

Note:

When choosing the certification tiles for the **RootCA** chain, you must choose the Base-64 encoded x.509 (**.CER**) file, as the server only works with this format.

You must enter the digital signature certification information in Teamcenter using either of the following methods:

- Using Teamcenter Environment Manager (TEM) during a new installation or a patch installation

    1. From the **Foundation Settings** panel, click the **Advanced** button to display the **Digital Signature Certificates** tab.

2. From the **Digital Signature Certificates** tab, click the **Add** button to add a certificate path in the **Certificates** list.

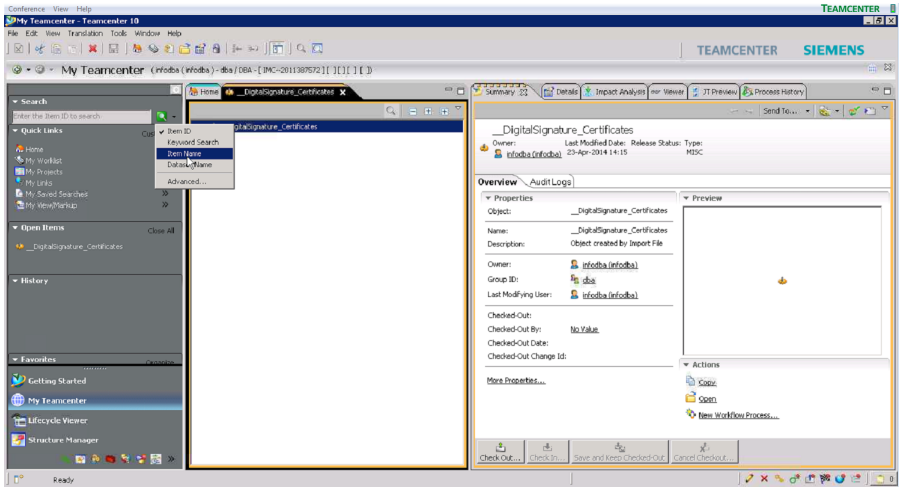
> Note:
>
> The certificates that are loaded into the database must be loaded as **MISC_BINARY**. The default is **MISC_TEXT** when loading them manually.

3. After you compete adding the certificates for your system, click **OK** from the **Digital Signature Certificates** tab to save the paths and continue your installation in TEM.

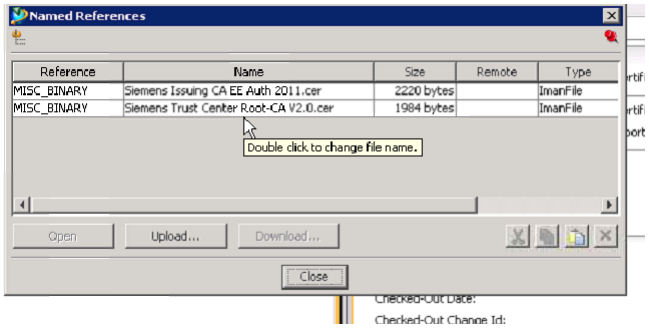
- Using the rich client after upgrading or anytime following installation (new installation or patch installation)

> Note:
>
> Use this method to update your certificate authority file information when your certificate expires or if you have a new certificate.

1. Search for the **__DigitalSignature_Certificates** dataset (note the leading double underscore).



2. Check out and modify the dataset as needed.



Step 2: Apply business constants

Before you can apply a PKI digital signature to an object, you must set three business constants in the Business Modeler IDE. Setting these constants allows the selected business objects, for example, an **Item** object, to have a PKI digital signature.

- **Fnd0AllowDigitalSignature**

  This constant, which is placed on the **WorkspaceObject** business object and its children, allows the selected business object have a PKI digital signature.

  By default, this constant is set to **false**. Select the **Value** check box to change the value between **true** and **false**.

- **Fnd0DigitalSignatureAttributes**

  This constant contains a list of properties that are approved when applying a PKI digital signature to a business object instance.

  When listing the properties to be used with **Fnd0DigitalSignatureAttributes**, use the following format:

  ```
  {property1,property2,...propertyN}
  ```

  Note:

  This constant can hold a maximum of 2000 characters to allow for multiple configuration versions.

  Caution:

  Following are the properties that cannot be used with this constant:

  acl_bits
  active_seq
  actual_finish_date
  archive_date
  archive_info
  backup_date
  checked_out
  checked_out_date
  checked_out_user
  CMClosure
  CMDisposition
  CMMaturity
  complete_percent

creation_date
date_released
fnd0state
last_mod_date
last_mod_user
last_sync_date
lsd
owning_group
owning_site
owning_user
process_stage
process_stage_list
process_status_list
proj_assign_mod_date
project_ids
project_list
projects_list
release_status_list

- **Fnd0DigitalSignatureChildObjects**

  This constant contains a list of relation or reference properties to retrieve child objects for PKI digital signature. This constant works in conjunction with the **Fnd0DigitalSignatureAttributes** constant.

  Note:

  This constant can hold a maximum of 2000 characters to allow for multiple configuration versions.

Step 3: Edit XRT (style sheet)

You must edit the XRT style sheet for the object type (for example, **ItemRevSummary.xml**) to which you want to apply PKI digital signatures. This allows users of the client to see digital signature information in the user interface.

Note:

You can edit more than one object type's XRT style sheet depending on which objects you want to apply PKI digital signatures.

1. From the rich client, go to the **Search** pane in the upper-left corner and search for the XRT dataset of the object type, for example, **ItemRevSummary.xml**.

2. Click the **Viewer** tab and the style sheet code appears.

   Locate the end of the **tc_xrt_ClassificationProperties** section:

```
<section titleKey=”tc_xrt_ClassificationProperties”>
...
</classificationiProperties/>
</section>
```

3. Add the following **tc_xrt_Signatures** code segment and save the file.

   This code segment ensures that once a user digitally signs any item from the **Home** menu, the **Summary** page displays the details.

```
<section titleKey="tc_xrt_Signatures">
     <objectSet
source="Fnd0DigitalSignatureRel.Fnd0DigitalSignature"
                        defaultdisplay="tableDisplay"
sortby="object_string"
                        sortdirection="ascending">
        <tableDisplay>
           <property name="owning_user"/>
           <property name="fnd0State"/>
           <property name="fnd0Comments"/>
           <property name="fnd0VoidedBy"/>
        </tableDisplay>
        <listDisplay/>
        <command commandId="com.teamcenter.rac.applyDigitalSign"
                        titleKey="tc_xrt_addNew"
                 renderingHint="commandbutton"/>
        <command commandId="com.teamcenter.rac.voidDigitalSign"
                 renderingHint="commandbutton"/>
        <parameter name="localSelection" value="true"/>
     </objectSet>
</section>
```

Step 4: Create the rich client plug-in

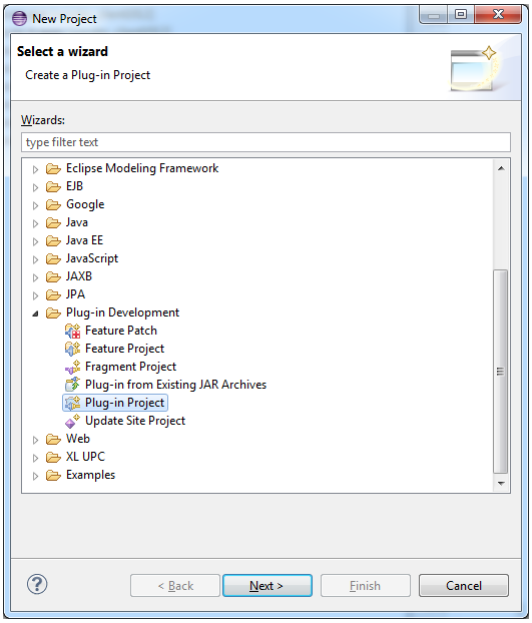
To add the **PKI Sign** and **Void** commands to the **File** menu and the shortcut menu in the rich client, you must define a custom **plugin.xml** file.

> Note:
>
> Steps for creating a rich client plug-in are included in this topic. However, you can view examples for creating plug-ins.

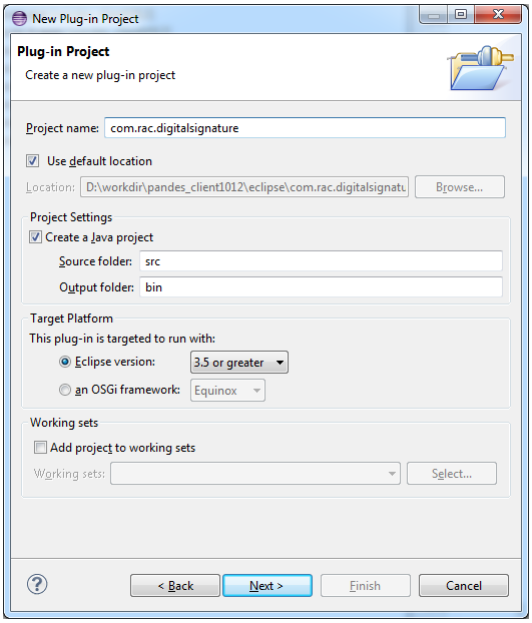
1. Create a new plug-in project.

   a. In Eclipse, choose **File→New→Project**.

b. In the **New Project** dialog box, select **Plug-in Project**. Click **Next**.



c. In the New Plug-in Project wizard **Plug-in Project** dialog box, type **com.rac.digitalsignature** in the **Project name** box. Click **Next**.



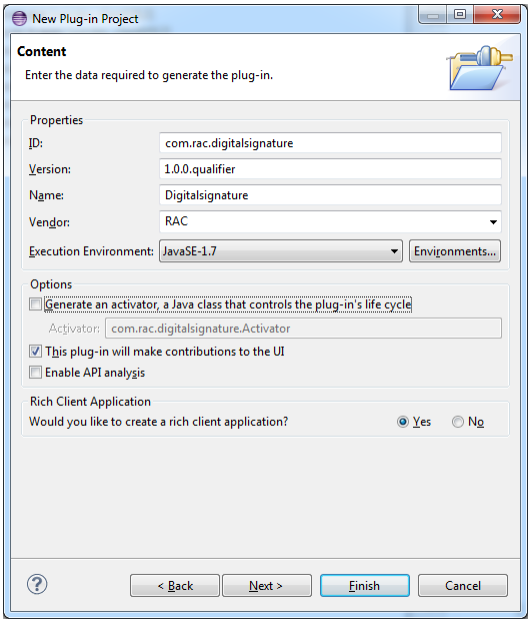
d. In the New Plug-in Project wizard **Content** dialog box, do the following:

A. Under **Options**, ensure the **Generate an activator, a Java class that controls the plug-in's life cycle** is *not* selected.

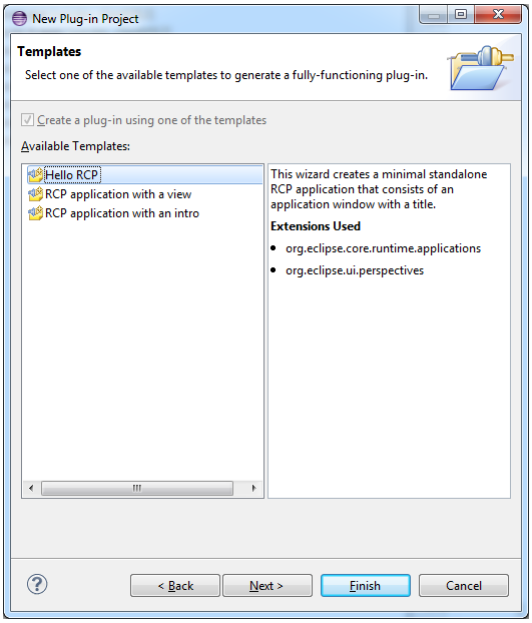
B. Ensure the **This plug-in will make contributions to the UI** check box is selected.

C. Click the **Yes** button next to **Would you like to create a rich client application?**.

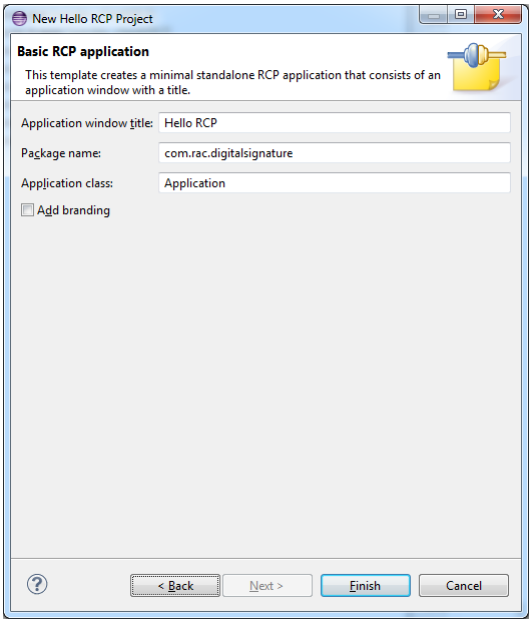
D. Click **Next**.



e. In the New Plug-in Project wizard **Templates** pane, select the **Hello RCP** template to create your plug-in. Click **Finish**.

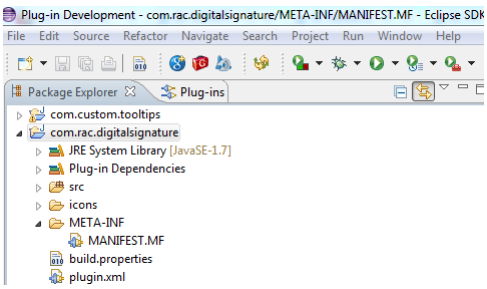


f. In the New Hello RCP Project wizard **Basic RCP application** pane, enter the following information, designating the package name as **com.rac.digitalsignature**. Click **Finish**.

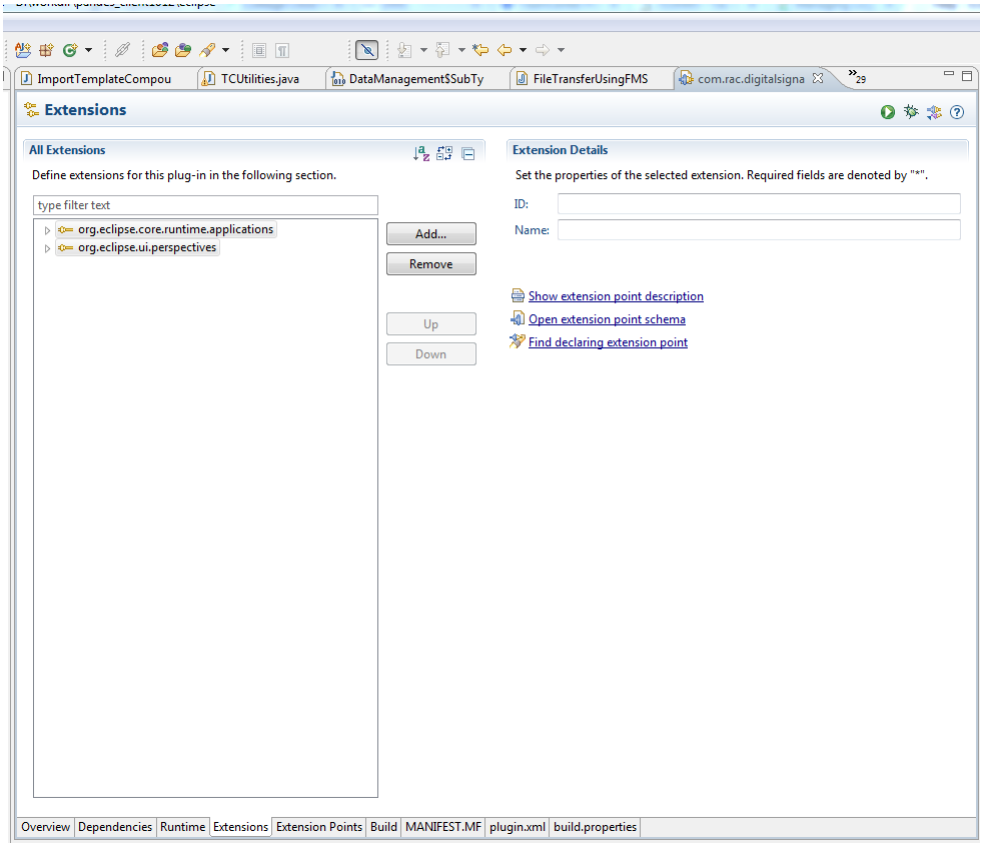


2. Add the menu code to the project plug-in.

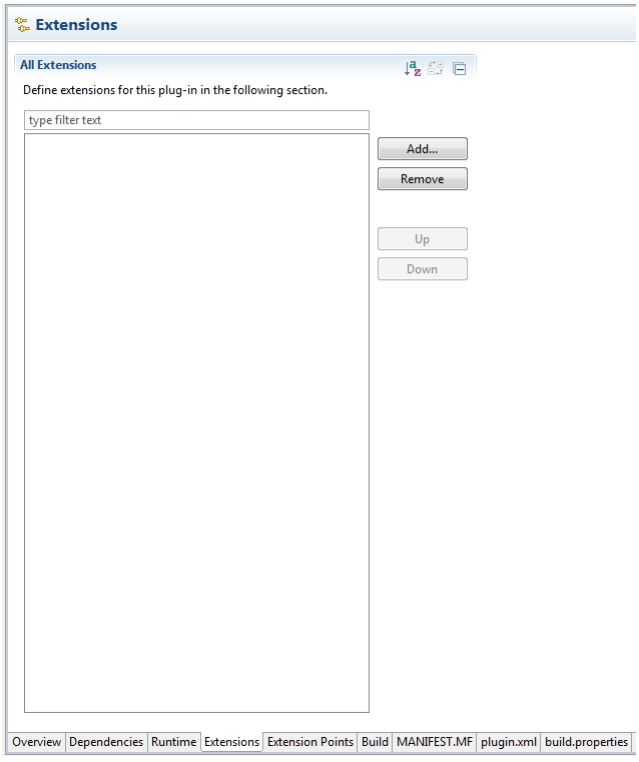
   a. Create a file launcher by expanding the **com.rac.digitalsignature** package to the **META-INF** folder in the **Package Explorer** tab and double-click **MANIFEST.MF**.



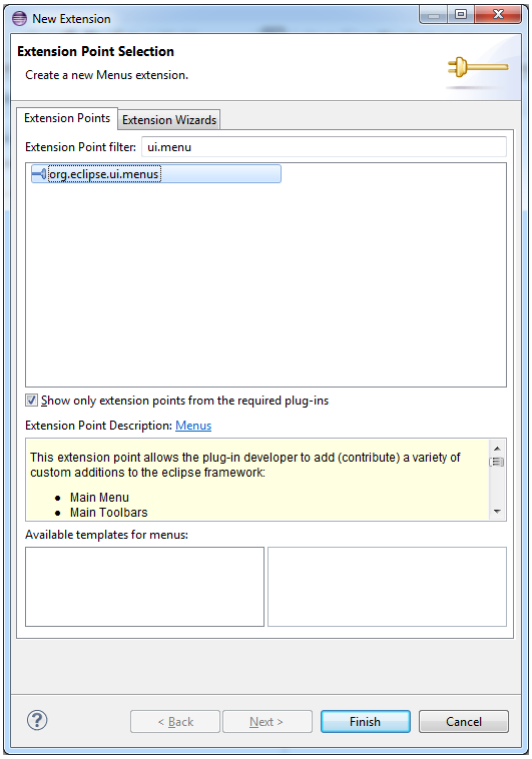
   b. Click the **Extensions** tab at the bottom of the center pane.



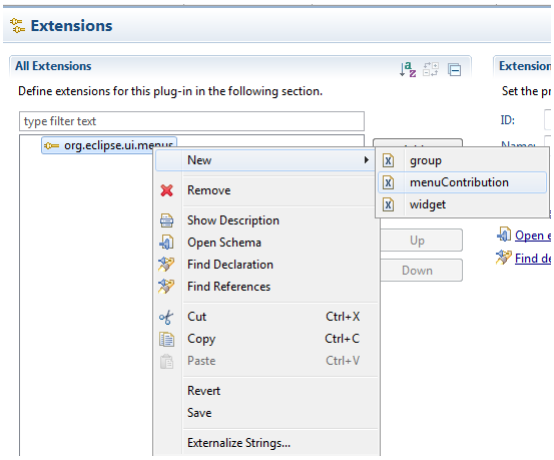
c. Click **Add** in the **Extensions** pane.



d. In the **Extension Point Selection** pane, type **ui.menu** in the **Extension Point** filter box to filter the extension points. Select **org.eclipse.ui.menus**, and click **Finish**.



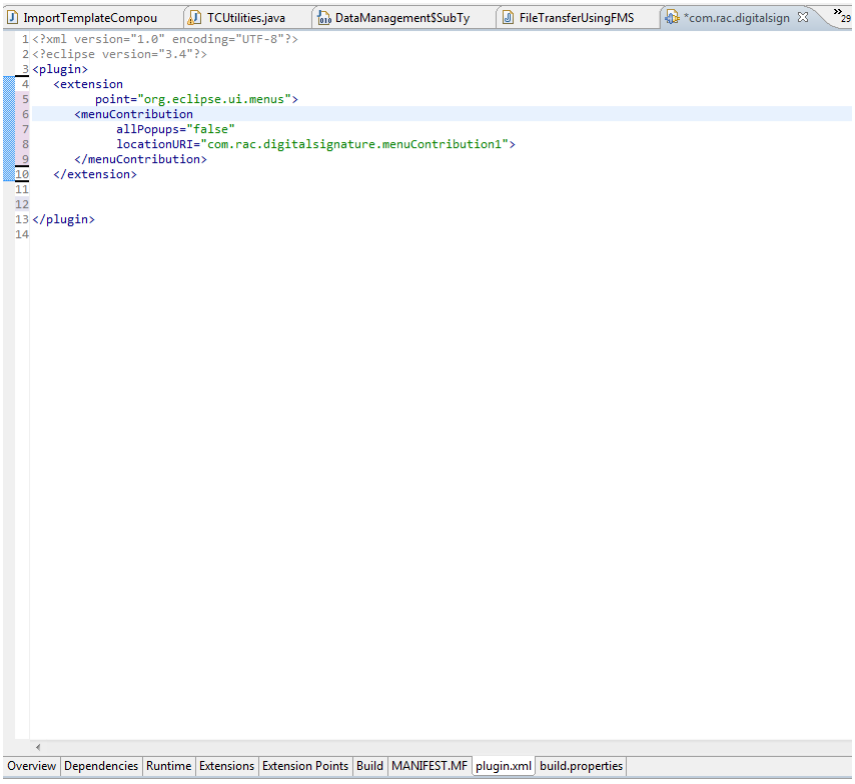
e. In the **All Extensions** list, right-click **org.eclipse.ul.menus** and choose **New→menuContribution**.



f. In the **plugin.xml** tab, replace the following code:

```
<menuContribution
        all Popups=”false”
```

```
            locationURI=”com.rac.digitalsignature.menuContribution1”>
    <menuContribution>
```

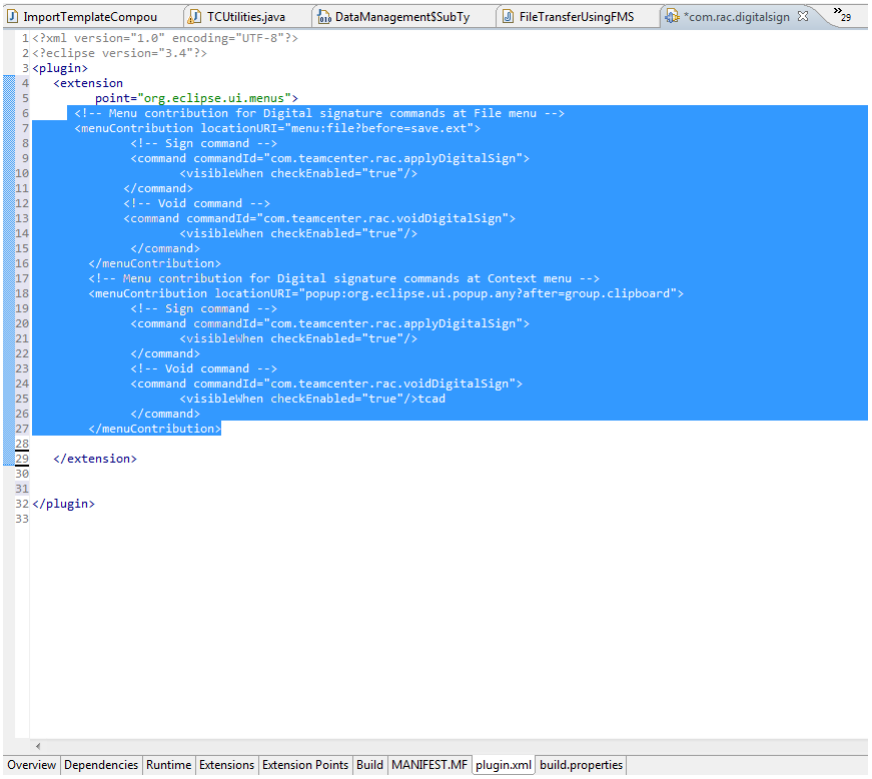


with the following:

```
<!-- Menu contribution for Digital signature commands at File
menu -->
<menuContribution locationURI="menu:file?before=save.ext">
     <!-- Sign command -->
     <command commandId="com.teamcenter.rac.applyDigitalSign">
          <visibleWhen checkEnabled=”true”/>
     </command>
     <!-- Void command -->
     <command commandId="com.teamcenter.rac.voidDigitalSign">
          <visibleWhen checkEnabled=”true”/>
     </command>
</menuContribution>
<!-- Menu contribution for Digital signature commands at Context
menu -->
<menuContribution
   locationURI="popup:org.eclipse.ui.popup.any?
after=group.clipboard">
     <!-- Sign command -->
     <command commandId="com.teamcenter.rac.applyDigitalSign">
```

```
            <visibleWhen checkEnabled=”true”/>
        </command>
        <!-- Void command -->
        <command commandId="com.teamcenter.rac.voidDigitalSign">
            <visibleWhen checkEnabled=”true”/>
        </command>
    </menuContribution>
```

Your file should resemble the following.

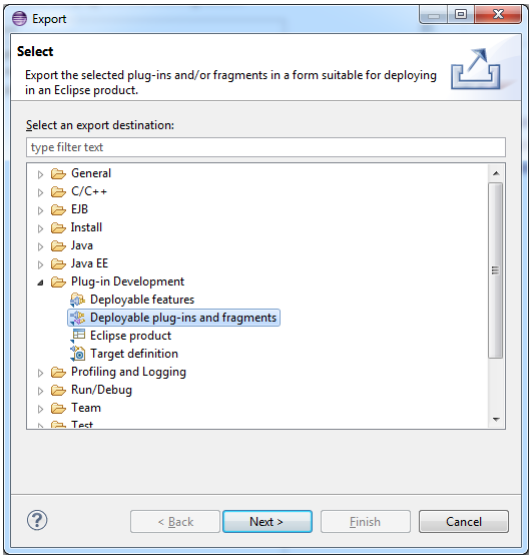


g. Save your changes.

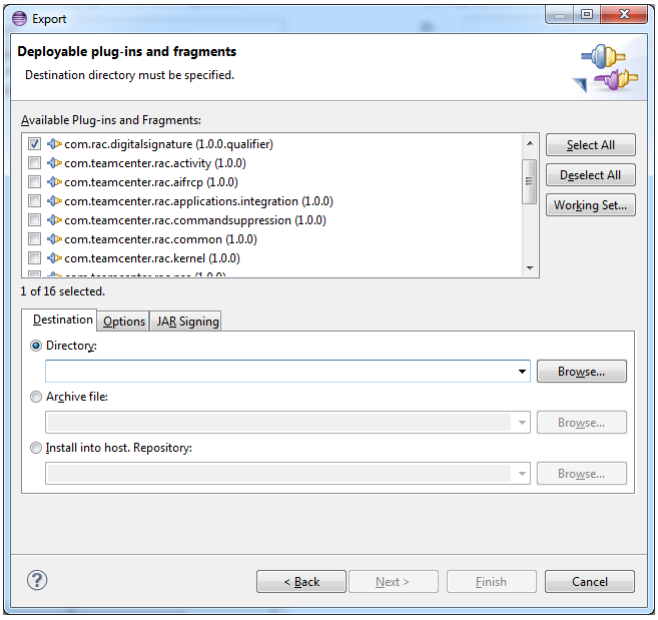
3. Export the project plug-in into a file.

a. In the **Package Explorer** tab, right-click **com.rac.digitalsignature** and choose **Export→Plug-in Deployment**.

b. Choose **Deployable plug-ins and fragments** and click **Next**.



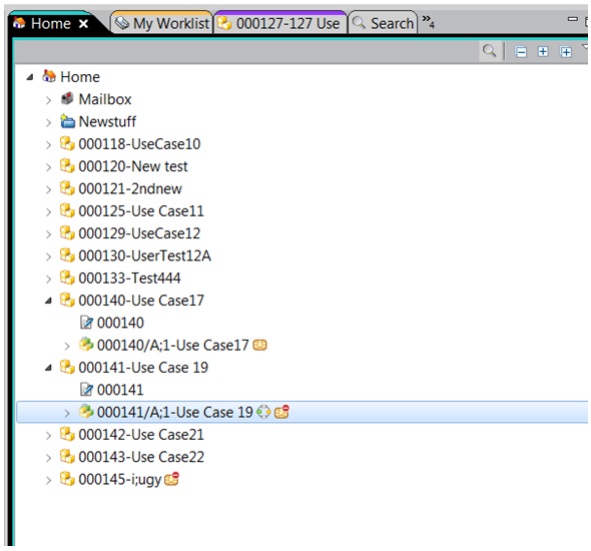
c. Select **com.rac.digitalsignature** and browse to the directory where you want the plug-in exported and click **Finish**.



4. Run the **genregxml.bat** file to register your plug-in changes with the rich client.

Step 5: Set the preference to display digital signature icons

Icons appear to the right of an object in the **Home** view indicating it has either a valid PKI digital signature or a voided PKI digital signature.



To enable the icons showing either a valid PKI digital signature or a voided PKI digital signature on a work item, you must set the **TC_show_digital_signature** icon preference to **TRUE**.

Step 6: Set PKI digital signature environment variables

> Note:
>
> Setting these environment variables for PKI digital signature is optional. However, you may find these helpful if you have a custom configuration.

You can set two Teamcenter environment variables in either your **tc_profilevars.bat** property file or your **tc_profilevars.sh** property file.

- **TC_DS_USERID_VALIDATION_ENABLED**

  > Note:
  >
  > This environment variable is unavailable in the **tc_profilevars.bat** property file or your **tc_profilevars.sh** property file; you must add it manually.

  Enables or disables user ID validation performed when creating new PKI digital signatures. As part of user ID validation, Teamcenter makes certain that the signer (person who is creating the digital signature) is the same as the authenticated user.

  This variable uses a Boolean data type, for example, **True** or **False**. The default value is **True**.

- **TC_DS_HASH_ALGORITHMS**

  Specifies a list of hash algorithms that are accepted by Teamcenter when creating new PKI digital signatures.

The default Teamcenter installation is configured to use strong hash algorithms as specified in the default value (**SHA256, SHA224,SHA386,SHA512**). You can add hash algorithms that are unavailable in the default value, such as **SHA1**.

You can add this variable to your **tc_profilevars.bat** property file or your **tc_profilevars.sh** property file to override the default values.

Note:

- If you add digital signature during a patch, check that this variable is set in your **tc_profilevars.bat** file or your **tc_profilevars.sh** file and provide the default list of values (**SHA256, SHA224,SHA386,SHA512**).
- Using weak has algorithm, such as **MD2, MD5**, may allow PKI digital signature forgery. Analyze the security risk before adding weak algorithms.

Step 7: Generate Access Manager rules

Note:

If you installed Teamcenter as a new installation, do not perform this step; it has already been done as part of the installation.

If you have installed Teamcenter as an upgrade or a patch, you must generate Access Manager rules.

Use the following steps to manually add the Access Manager rules following an upgrade or a patch:

1. Select the **Has Class (POM_object)** root node.
2. Select the appropriate condition and value and specify the appropriate ACL name.
3. Click the **Add** button to the right of the access control entry (ACE) table to add a new row to the table.
4. Grant and revoke permissions.
5. Click the **Add** button at the bottom of the pane to add the rule to the Access Manager tree.
6. Select the newly added rule in the rule tree.
7. Scroll to the appropriate locating using the arrow keys (**Move Rule Up** and **Move Rule Down**) in the toolbar.
8. Click the **Save** button in the toolbar.

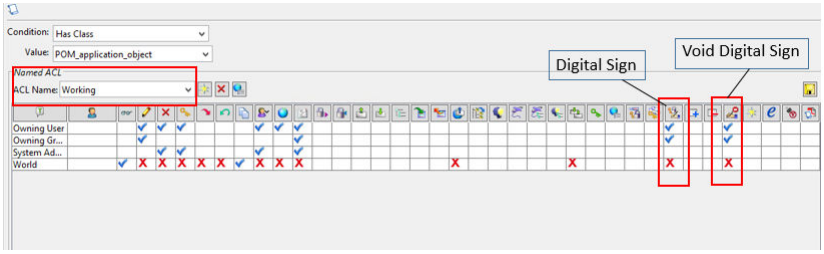
After you generate the Access Manager rules, you can assign the following access privileges associated with the PKI digital signature feature:

- **Digitally Sign**

  Allows the owning group and owning user the privilege to apply a digital signature using PKI authentication. World users do not have digitally sign privileges.

- **Void Digital Sign**

  Allows the owning group and owning user to revoke or cancel an existing PKI digital signature for a business object. World users do not have void digital sign privileges.



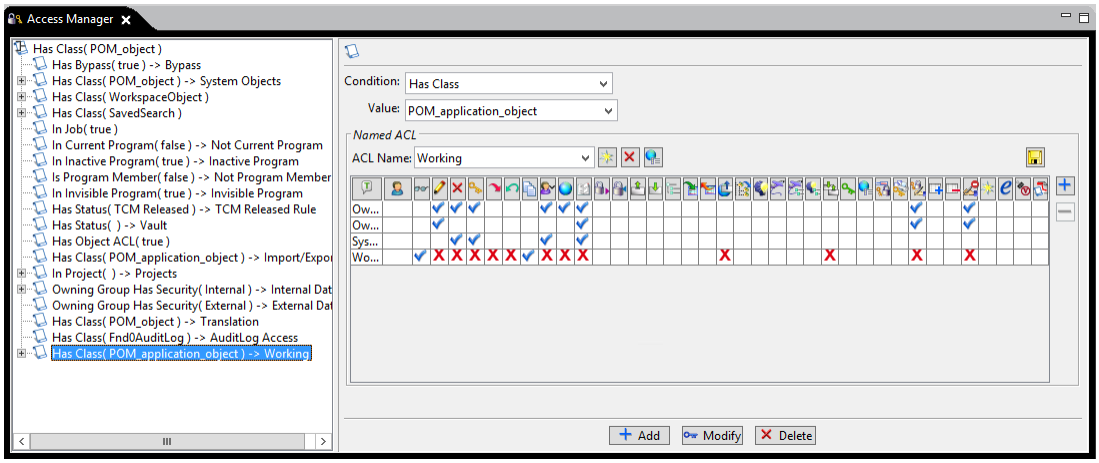
The **Has Digital Signature** Access Manager condition has the following values: **Valid**, **Invalid**, **Propagated**, **Revoked**, and **Voided**.

> Caution:
>
> A dataset digital signature will not show as invalid if the file on the volume is modified outside of Teamcenter control. A Teamcenter dataset-level digital signature will not protect against, or notify users, of a malicious modification of the file at the OS storage device level.

There are five AM rules you must configure manually:

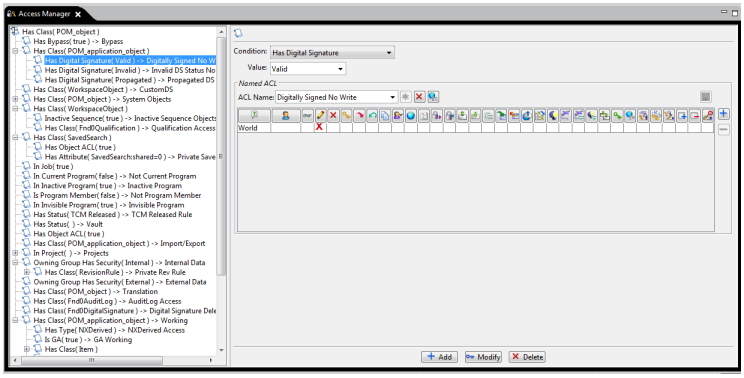
- Use AM rules to control access to applying and voiding PKI digital signatures on **POM_application_object** business objects.

  To do this, you must modify **Working** to grant owning group and owning user the privilege to apply a digital signature and void an existing digital signature while denying all other users digitally sign and void digital sign privilege.



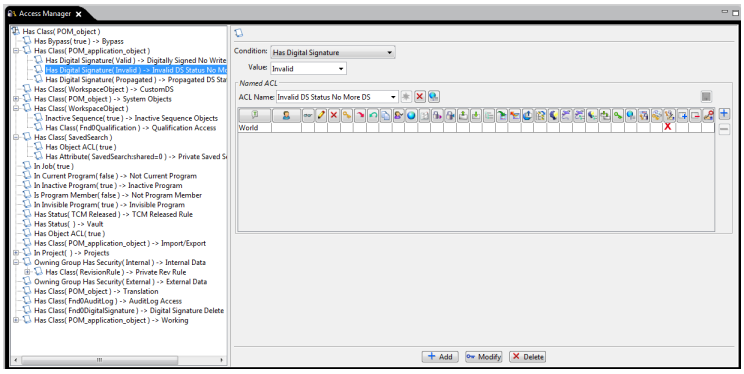
- Use AM rules to control the **Write** privilege on digitally signed objects by setting **Digitally Signed No Write**.

  Configure Access Manager rules to control the user's **Write** privilege on **POM_application_object** business objects based on the existence of digital signature on the object.



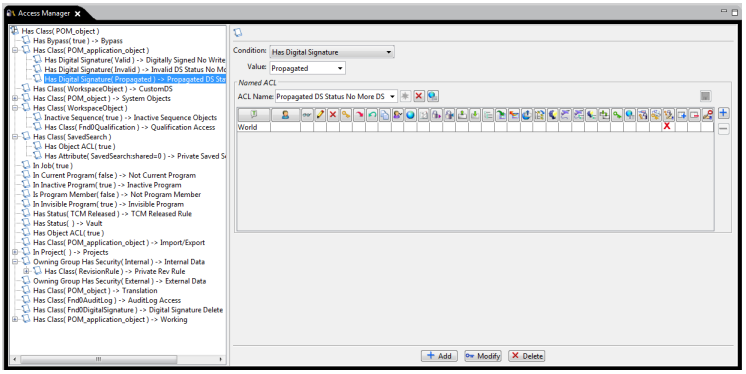
- Use AM rules to prevent a digital signature on a **POM_application_object** with an invalid PKI digital signature by setting **Invalid DS Status No More DS**.

  This configures Access Manager rules to disallow any user to digitally sign target **POM_application_object** business objects that have invalid digital signatures.



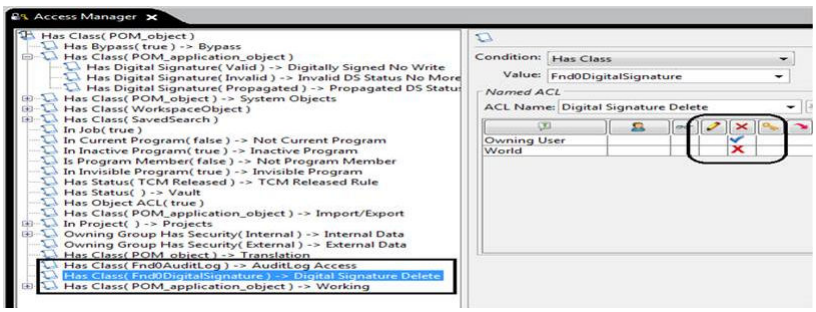
- Use AM rules to prevent the reapplication of a PKI digital signature on a **POM_application_object** that has a propagated PKI digital signature by setting **Propagated DS Status No More DS**.

  This configures Access Manager rules to disallow any user to digitally sign target **POM_application_object** business objects that have propagated digital signature.



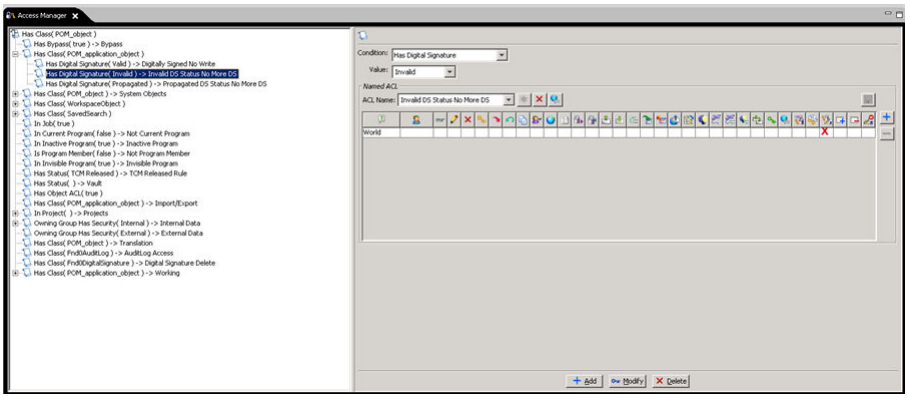
- Use AM rules to control the deletion of a PKI digital signature by setting **Digital Signature Delete**.

  This configures Access Manager rules to grant only owning user privilege to delete digital signature when the **Digital Signature Delete** rule is configured. This rule must be added before any rule that grants delete privileges to **POM_object** or **POM_application_object** business objects.



You can configure Teamcenter to block all the users or invalid digital signature owners from reapplying a digital signature if the target is set to **Invalid** on the business requirement. Use AM rules to prevent

reapplication of a digital signature on **POM_application_object** business objects with the **User Has Digital Signature** condition set to **Invalid**.



Step 8: Apply PKI digital signature to objects in workflow

The following workflow handlers are used for PKI digital signature:

- **EPM-apply-digital-signature**

  Applies the PKI digital signature of the logged-on user to the target objects and, optionally, the schedule task.

- **EPM-request-PKI-authentication**

  Displays a PKI authentication box in the **Perform** dialog box or pane of the task within which it has been placed. Users must type their PKI PIN in the box before the task can be completed.

- **EPM-verify-digital-signature**

  Verifies if the target objects and, optionally, the schedule task have a valid PKI digital signature.

If you want users to apply their PKI digital signature to objects in workflow, place the **EPM-apply-digital-signature** handler on an interactive workflow task. Because PKI digital signature requires users to use their PKI card, you must use the **EPM-request-PKI-authentication** handler to display the PKI authentication box in the **Perform** dialog box or pane of the task within which it has been placed.

If you want to check for valid PKI digital signatures during the workflow, place the **EPM-verify-digital-signature** handler on a workflow task, such as a **Validate** task.

Step 9: Add property to property policy file

To get the best performance, add the **Fnd0DigitalSignatureRel** property on the object in the property policy file.

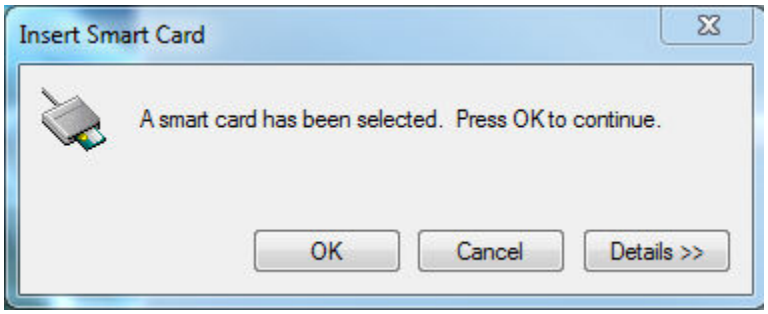
Using PKI digital signature

Using your smart card

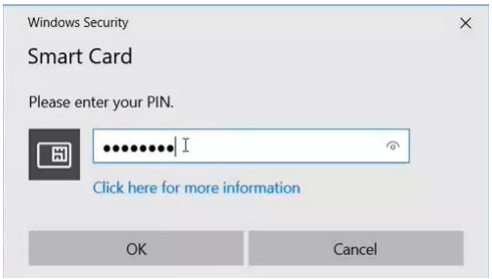
> Note:
>
> When you are required to insert your smart card can vary, either when you log on to Teamcenter or before you apply a PKI digital signature.

Before you can apply or remove a PKI digital signature, you must insert a smart card, which uses a digital certificate in a public key infrastructure (PKI) for authentication.

After you insert your PKI card, you are prompted to click **OK**.



Then, enter your PIN and click **OK**.



Applying a PKI digital signature

There are two states of PKI digital signatures that you can apply using the rich client:

- Valid

  Indicates the key used in the digital signature computes correctly indicating there has not been any modification.

- Void

  Indicates the workspace object that was once digitally signed must be modified. A void action on a digital signature always contains the full traceability of who signed, and who voided and when, ensuring full legal traceability on the object to the point of the void.

In addition, users can delete a digital signature they have applied.

A digital signature can be invalid if the key used in the digital signature does not compute correctly.

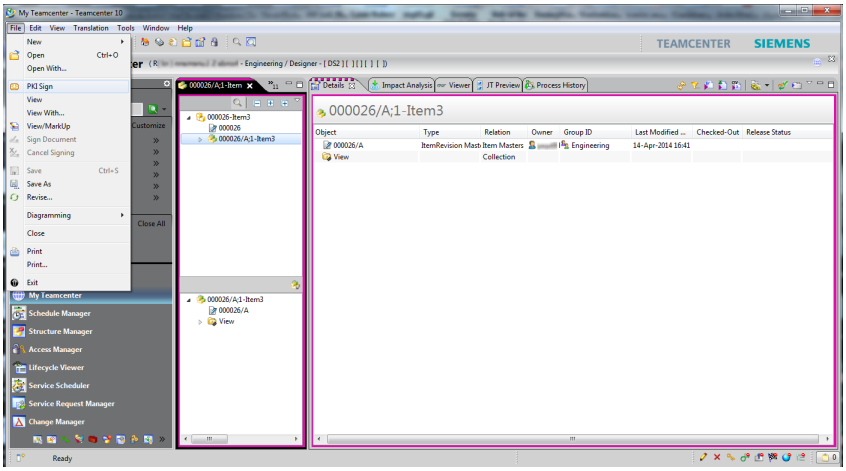
Apply a valid PKI digital signature

> Note:
>
> Do not apply a digital signature on ImanFile if you plan to transfer this object to a remote site.

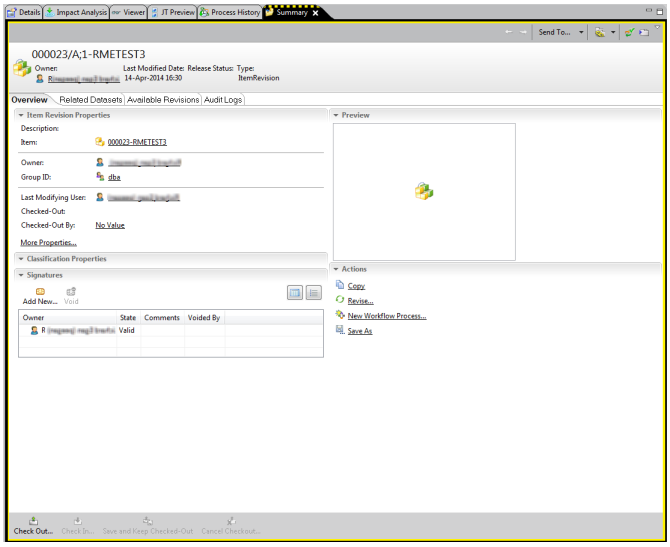
You can directly apply a PKI digital signature directly on a workspace object (if so configured) to establish authenticity and integrity between data and users. The PKI digital signature acts as a stamp on a workspace object showing that it has not been modified accidentally or maliciously.

> Note:
>
> A digital signature does not signify a user's approval of an object; it is intended only as a mathematical check on the state of the data entered.

- To apply a valid PKI digital signature, choose **File→PKI Sign**.



A valid PKI digital signature icon is applied.

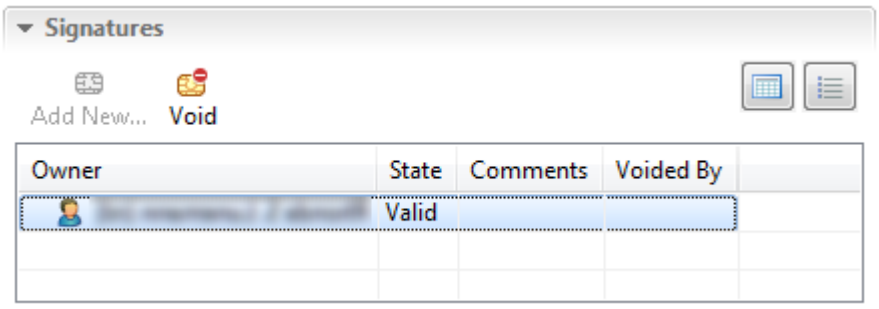


Void a PKI digital signature

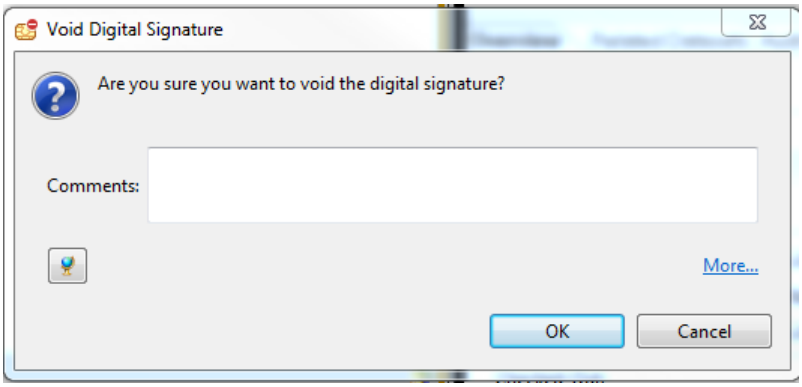
Before making changes to an object containing PKI digital signatures, you must void all valid digital signatures on the workspace object. Otherwise, your modifications to the workspace object cause the digital signatures to become invalid.

A void action on a digital signature always contains the full traceability of who signed, and who voided and when, ensuring full legal traceability on the object to the point of the void.

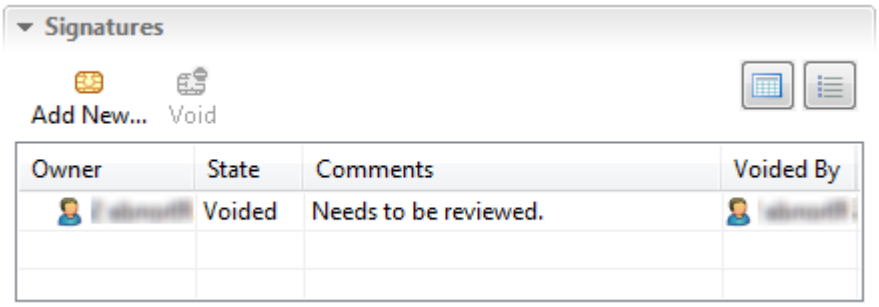
1. Select the valid signature that you want to void and click the **Void** icon.



2. In the **Void Digital Signature** dialog box, type your comments as to why you want to void the digital signature and click **OK**.



Your comments can now be viewed by others.



Delete a PKI digital signature

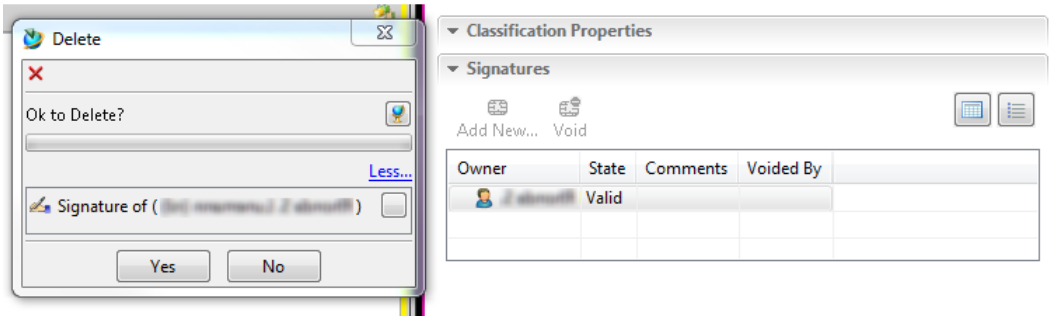
You can only delete a digital signature on a workspace object in the rich client if you have delete privilege.

Only the owning user can delete a PKI digital signature.

1. Select the valid signature.

2. Click **Delete**.

   This displays the **Delete** dialog box to confirm you want to delete the selected digital signature.



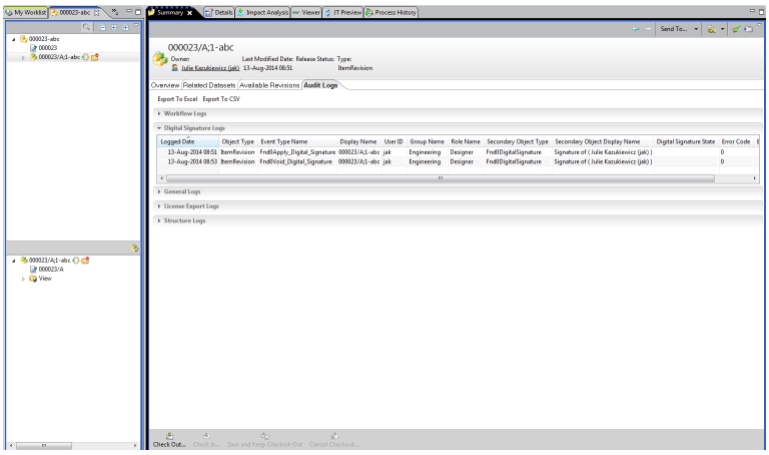
3. Click **Yes** to delete the digital signature from the workspace object.

View digital signature audit logs

1. Select a workspace object.

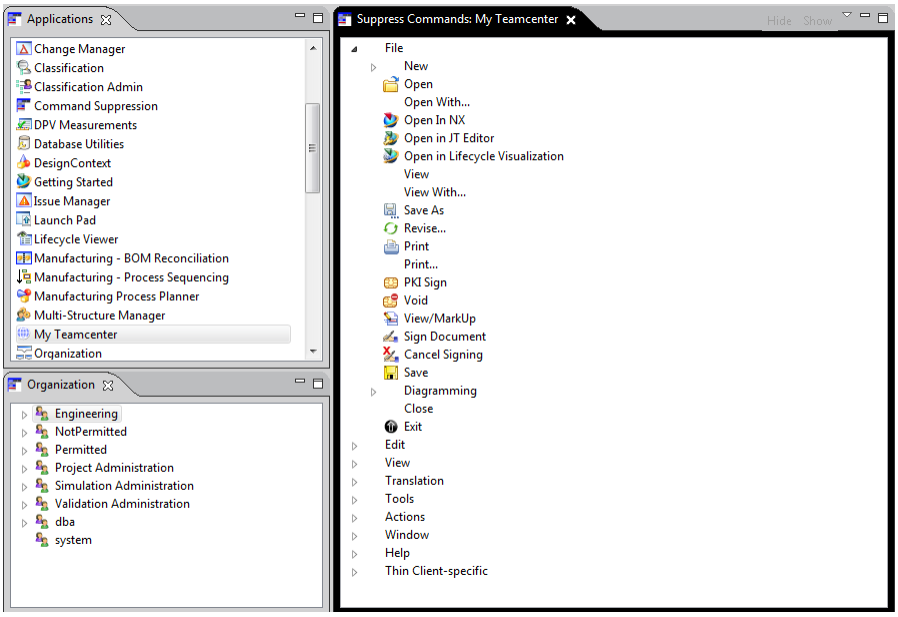
2. Select the **Summary** view.

3. Click the **Audit Log** tab.



Suppress digital signature commands

You can turn off the display of digital signatures from the menu by using Command Suppression.



Note:

You cannot suppress the **Delete** command only for digital signature objects. However, you can use Access Manager to control the deletion of digital signatures.

1. Select the My Teamcenter application from the **Applications** tree.
2. Select a role, subgroup, or group (for example, **Engineering**) from the **Organization** tree.
3. Select the menu, submenu, or command (for example, **PKI Sign**) from the **Suppress Commands** tree.
4. Click **Hide**.

   A red line appears on the selected menu, submenu, or command node selected for suppression.
5. Choose **File→Close** to save the changes.

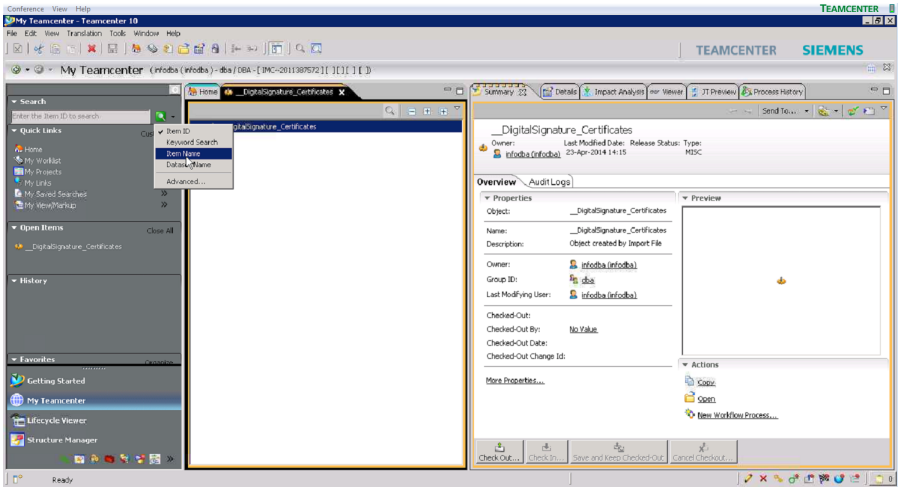
Troubleshooting digital signature

What do I do if my certificate expires or I have a new certificate?

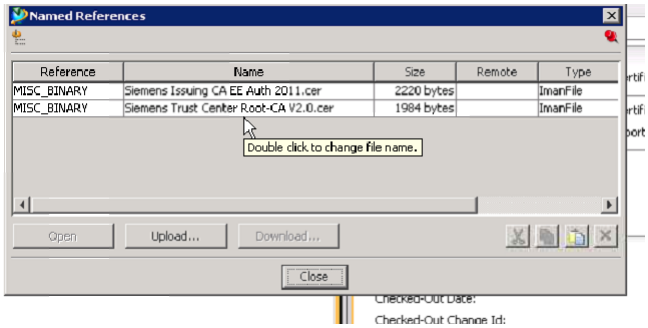
Note:

Use this method to update your certificate authority file information when your certificate expires or if you have a new certificate.

1. Search for **__DigitalSignature_Certificates** dataset (note the leading double underscore).



2. Check out and modify the dataset as needed.



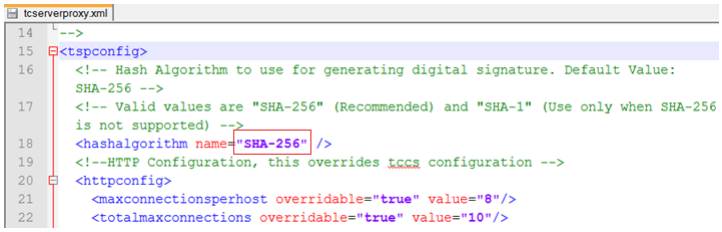
How do I customize a hash algorithm for digital signature?

By default, the Teamcenter client communication system (TCCS) component uses the SHA-256 hash algorithm. However, you can change this algorithm to match your need.

1. Open your configuration file. For example:

   ```
   %USERPROFILE%\Siemens\cfg\tccs\Teamcenter\tcserverproxy.xml
   ```

2. Edit the hash algorithm tag value to include the appropriate hash algorithm, such as **SHA-256** or **SHA-384** or **SHA-512** or **SHA-1**.



3. Save your changes.

15. Controlling access to classification objects

Classification access control overview

The Classification Admin access control feature allows you to control access to Classification objects (ICOs) and hierarchy components (groups and classes). This feature is an extension of the Access Manager (AM) application and employs the AM tree of rules and access permissions to define access to objects. Rules created for Classification groups and classes are inserted into branches of the rule tree.

The basic concepts of access control are the same for both the Classification Admin and Access Manager applications.

> Caution:
>
> To maintain consistency, Classification rules should not be edited in the Access Manager application. Additionally, the Classification access control feature and the Access Manager application cannot be used simultaneously.

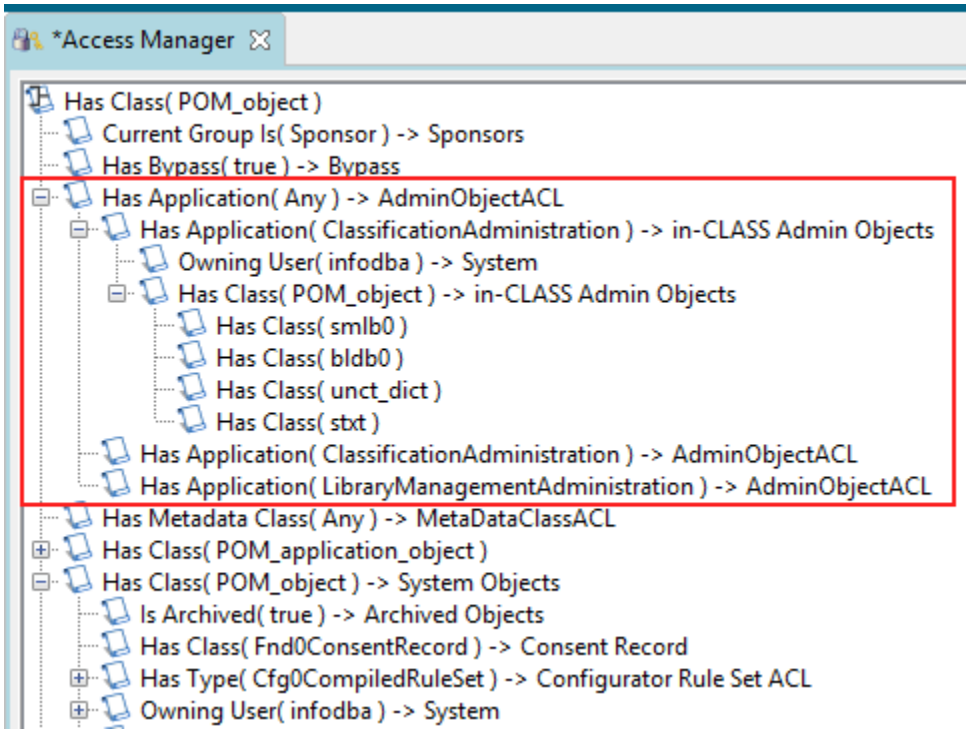
You can suppress the display of individual groups and classes in the hierarchy based on group, role, or user-specific controls. This enables you to customize the display of the Classification hierarchy tree, providing users with only the Classification data that is relevant to their tasks. Component display suppression affects the display of the hierarchy tree in both the Classification and Classification Admin applications. When visibility of a hierarchy component is suppressed, the display of the component's children is also suppressed.

Creation and modification of groups, classes, and subclasses is controlled, enabling different individuals to be responsible for maintaining different parts of the hierarchy.

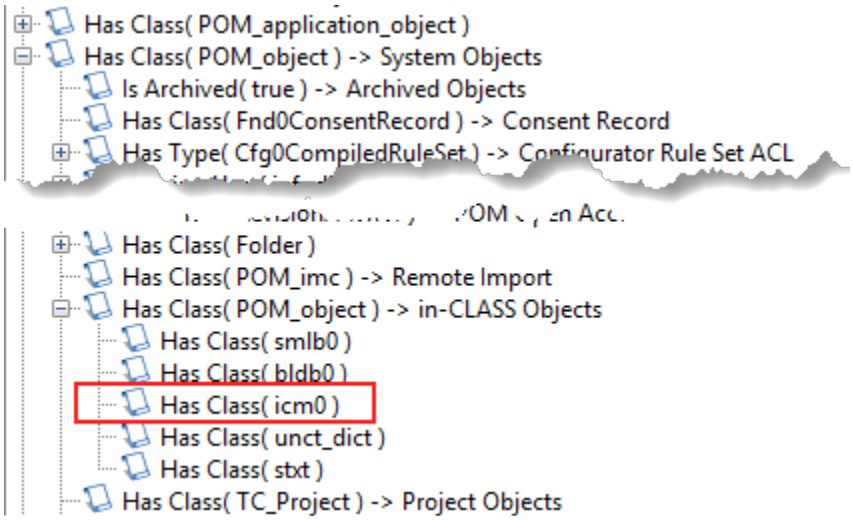
The following restrictions apply:

- The IDs of classes and groups to which you apply privileges must not contain spaces.
- To maintain consistency, Classification rules should not be edited in the Access Manager application. Additionally, the Classification access control feature and the Access Manager application should not be used simultaneously.
- Only members of the system administrator group can modify access privileges.

All access rules created in the Classification Admin application except those pertaining to classification objects (ICOs) are stored in the following location in the Access Manager rule tree:



Access rights to stand-alone ICOs are stored in the following location:



Caution:

Do not manually create access rules for administrative objects in the **Has Class(POM_object)** group as they are overwritten by entries higher in the tree.

Component display suppression

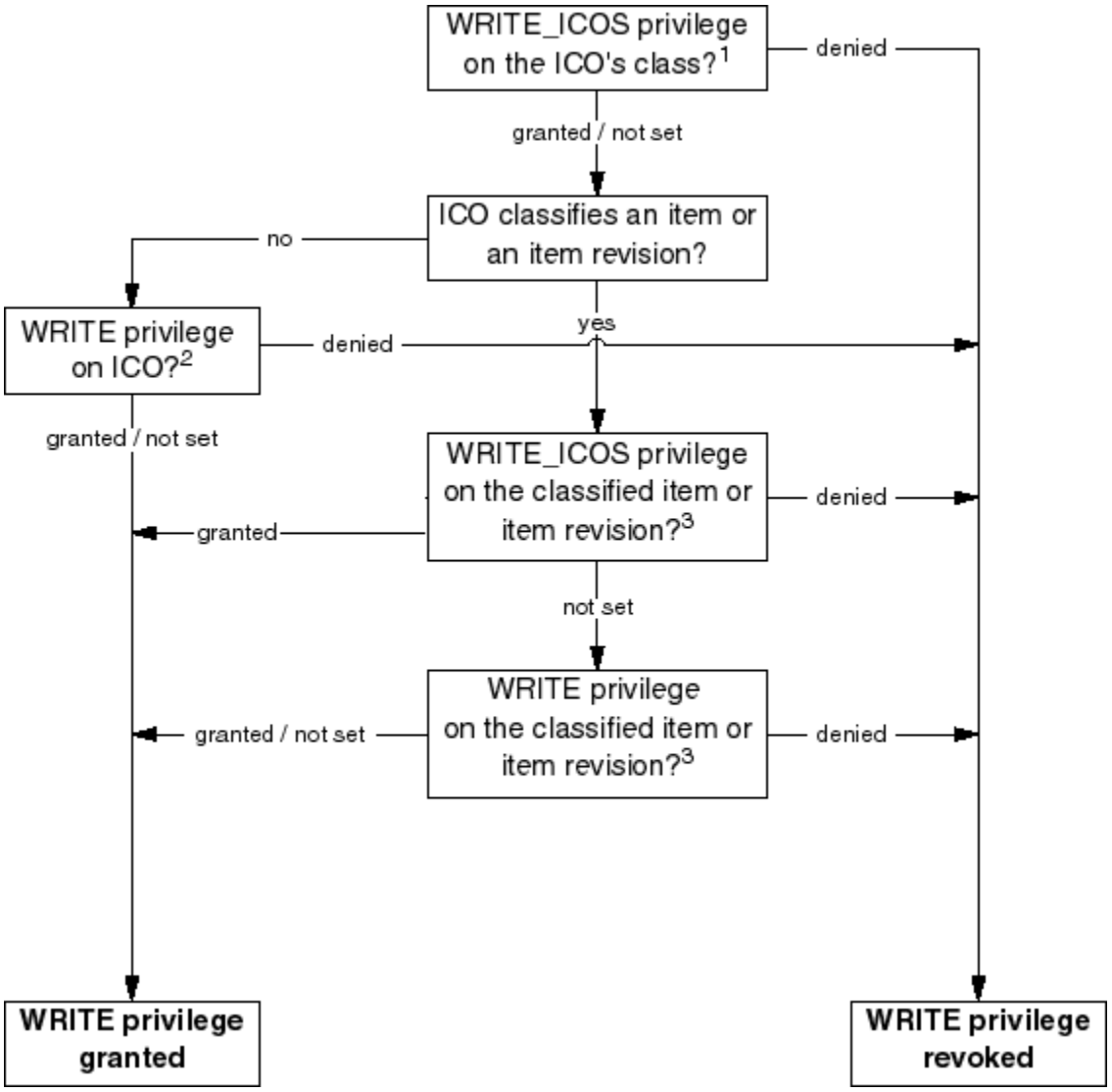
You can suppress the display of individual groups and classes in the hierarchy based on group, role, or user-specific controls. This enables you to customize the display of the hierarchy tree, providing users with only the Classification data that is relevant to their tasks. Component display suppression affects the display of the hierarchy tree in both the Classification and Classification Admin applications. When visibility of a hierarchy component is suppressed, the display of the component's children is also suppressed.

Hierarchy component protection

Creation and modification of groups, classes, and subclasses is controlled, enabling different individuals to be responsible for maintaining different parts of the hierarchy.

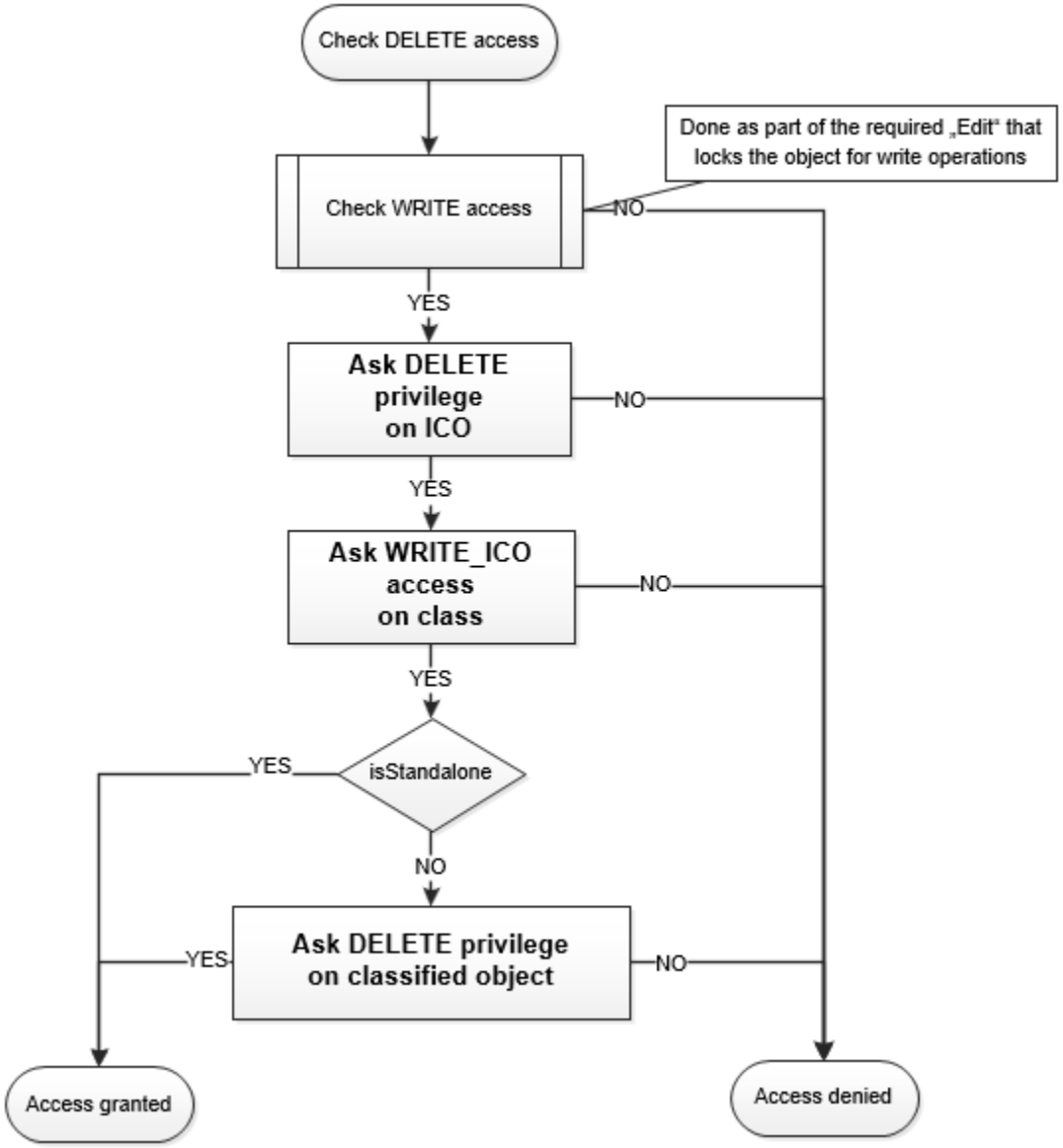
ICO protection

Privileges applied to groups and classes determine which user, or groups of users, can view, add, or modify the Classification objects (ICOs) associated with the group or class. The following figure illustrates how privileges are evaluated to determine if a user has the privileges required to update an ICO.



1. Set in Classification Admin, **privileges on class definition**.
2. Set in Classification Admin, **privileges on ICOs**.
3. Set in Access Manager application or as **Object ACL** on the workspace object (for example, in My Teamcenter).

The following figure illustrates how privileges are evaluated to determine if a user has the privileges required to delete an ICO.



Restrictions

- The IDs of classes and groups to which you apply privileges must not contain spaces.
- To maintain consistency, Classification rules should not be edited in the Access Manager application. Additionally, the Classification access control feature and the Access Manager application should not be used simultaneously.

- Because you achieve access control in Classification by adding named ACLs to the Access Manager rule tree, only members of the system administrator group can modify access privileges.

Classification access privileges

The following table describes how access privileges apply to Classification objects, and whether the privileges are inherited by children of the selected object within the hierarchy.

Privilege	Used by Classification access control	Purpose	Inherited?
Read (R)	Yes	Controls visibility of a group or class in the hierarchy tree. When read access is denied, the object is not displayed in the tree. This privilege overrides privileges set for the Classification objects (ICOs) of a class. If read privileges are denied at the class level, but granted for the ICOs of the class, the ICOs are inaccessible. Note: Revoking the read privilege cannot be overridden in a subclass; however, granting the Read privilege can be overridden in a subclass.	Yes
Write (W)	Yes	Controls whether a group or class can be modified, and when applied to a class, controls whether subclasses and views can be added to the class.	Yes
Delete (D)	Yes	Controls whether a group or class can be deleted from the hierarchy. Restrictions on deleting groups and classes may prevent you from deleting an object to which you have delete privileges. For example, you cannot delete a class that has been referenced, regardless of the privileges granted.	Yes
Change (C)	Yes	Controls the right to define access control privileges.	Yes
Promote (p)	No	Not applicable.	Not applicable.
Demote (d)	No	Not applicable.	Not applicable.
Copy (c)	No	Not applicable.	Not applicable.
Export (X)	No	Not applicable.	Not applicable.
Import (I)	No	Not applicable.	Not applicable.
Transfer-out (x)	No	Not applicable.	Not applicable.
Transfer-in (i)	Yes	Controls whether ownership of an object can be transferred from one site to another.	No
Change Ownership	No	Grants or revokes the privilege to select shared sites for sharing classification data in Multi-Site Collaboration.	No
Publish	Yes	Controls whether the group or class and its children can be shared to other sites. Additionally, if this privilege is denied, the user will	No

Privilege	Used by Classification access control	Purpose	Inherited?
		not be able to modify the list of shared sites in the class or group definition.	
Subscribe	No	Not applicable.	Not applicable.
Write ICOs	Yes	Controls whether objects can be classified and stored within a class. Also controls whether existing Classification objects (ICOs) can be modified. Attributes of the ICOs associated with part family members cannot be modified unless write access is granted to the part family template.	Yes

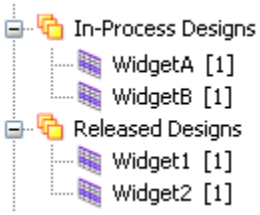
Note:

ICOs that classify workspace objects are subject to further restrictions.

Applying access controls examples

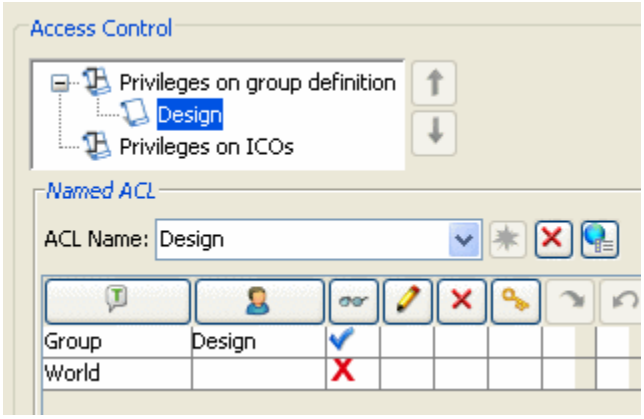
Example: controlling the display of the hierarchy tree for Classification users

ABC Corporation manufactures widgets and uses Classification to classify their design data, using the following hierarchy structure.



In-process designs are considered to be strictly confidential and only the Design work group is allowed to view them prior to release.

To suppress the display of this Classification hierarchy data for all users except those in the Design work group, protections are applied to the In-Process Designs Classification group, as shown next.

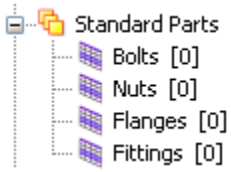


By granting read privileges to users in the Design work group and denying read privileges to the world, the data in the In-Process Design Classification group and all of its child classes are only visible to users who have a role in the Design work group. In addition, the protected objects are only returned as query results to users in the Design work group.

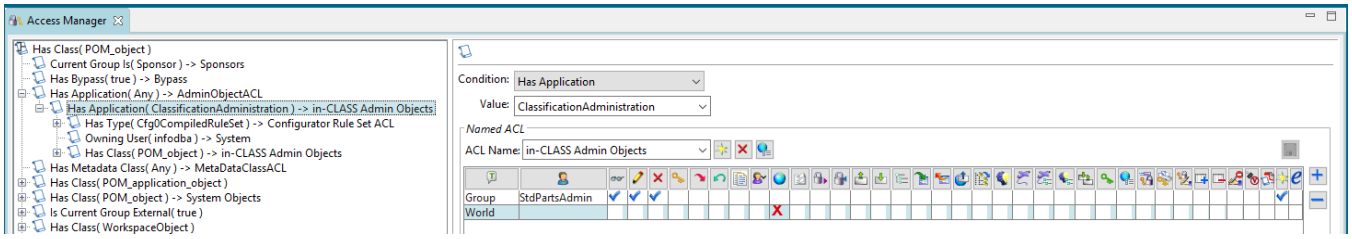
> Note:
>
> Read privileges to child classes of a read-protected Classification group cannot be granted unless the user is one to whom read privileges are also granted at the Classification group level.
>
> For example, John Smith, a member of the Marketing work group, cannot be granted read privileges to the Widget A class because it is a child class of the In-Process Design group. As a child, it inherits the privileges of the parent group, and according to the rule defined in the figure above, only users with a role in the Design work group can be granted read privileges to the Widget A class. If, however, John Smith maintained dual roles in both the Marketing and Design work groups, he could be granted read privileges on the Widget A class.

Example: controlling access to hierarchy definitions

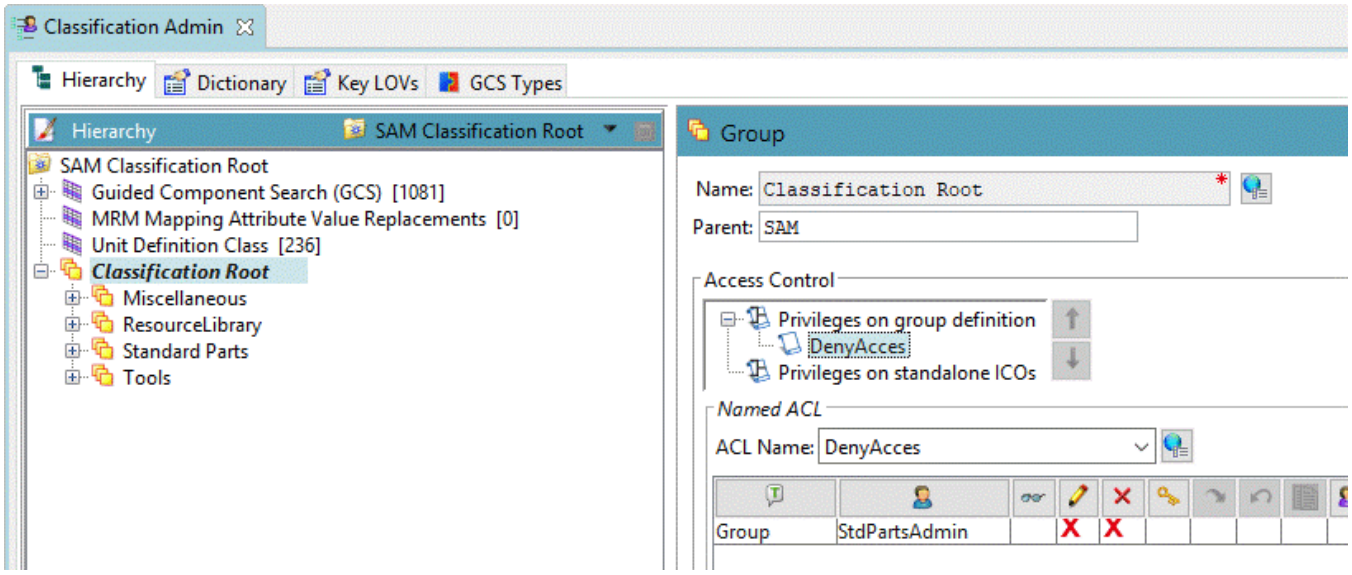
ABC Corporation also maintains a library of standard parts that are accessible to users throughout the organization. These parts are classified according to a hierarchy that includes storage classes for different types of parts.



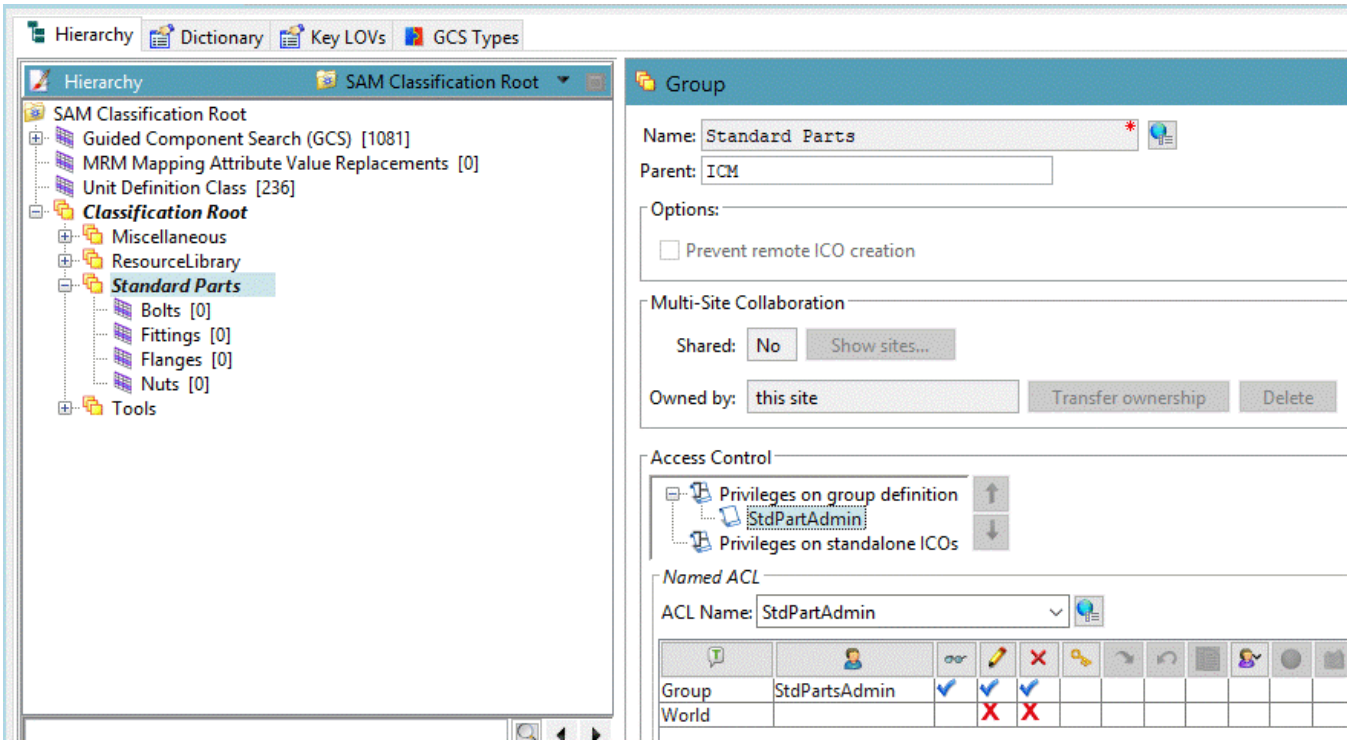
Only users in the **StdPartsAdmin** work group are allowed to perform maintenance tasks on this portion of the hierarchy (beginning with the Standard Parts group as the root). By default, access to classification administration objects is denied to non-DBA users. To grant a group of non-DBA users, the group must first be granted access to classification administration objects in the rule tree.



Now, the **StdPartsAdmin** group has access to the complete classification hierarchy. This must be restricted to the **StdPartsAdmin** work group only. Although there may be different approaches to assigning access rights depending on your business needs, this example begins by restricting access to the entire hierarchy in **Classification Admin**.



Subsequently, the **StdPartsAdmin** group is granted access to the **StdPartsAdmin** branch of the hierarchy only.



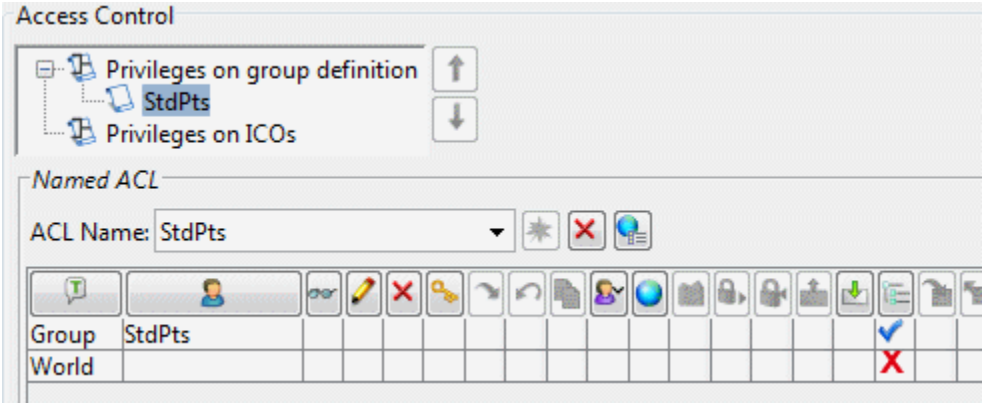
By granting write privileges to users in the **StdPartsAdmin** work group, the definitions of the **Standard Parts** classification group and all of its child classes are only modifiable by users who belong to the **StdPartsAdmin** work group.

> Note:
>
> Write privileges can be granted on child classes of a write-protected group.

Example: controlling access to ICOs

Classifying the standard parts used to produce various models of widgets is a task assigned to the **StdPts** work group at ABC Corporation. While other work groups within the organization must be able to view the data associated with the Classification objects (ICOs), such as the physical attributes of the part or cost data, only the **StdPts** group is allowed to classify new parts or modify the attribute values of existing part classifications.

To restrict write access to standard part ICOs for all users except those in the **StdPts** work group, protections can be applied to the Standard Parts Classification group, as shown below.



By granting write privileges on the ICOs that are created under the Standard Parts group to users in the **StdPts** work group and denying those privileges to the world, the ability to create and modify ICOs is restricted to the **StdPts** group. However, all other users are able to view the ICOs and their data.

> Note:
>
> Creating an access rule under **Privileges on ICOs** controls access to stand-alone (nonclassifying) ICOs only.

Creating access rules

Introduction to creating access rules

Access Manager uses various conditions, or rules, to control and protect your data. These rules are global; they affect your entire Teamcenter site. Rules that apply to Classification data specify a named access control list (ACL) that is applied to the object.

The AM rule tree displays the rules in force at your site. Each rule is assigned some level of relative importance. The rules near the top of the tree take precedence over rules lower in the tree. The Classification Admin **Access Control** pane displays only the portion of the tree that applies to Classification objects.

Create a classification access rule

To control and protect your data,various conditions or rules are applied. These rules are global, they affect your entire Teamcenter site. Rules that apply to Classification data specify a named access control list (ACL) that is applied to the object.

The AM rule tree displays the rules in force at your site. Each rule is assigned some level of relative importance. The rules near the top of the tree take precedence over rules lower in the tree. The Classification Admin **Access Control** pane displays only the portion of the tree that applies to Classification objects.

1. Click the **Hierarchy** tab.

Teamcenter displays the **Hierarchy** pane.

2. Choose the group or class in the hierarchy tree that is affected by the rule.

   Teamcenter displays the definition pane for the group or class.

3. Click the **Edit** button on the toolbar to activate the definition pane.

4. If defining rules for a class, click the **Access Control** tab to display the **Access Control** pane. When working with groups, the **Access Control** pane is displayed on the definition pane when you select the group.

   The **Access Control** pane displays the two AM rule tree roots related to Classification, along with the standard Teamcenter **Named ACL** dialog box.

5. Choose the AM Rule tree root node that represents one of the following types of rule you want to define:

   - Privileges on class/group definition
   - Privileges on ICOs

   Privileges on class/group definition applies rules to the group or class and its descendants. Privileges on ICOs applies controls to the ICOs that are created within the group or class.

6. Choose a named ACL from the list or **create a new ACL**.

   Teamcenter displays the access control entries (ACEs) that comprise the named ACL in the table.

7. **(Optional) Modify the access control entries.**

8. Click the **Add** button located at the bottom of the **Access Control** pane. Teamcenter adds the ACL to the rule tree.

9. Order the rules in the tree, as required, by using the up-arrow and down-arrow buttons next to the tree. These rules are evaluated in order from top to bottom when a user attempts to access an object. Thus, a rule directly beneath the root takes precedence over one further down the tree.

10. Click **Save** on the toolbar to save the new rule.

Create a named access control list (ACL)

> Note:
>
> You must be in edit mode to create named ACLs.

1. Type a name for the new ACL in the **ACL Name** box.
2. Click the **Create** button located next to the **ACL Name** box.

   Teamcenter creates the ACL. However, there are no entries associated with it.
3. Click the **Add New ACL** button .

   A blank line appears in the ACL table.
4. Double-click in a blank cell in the **Type of Accessor** column to display a list of predefined accessor types.
5. Select the accessor type that you want to use for this entry.
6. Double-click in a blank cell in the **ID of Accessor** column to display the **Select Accessor** dialog box. This dialog box contains a list of predefined roles corresponding to the type of accessor you selected in step 4.
7. Double-click the role that you want to apply to the accessor. You can also select the role in the dialog box and clicking **OK**.

   Teamcenter displays the role of the accessor you selected in the **ID of Accessor** column.
8. Define privileges for the accessor by double-clicking in the **Privilege** column and choosing one of the following options:

   - Grant privilege
   - Deny privilege

   Blank entries are also valid. Using blank entries enables rules to accomplish focused objectives by allowing objects and accessors to *fall through* rules that do not apply to them.
9. To add additional entries to the named ACL, repeat steps 1 through 7.
10. Click **Save** located to the upper right of the ACL table.

Modify access control list entries

The following task is performed in the **Access Control** pane in Classification Admin.

> Note:
>
> You must be in edit mode to modify named ACLs.

1. Choose the named ACL you want to change from the **Named ACL** list.

   Teamcenter displays the details of the ACL in the table.

2. Modify privileges by double-clicking the column corresponding to the privilege and choosing one of the following options:

   ✔ Grant privilege

   ✘ Deny privilege

   Blank entries are also valid. Using blank entries enables rules to accomplish focused objectives by allowing objects and accessors to *fall through* rules that do not apply to them.

3. Repeat steps 1 and 2 until all desired privileges have been granted or denied for this ACL.

4. Click the **Modify** button located at the bottom of the **Access Control** pane.

5. Click **Save** located to the upper right of the ACL table.

Delete access rules

The following task is performed in the **Access Control** pane in Classification Admin.

Access control rules can be removed from the rule tree and deleted from the database.

> Note:
>
> You must be in edit mode to delete access rules.

1. Click the **Hierarchy** tab. Teamcenter displays the **Hierarchy** pane.

2. Choose the group or class in the hierarchy tree that is affected by the rule.

   Teamcenter displays the definition pane for the group or class.

3. Click the **Edit** button on the toolbar to activate the definition pane.

4. If you are deleting rules relative to a class, click the **Access Control** tab to display the **Access Control** pane. When working with groups, the **Access Control** pane is displayed on the definition pane when you select the group.

The **Access Control** pane displays the two AM rule tree roots related to Classification, along with the standard Teamcenter **Named ACL** dialog box.

5. Choose the rule in the tree that you want to delete.

6. Click the **Delete** button ✕ located at the bottom of the **Access Control** pane. The rule is removed from the tree.

7. Click **Save** on the toolbar to save the change and delete the rule from the database.

16. Controlling access based on compound property values

About controlling access based on compound property values

A compound property is a property on a business object that can be displayed as a property of an object (the display object) although it is defined and resides on a different object (the source object). Use the **Has Property** condition to control access privileges based on compound property values.

Access Manager derives the value of compound properties on a given target object from the attribute values on one or more secondary objects (auxiliary objects). The **Has Property** condition loads those secondary objects to retrieve the given compound property value. If while retrieving the compound value an error occurs because there is no read access on any secondary object or for similar reasons, the rule evaluation fails and the rule is not applied to the target object.

Has Property works with types and compound properties. Configuring rules based on compound properties whose values come from custom objects can only be done using the **Has Property** condition.

Note:

- Do not use **Has Attribute** conditions against custom properties. **Has Attribute** only works with classes and their persistent attributes. For example:

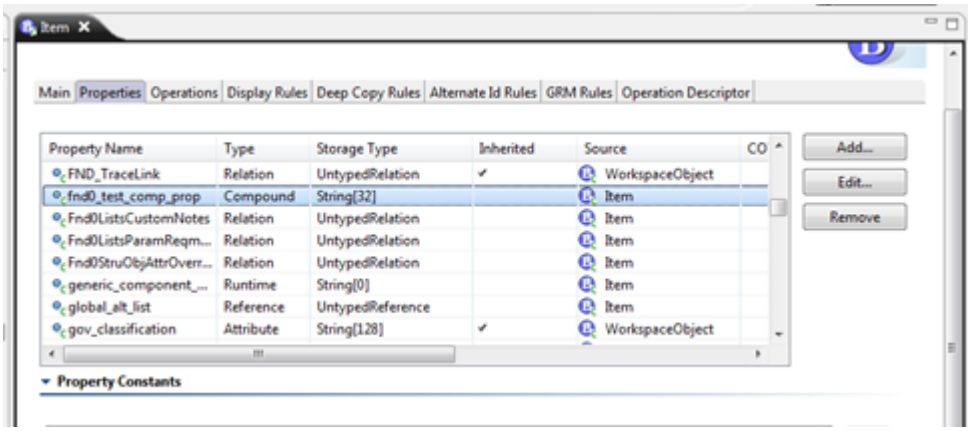
  ```
  Has Attribute (Item:item_id=000013)
  ```

  In this example, `Item` is the class and `item_id` is the attribute on the item class.

- When creating rules containing both the **Has Attribute** and **Has Property** conditions always use valid class and type names, respectively. If you use invalid class or type names, the rule evaluation fails and the predicted access control behavior is not achieved.

Has Property condition example

1. Using the Business Modeler IDE, create a compound property. In this case, the **fnd0_test_comp_prop** compound property was added to the item with a string property value.



2. Using the Access Manager, create a rule as given below.

For the option	Do the following
Condition	Select **Has Property**.
Value	Type **Item:fnd0_test_comp_prop=test value**
ACL Name	Select an existing named ACL or create and select a new named ACL. For example **CompoundACL** as given below.

The **CompoundACL** ACL grants privileges to **Owning group** and **World** as follows.

Owning group		✓	✓	✓	✓	✓					
World		✗									

The **Owning group** is explicitly granted read, write, delete, change, and demote privileges.

The **World** accessor is explicitly denied read privileges.

17. Configuring Oracle PKI

Configure Oracle Wallet

There are two components to Oracle security:

- Administration of public key infrastructure (PKI) certificates and requests.
- Configuration of Oracle network layer to authenticate using an external PKI method.

As an administrator. you can use the Oracle Wallet application to administer PKI certificates and requests. While the graphical Oracle Wallet Manager (owm) can be used to administer the wallet, it is simpler and clearer to use the command line tool, orapki, for these purposes.

The following components are required to use Oracle Wallet to authenticate using PKI:

- A trusted root certificate used for signing server and user certificates.

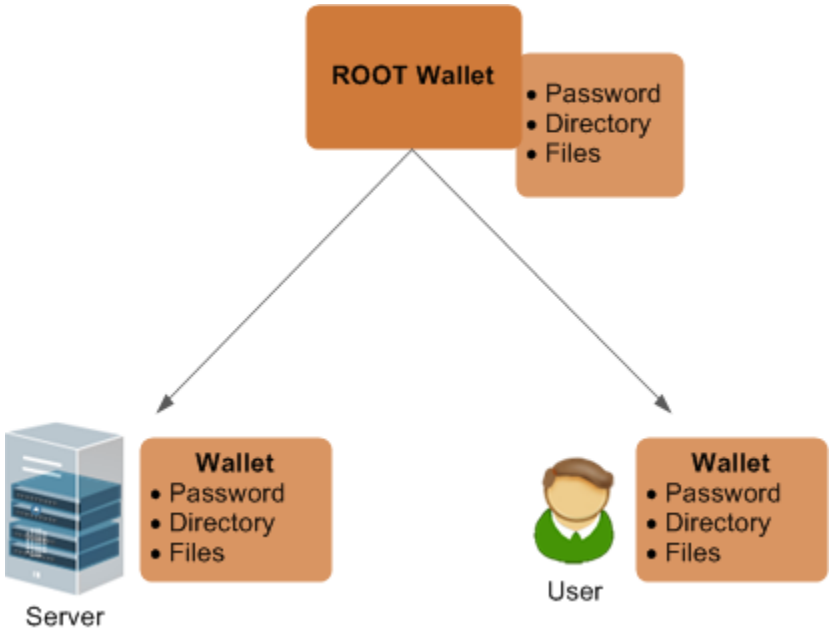
  This involves installation with optional creation.

- A PKI certificate for the Oracle server.

  This involves creation, signing, and installation.

- A PKI certificate for each user requiring authentication in the database.

  This involves creation, signing, and installation.



Each of these components requires its own secure repository, called a wallet. Each wallet has its own password, directory, and files. Thus, three wallets will be required. These can be on separate systems, or the same system.

In these examples, all wallets are assumed to be stored under the **$ORACLE_HOME/wallet**. On the system used in the examples, this is **D:\ORACLE\WALLET**.

> Note:
>
> All commands should be entered on a single line unless otherwise noted.

Create the root wallet and trusted certificate

The trusted root certificate signs all user and database certificates. This root certificate may come from a certificate authority (CA) or be self-signed.

> Note:
>
> The following examples use a self-signed certificate.

1. Create the root wallet.

   ```
   orapki wallet create -wallet d:\oracle\wallet\root
   ```

   You are prompted for a wallet password, which is required for any manipulation of the wallet.

2. Create a new self-signed root certificate in the wallet.

```
orapki wallet add -wallet d:\oracle\wallet\root -dn
"CN=root,O=Devco,
      OU=Devco Department,C=US" -keysize 2048  -self_signed
-validity 365
```

Adjust the distinguished name (DN) as needed.

3. Export the self-signed certificate, which will be loaded in the other wallets in later steps.

```
orapki wallet export -wallet d:\oracle\wallet\root -dn
"CN=root,O=Devco,
      OU=Devco Department,C=US" -cert d:\oracle\wallet\root\root.cer
```

Create the database server wallet and its certification

1. Create the database server wallet, specified here as **auto_login**.

```
orapki wallet create -wallet d:\oracle\wallet\db
      -auto_login
```

As in the previous section, you will be prompted for a password when creating the wallet. You will use this password on all further access to the wallet.

2. Add the trusted certificate, which you created in *Create the Root Wallet and Trusted Certificate*, to the database server wallet.

```
orapki wallet add -wallet d:\oracle\wallet\db
      -trusted_cert -cert d:\oracle\wallet\root\root.cer
```

You will be prompted for the database server wallet password.

3. Initiate the creation of a certificate for the database server. It will be incomplete until signed.

```
orapki wallet add -wallet d:\oracle\wallet\db  -dn
      "CN=TC12,DC=oracle,dc=local" -keysize 1024
```

4. Export the unsigned database server certificate as a request for signature. The request is placed in the **dbcert.req** file.

```
orapki wallet export -wallet d:\oracle\wallet\db
      -dn "CN=TC12,DC=oracle,dc=local"
      -request d:\oracle\wallet\db\dbcert.req
```

5. Sign the request with the root trusted certificate, generating a certificate in the **dbcert.req** file.

```
orapki cert create -wallet d:\oracle\wallet\root
      -request d:\oracle\wallet\db\dbcert.req
      -cert d:\oracle\wallet\db\dbcert.cer -validity 365
```

6. Import the signed certificate (in **dbcert.cer**) into the database wallet.

```
orapki wallet add -wallet d:\oracle\wallet\db
      -user_cert -cert d:\oracle\wallet\db\dbcert.cer
```

Create the user wallet and its certificates

1. Create the user wallet on the system where Teamcenter will be running.

```
orapki wallet create -wallet d:\oracle\wallet\user
      -auto_login
```

   As with the database server wallet, this one will be specified as **auto_login**. Likewise, a wallet password is required for any wallet maintenance.

2. Add the trusted certificate from the first section to the user wallet. You will be prompted for the user wallet password.

```
orapki wallet add -wallet d:\oracle\wallet\user
      -trusted_cert -cert d:\oracle\wallet\root\root.cer
```

   You may need to copy the **root.cer** file to a location on this system; adjust the path accordingly.

3. Initiate the creation of a certificate for the user. (Steps 3-6 will be required for each database user requiring authentication.)

```
orapki wallet add -wallet d:\oracle\wallet\user
      -dn "CN=Tc-admin-user,DC=oracle,dc=local" -keysize 1024
```

   Within the distinguished name (DN), CN is the user name. The certificate is incomplete until signed.

4. Export this unsigned certificate as a request for signature.

```
orapki wallet export -wallet d:\oracle\wallet\db
      -dn "CN=Tc-admin-user,DC=oracle,dc=local"
      -request d:\oracle\wallet\db\usercert.req
```

   The request is placed in the **usercert.req** file.

5. Sign the request with the root trusted certificate.

```
orapki cert create -wallet d:\oracle\wallet\root
      -request d:\oracle\wallet\user\usercert.req
      -cert d:\oracle\wallet\user\usercert.cer -validity 365
```

If the root wallet is on another system, you must transfer the **usercert.req** request file to that system, with path adjusted accordingly, and the resulting **usercert.cer** file moved back to the user system.

6. Import the signed certificate (in **dbcert.cer**) into the database wallet.

```
orapki wallet add -wallet d:\oracle\wallet\user -user_cert
      -cert d:\oracle\wallet\user\usercert.cer
```

A. Rule conditions, accessor types, and privileges

What are access rules composed of?

Access rules are composed of conditions, accessor types, and privileges. Teamcenter provides a tree of default rules, as well as accessor types and privileges that can be used to configure rules that grant or deny object access for users, groups of users, or project teams.

Access conditions by group

The following table lists the access conditions by category. Click a condition to learn more about it.

Condition	Description
Administrative	
Has Application	Provides additional security to administration applications, for example, Organization, Access Manager, and Authorization.
Has Bypass	Specifies whether the user has bypass privileges set. Bypass privilege supersedes other privileges. This privilege allows administrators to make changes that could potentially cause unintended loss of data and have serious repercussions that are normally guarded against by access rules.
Has Metadata Class	Provides additional security to property conditions and other metadata.
General	
Has Attribute	Specifies an attribute and value associated with a particular class.
Has Class	Specifies an object class. The object is evaluated to determine if it is of the specified class.
Has Classification	Validates the custom classification attribute value of the object against the value specified for the condition.
Has Description	Specifies a description for the object. The object is evaluated to determine whether the description matches this value.
Has Digital Signature	Specifies whether a business object has a digital signature of the specified status.
Has Form Attribute	Enables access control of items and item revisions by setting conditions on attributes of the **Masterform** class.
Has Item ID	Specifies an item ID against which the item is evaluated.
Has Item Key	Specifies a multifield key identifier against which the item is evaluated.
Has Name	Specifies a name against which the object is evaluated.
Has Object ACL	Specifies that an ACL is associated with an object. This condition does not expect an ACL attached to a rule. It is a placeholder that indicates the point at which process ACLs and object ACLs are applied in the rule tree hierarchy.

Condition	Description
Has Property	Specifies the value of a compound property against which an object is evaluated.
Has Status	Specifies the status type against which the object is evaluated.
Has Type	Specifies the object type against which the object is evaluated.
Inactive Sequence	Specifies that previous sequences are historical and cannot be worked on independently. The latest sequence is always the working sequence for the revision. Note: This condition is used in conjunction with the **Inactive Sequence Objects** ACL.
In Job	Specifies whether the target object is in a workflow job (process). This condition does not expect an ACL attached to a rule. It is a placeholder that indicates the point at which workflow ACLs are applied in the rule tree hierarchy. Note: No subbranches can be added below the **In Job** branch in the Access Manager rule tree.
Is Archived	Specifies that the object's archive status is evaluated.
Is Local	Specifies whether the object's residence in the local database is evaluated. This condition is used when Multi-Site Collaboration is implemented.
Is Sponsored Mode	Checks whether the Teamcenter session is in sponsored mode. It enables end users to configure rules to enforce data access control when the Teamcenter session is launched in sponsored mode.
Site Geography	Checks whether the given geography matches the geography of the site being evaluated.
User Has Digital Signature	Specifies whether a business object has a digital signature of the specified status in the context of the logged-on user.
Ownership/Accessor based	
Current Group Is	Checks the current logged-on group that is set in the session. It enables end users to configure access rules for the **Sponsor** group.
Is Current Group External	Evaluates whether the security of the current logged in group is external.
Is GA	Specifies whether the user's status as a group administrator in the current group is evaluated.
Is Group External	Evaluates whether the object under consideration is Group object and has external security.
Is Group Member External	Evaluates whether the object under consideration is GroupMember and belongs to a group that has external security.
Is Group Same As Current Group	Evaluates whether the object under consideration is Group and is the same as the current logged in group that has external security.
Is Member Group Same As Current Group	Evaluates whether the group member object belongs to the same group as the current logged on group.
Is SA	Specifies whether the user's system administration group membership is evaluated.

Condition	Description
Owning Group	Evaluates whether the object is owned by the group under which the user is logged on to Teamcenter.
Owning Group Has Security	Evaluates whether the owning group of the object has a security string. This condition is true only if the security value of the owning group is equal to the value of this condition.
Owning Site	Evaluates whether the object is owned by the specified site. This condition is used when Multi-Site Collaboration is implemented.
Owning User	Evaluates whether the object is owned by the specified user.
Is User External	Evaluates whether the user object is from a group whose security is external.
Is User In Current Group	Evaluates whether the user object under evaluation has current group membership.
Incremental Change	
In IC Context	Enables structure edits (occurrence edits, occurrence notes, transform edits, and attachment edits) to be controlled by the Structure Manager, Manufacturing Process Planner, Multi-Structure Manager, or Part Planner application.
Project	
In Current Project	Specifies the project ID against which the object is evaluated. Note: This rule is not delivered with the default installation of Teamcenter. It must be added manually.
In Project	Specifies a project to which the object must be assigned.
Is Project Member	Specifies whether the user's membership in the project is evaluated. This condition is only true when the user is a current member of the project.
Has Project Of Category	Checks whether the workspace object being evaluated has any project assigned of the given category.
Program	
In Current Program	Specifies access based on whether the program to which the data is assigned is the current program under which the user is logged on to Teamcenter.
In Inactive Program	Controls access to data based on whether the status of the owning program is **inactive**.
In Invisible Program	Controls access to data based on whether the status of the owning program is **invisible**.
Is Owned By Program	Controls access to data based on whether data is owned by the program specified as a value for the **Is Owned By Program** condition.
Is Program Member	Specifies whether the user's membership in the program is evaluated. This condition is only true when the user is a member of the owning program or a shared program.
General Authorized data access (ADA) licenses	
ADA License Has Citizenship	Checks whether the ADA license being evaluated has the given citizenship.
Citizenship On Any ADA Lic	Checks whether the citizenship of the user being evaluated matches any of the citizenships applied to the ADA licenses attached to the workspace objects.

Condition	Description
Has ADA License Of Category	Checks whether the workspace object being evaluated has any ADA license of the given category.
Has Named ADA License	Checks whether a specific ADA license is attached to the workspace objects being evaluated.
User In Attach ADA Lic of Ctgry	Checks whether the user being evaluated is listed in the ADA license attached to the workspace objects. The given category must match that on the ADA license.
User In Attached License	Checks whether the user being evaluated is listed on any or all of the ADA licenses attached to the workspace objects.
User In License	Verifies that the user being evaluated is listed in the ADA license.
User In Named License	Checks whether the user being evaluated is listed on an ADA license of the specified name. It does not check if the license is attached to the workspace objects being evaluated.
User-ADA Lic Has Citizenship	Checks whether the user's citizenship matches the passed-in value and then sees if the user's citizenship is on any of the ADA licenses attached to the workspace object being evaluated.
International Traffic in Arms Regulations (ITAR)	
Citizenship On Any ITAR Lic	Checks whether a citizenship of the user being evaluated matches any of the citizenships applied to the ITAR licenses attached to the workspace objects.
Group Nationality	Checks whether the given nationality matches the group nationality.
Has Government Classification	Compares the classification level in the condition argument with the object classification level. If the object is not classified, or if the object classification level is less than that of the given classification in the argument, this condition returns **True**.
Has ITAR License Of Category	Checks whether the workspace object being evaluated has any ITAR license of the given category.
Has Named ITAR License	Checks whether a specific ITAR license is attached to the workspace objects being evaluated.
Has No Government Classification	Checks if there is no government classification value on the workspace object.
ITAR License Has Citizenship	Checks whether the ITAR license being evaluated has the given citizenship.
Site Geography	Checks whether the given geography matches the geography of the site being evaluated.
User Citizenship	Checks whether the given citizenship matches the citizenships of the user being evaluated.
User Citizenship Or Nationality	Checks whether the given citizenship matches the citizenship or nationality of the user being evaluated.
User Declared Geography	Checks whether the given geography matches the geography the user declared when logging on to the system. Note: For more information about **USER_DECLARED_GEOGRAPHY**, see Configure geography access.
User Geography	Checks whether the given geography matches the geography of the user being evaluated.
User Has Government Clearance	Checks whether the government classification level of the user being evaluated is equal to, greater than, or less than the value specified in the condition.

Condition	Description
User In Attach ITAR Lic of Ctgry	Checks whether the user being evaluated is listed in the ITAR licenses attached to the workspace objects. The given category must match that on the ITAR license.
User In Attached ITAR License	Checks whether the user being evaluated is listed on any or all of the ITAR licenses attached to the workspace objects.
User In Named ITAR License	Checks whether the user being evaluated is listed on an ITAR license of the specified name. It does not check if the license is attached to the workspace objects being evaluated.
User Is ITAR Licensed	Checks whether the user currently logged on is cited in a valid (not expired) ITAR license attached to the workspace object either directly or by membership in a cited organization (group).
User Nationality	Checks whether the given nationality matches the nationality of the user being evaluated.
User TTC Expired	Checks whether the current date is later than the technology transfer certification (TTC) date on the **User** object.
User-ITAR Lic Has Citizenship	Checks whether the user's citizenship matches the passed-in value and then sees if the user's citizenship is on any of the ITAR licenses attached to the workspace object being evaluated.
Intellectual property (IP) license	
Citizenship On Any IP Lic	Checks whether the citizenship of the user being evaluated matches any of the citizenships applied to the IP licenses attached to the workspace objects.
Has IP Classification	Checks whether the IP classification of the workspace object being evaluated is equal to, greater than, or less than the value specified in the condition.
Has IP License Of Category	Checks whether the workspace object being evaluated has any IP license of the given category.
Has Named IP License	Checks whether a specific IP license is attached to the workspace objects being evaluated.
Has No IP Classification	Checks whether the workspace object does not have a value specified in the IP classification attribute.
IP License Has Citizenship	Checks whether the IP license being evaluated has the given citizenship.
User Has IP Clearance	Checks whether the IP clearance level of the user being evaluated is equal to, greater than, or less than the value specified in the condition.
User In Attach IP Lic of Ctgry	Checks whether the user being evaluated is listed in the IP license attached to the workspace objects. The given category must match that on the IP license.
User In Attached IP License	Checks whether the user being evaluated is listed on any or all of the IP licenses attached to the workspace objects.
User In Named IP License	Checks whether the user being evaluated is listed on an IP license of the specified name. It does not check if the license is attached to the workspace objects being evaluated.
User Is IP Licensed	Checks whether the user being evaluated is listed on an IP license attached to the workspace object.
User-IP Lic Has Citizenship	Checks whether the user's citizenship matches the passed-in value and then sees if the user's citizenship is on any of the IP licenses attached to the workspace object being evaluated.
Exclude licenses	
Citizenship On Any Exclude Lic	Checks whether the citizenship of the user being evaluated matches any of the citizenships applied to the exclude licenses attached to the workspace objects.

Condition	Description
Exclude License Has Citizenship	Checks whether the exclude license being evaluated has the given citizenship.
Has Exclude License Of Category	Checks whether the workspace object being evaluated has any exclude license of the given category.
Has Named Exclude License	Checks whether a specific exclude license is attached to the workspace objects being evaluated.
User In Attach Excl Lic of Ctgry	Checks whether the user being evaluated is listed in the exclude license attached to the workspace objects. The given category must match that on the exclude license.
User In Attached Exclude License	Checks whether the user being evaluated is listed on any or all of the exclude licenses attached to the workspace objects.
User In Named Exclude License	Checks whether the user being evaluated is listed on an exclude license of the specified name. It does not check if the license is attached to the workspace objects being evaluated.
User Is Excluded	Checks whether the user being evaluated is listed on an exclude license attached to the workspace object.
User-Exclude Lic Has Citizenship	Checks whether the user's citizenship matches the passed-in value and then sees if the user's citizenship is on any of the exclude licenses attached to the workspace object being evaluated.

Accessor types by category

The following table lists the accessor types by category.

Accessor type	Accessor (input argument)	Description
General		
Owning User	Not applicable	Evaluates any **POM_application_object**. Evaluates to true if the current logged-on user matches the user listed on the **owning_user** attribute of the object being evaluated. **Example:** `ObjecA.owning_user=User1` If **User1** logs on, this accessor type evaluates to true. If **User2** logs on, this accessor type evaluates to false.
Owning Group	The group of the user who first created the object	Evaluates to true if the current logged-on user's group membership is the group listed on the **owning_group** attribute of the object being evaluated. The **owning_group** attribute is always set to the group of the user who first created the object. Additional privileges (for example, write) may be granted to the owning group, because it is common for users to share data with other members of their group.

Accessor type	Accessor (input argument)	Description
		Note: By default, members of a subgroup receive the same access privileges set on workspace objects as their parent group who owns the object (the owning group). To change the privilege inheritance, use the **TC_allow_group_hierarchy_traversal** preference. **Example:** `ObjecA.owning_group=Group1` If **Group1** logs on, this accessor type evaluates to true. If **Group2** logs on, this accessor type evaluates to false.
Participant	Defines a specific participant type.	Evaluates to true if the current logged-on user is added as a dynamic participant on the object being evaluated and its **Participant** type matches the configured accessor (in the ACL). **Example:** `Accessor Type=Participant and the Accessor=Analyst` **User1** is added as an Analyst on source object. **User2** is not added as an Analyst on source object. If **User1** logs on with any group/role combination, this accessor type evaluates to true for the source object being evaluated. If **User2** logs on with any group/role combination, this accessor type evaluates to false for the source object being evaluated.
Group	Any group named in the Organization application	Evaluates to true if the current logged-on user's group membership matches the current logged-on user's group. **Example:** `Accessor Type=Group and the Accessor=Group1` If **User1** logs on as a member of **Group1**, this accessor type evaluates to true. If **User1** logs on as a member of **Group2**, this accessor type evaluates to false.
Groups with Security	A user whose group has the given security value, either **Internal** or **External**	Evaluates to true if the current logged-on user has the given security value, either **Internal** or **External**. This value is used to distinguish between groups in the parent company (internal) and suppliers (external). **Example:** `Accessor Type=Groups with Security and the Accessor=Internal` If **Group1** user logs on as **Internal** (for example, company employee), this accessor type evaluates to true.

Accessor type	Accessor (input argument)	Description
		If **Group2** user logs on as a member of **External** (for example, supplier), this accessor type evaluates to false.
Role	Any role named in the Organization application	Evaluates to true if the current logged-on user's role membership matches the current logged-on user's role. **Example:** `Accessor Type=Role and the Accessor=Role1` If **User1** logs on as **Role1**, this accessor type evaluates to true. If **User1** logs on as **Role2**, this accessor type evaluates to false.

Role in Group	A specific role	Evaluates to true if the current logged-on user performs the same skills and/or responsibilities as other users on the same project.
		Example:
		`Accessor Type=Role in Group`
		`and the Accessor=TranslatorFrench`
		If **User1** logs on as **TranslatorFrench**, this accessor type evaluates to true.
		If **User1** logs on as **TranslatorSpanish**, this accessor type evaluates to false.

Role in Owning Group	A specific role	Evaluates to true if the current logged-on user's role grants specific privileges. For example, all designers in the owning group are usually granted write privilege on their development data.
		Example:
		`Accessor Type=Role in Group and the`
		`Accessor=Designer`
		If **User1** logs on as **Designer**, this accessor type evaluates to true.
		If **User2** logs on as **Consultant**, this accessor type evaluates to false.

System Administrator	A user who is a member of the system administration group	Evaluates to true if the current logged-on user is a member of the system administration group.
		Example:
		`Accessor Type=System Administrator`
		`and the Accessor=SystemAdministrationGroup`
		If **User1** logs on as belonging to **SystemAdministrationGroup**, this accessor type evaluates to true.
		If **User1** logs on as belonging to **Group2**, this accessor type evaluates to false.

Group Administrator	A user who has special maintenance privileges for the group	Evaluates to true if the current logged-on user has group administrator privileges. A group administrator is a group member who can add, modify, or remove group members. **Example:**

Accessor type	Accessor (input argument)	Description
		`Accessor Type=Group Administrator and the Accessor=User1` If **User1** logs on, this accessor type evaluates to true. If **User2** logs on, this accessor type evaluates to false.
Site	Any site named in the Organization application	Evaluates to true if the current logged-on site (Teamcenter installation) matches the site listed on the site attribute of the object being evaluated. **Example:** `Accessor Type=Site and the Accessor=Site1` If **User1** logs on as being on **Site1**, this accessor type evaluates to true. If **User1** logs on as being on **Site2**, this accessor type evaluates to false.
Remote Site	Any remote site	Evaluates to true if the current logged-on remote site (Teamcenter installation) matches the remote site listed on the remote site attribute of the object being evaluated. **Example:** `Accessor Type=Remote Site and the Accessor=RemoteSite1` If **User1** logs on as being on **RemoteSite1**, this accessor type evaluates to true. If **User1** logs on as being on **RemoteSite2**, this accessor type evaluates to false.
World	Any user on the system	Evaluates to true, as this represents *all* users. **Example:** `Accessor Type=World and the Accessor=User1` If **User1** logs on as **World**, this accessor type evaluates to true. If **User2** logs on as **World**, this accessor type evaluates to true.
User	Any user named in the Organization application	Evaluates to true if the current logged-on user matches the user listed on the user attribute of the object being evaluated.
User In License	A specific user	Evaluates to true if the current logged-on user is listed on the license either through the user or group value. The term ADA license refers to any ITAR, IP, or exclude license. **Example:** `Accessor Type=User and the Accessor=User1` If **User1** logs on as **User**, this accessor type evaluates to true. User is not listed in the ADA_License object being evaluated. If **User2** logs on as **User**, this accessor type evaluates to false.

Accessor type	Accessor (input argument)	Description
Workflow		

Approver (RIG)	Any role that is designed as an approver in the workflow process.	Evaluates to true if the current logged-on user's role matches the user who is a signoff team member in the workflow process for the group.
		
		Note:
		This accessor must only be used in a workflow ACL and match the signoff **RIG** requirements for the release level associated with the workflow ACL.
		
		Example:
		
		`Accessor Type=Approver (RIG)`
		`and the Accessor=Override Approver`
		`in Validation Administration`
		
		If **User1** logs on as **Override Approver in Validation**, this accessor type evaluates to true.
		
		If **User2** logs on as **Designer in Engineering**, this accessor type evaluates to false.

Approver (Role)	Any user designed as an approver in the workflow process.	Evaluates to true if the current logged-on user's role matches the user who is a signoff team member in the workflow process for the group.
		
		Note:
		This accessor must only be used in a workflow ACL.
		
		Example:
		
		`Accessor Type=Approver (Role)`
		`and the Accessor=Approver`
		
		If **User1** logs on with **Approver** role, this accessor type evaluates to true.
		
		If **User2** logs on with **Designer** role, this accessor type evaluates to false.

Approver (Group)	Any group that is designed as an approver in the workflow process.	Evaluates to true if the current logged-on user's role matches the user who is a signoff team member in the workflow process for the group.
		
		Note:
		This accessor must only be used in a workflow ACL.
		
		Example:
		
		`Accessor Type=Approver (Group)`
		`and the Accessor=Engineering`
		
		If **User1** logs on as a member of the **Engineering** group, this accessor type evaluates to true.
		
		If **User2** logs on as a member of the **Simulation** group, this accessor type evaluates to false.

Accessor type	Accessor (input argument)	Description
Approver	Any user designed as an approver in the workflow process.	Evaluates to true if the current logged-on user, who is a signoff team member, has approver privileges. Note: This accessor must only be used in a workflow ACL. **Example:** `Accessor Type=Approver and the Accessor=User1` If **User1** logs on, this accessor type evaluates to true. If **User2** logs on, this accessor type evaluates to false.

Task Owner	A user who is granted privileges for the task's target data.	Evaluates to true if the current logged-on user has task owner privileges for the task's target data.
		
		Note:
		This accessor must only be used in a workflow ACL.
		
		Example:
		
		`Accessor Type=Task Owner`
		`and the Accessor=User1`
		
		If **User1** logs on, this accessor type evaluates to true.
		
		If **User2** logs on, this accessor type evaluates to false.

Task Owning Group	A group that is granted privileges for the task's target data.	Evaluates to true if the current logged-on user is a member of the task owning group.
		
		Note:
		This accessor must only be used in a workflow ACL.
		
		Example:
		
		`Accessor Type=Task Owning Group`
		`and the Accessor=OwningGroup`
		
		If **User1** logs on as a member of **OwningGroup**, this accessor type evaluates to true.
		
		If **User2** logs on as a member of **Engineering**, this accessor type evaluates to false.

Responsible Party	A user assigned as responsible for performing a particular task	Evaluates to true if the current logged-on user is the person responsible for performing a particular task.

Accessor type	Accessor (input argument)	Description
		Note: This accessor must only be used in a workflow ACL. **Example:** `Accessor Type=Responsible Party and the Accessor=User1` If **User1** logs on, this accessor type evaluates to true. If **User2** logs on, this accessor type evaluates to false.
Project		
Project Team	Users in a project team to which the object is assigned	Evaluates to true if the current logged-on user is an active group member in a project team to which the object is assigned. **Example:** `Accessor Type=Project Team and the Accessor=User1` If **User1** logs on, this accessor type evaluates to true. If **User2** logs on, this accessor type evaluates to false. Note: This does not apply to project team members who are inactive group members.
Project Teams		Team members (active group members) in any active project for the object. Note: This does not apply to project team members who are inactive group members.
Current Project Team		Users who are members of a particular current project team. Applicable only when the project is set as the current project of the team members and if the current project is active.
Current Project Teams		Users who are members of current project teams. Applicable only when the object is in the current project of the team members, and the current project is active.
Regular Project Member		Evaluates to true if the logged-on user is a regular team member of the **TC_project** object on which access is being evaluated.
Administrator Project Member		Evaluates to true if the logged-on user is the administrator of the **TC_project** object on which access is being evaluated.
Team Admin Project Member		Evaluates to true if the logged-on user is the team administrator of the **TC_project** object on which access is being evaluated.

Accessor type	Accessor (input argument)	Description
Privileged Project Member		Evaluates to true if the logged-on user is the privileged team member of the **TC_project** object on which access is being evaluated.
Role in Projects of Object		Users who have a specific role in one of the projects of the object. This accessor is affected by the values set in the **AM_PROJECT_MODE** preference. It is effective only when the user is logged-on with the specified role in the current project, and the current project is one of the projects assigned to the defined object.
Role in Project		Project members with a specific role in a specific project. This is affected by the values set in the **AM_PROJECT_MODE** preference.
Scheduler		
Public Schedule		Access to all users for schedules that are templates or made public. This accessor applies to the Schedule Manager application.
RoleInAnySchedule		Membership privileges of the logged-on user across all schedules in the system. Member privileges (accessor IDs) can be **COORDINATOR**, **PARTICIPANT**, or **OBSERVER**. This accessor applies to the Schedule Manager application.
ADA		
User In License	Not applicable.	This accessor type controls access to a ADA License object only. Evaluates to true if the logged-on user is listed on the ADA license object through the user or group value. The term ADA license refers to any ITAR, IP, or exclude license. Note: If there are no licenses attached to the object being evaluated, this accessor type evaluates to false. **Example:** **Use case scenario:** **User1** is listed on **ADALicense1**. **User2** is *not* listed on **ADALicense1**. **Evaluation results:** • When the **ADALicense1** is evaluated, **User1** evaluates to true because **User1** is listed on **ADALicense1**. • When the **ADALicense1** is evaluated, **User2** evaluates to false because **User2** is *not* listed on the **ADALicense1**.
User Not In License	Not applicable	This accessor type controls access to a ADA License object only. Evaluates to true if the logged-on user is *not* listed on the ADA license object through the user or group value. The term ADA license refers to any ITAR, IP, or exclude license. **Example:** **User1** is listed on **ADALicense1**, but **User2** is not.

Accessor type	Accessor (input argument)	Description
		If **User2** logs on and access is getting evaluated for **ADALicense1** then this accessor type will evaluate to true since **User2** is *not* listed on license **ADALicense1** If **User1** logs on and access is getting evaluated for **ADALicense1** then this accessor type will evaluate to false since **User1** is listed on license **ADALicense1**
User Excluded	Not applicable.	The user or group is listed in a valid exclude license attached to the workspace object being evaluated. The term ADA license refers to any ITAR, IP, or exclude license. **Example:** `Accessor Type=User Excluded and the Accessor=User1` If **User1** logs on with a valid exclude license, this accessor type evaluates to true. If **User2** logs on with no valid exclude license, this accessor type evaluates to false.
ITAR		
User Has Government Clearance		Compares the user's clearance with the object classification and tests whether the user has clearance above, below, or equal to that required to access the object.
User ITAR Licensed	Not applicable.	Evaluates to true if the current logged-on user is cited in a current license associated with the selected object. **Example:** `Accessor Type=User ITAR Licensed and the Accessor=License1` If **User1** logs on with **License1**, this accessor type evaluates to true. If **User2** logs on with no **License1**, this accessor type evaluates to false.
User ITAR Unlicensed		The user is not cited in a current license associated with the selected object.
User Under Government Clearance		The user's clearance is below the level required by the object. This accessor is typically used to revoke access and is only applicable when the government clearance on the user and the government classification on the object come from a common multi-level scheme defined by the **ITAR_level_list_ordering** preference.
User Over Government Clearance		The user's clearance is over the level required by the object. This accessor is typically used to grant access and is only applicable when the government clearance on the user and the government classification on the object come from a common multilevel scheme defined by the **ITAR_level_list_ordering** preference.
IP		
User Is IP Licensed	Any user cited in a current license associated with the selected object either	Evaluates to true if the current logged-on user is cited in a current license associated with the selected object either directly or by membership in a cited organization (group). **Example:**

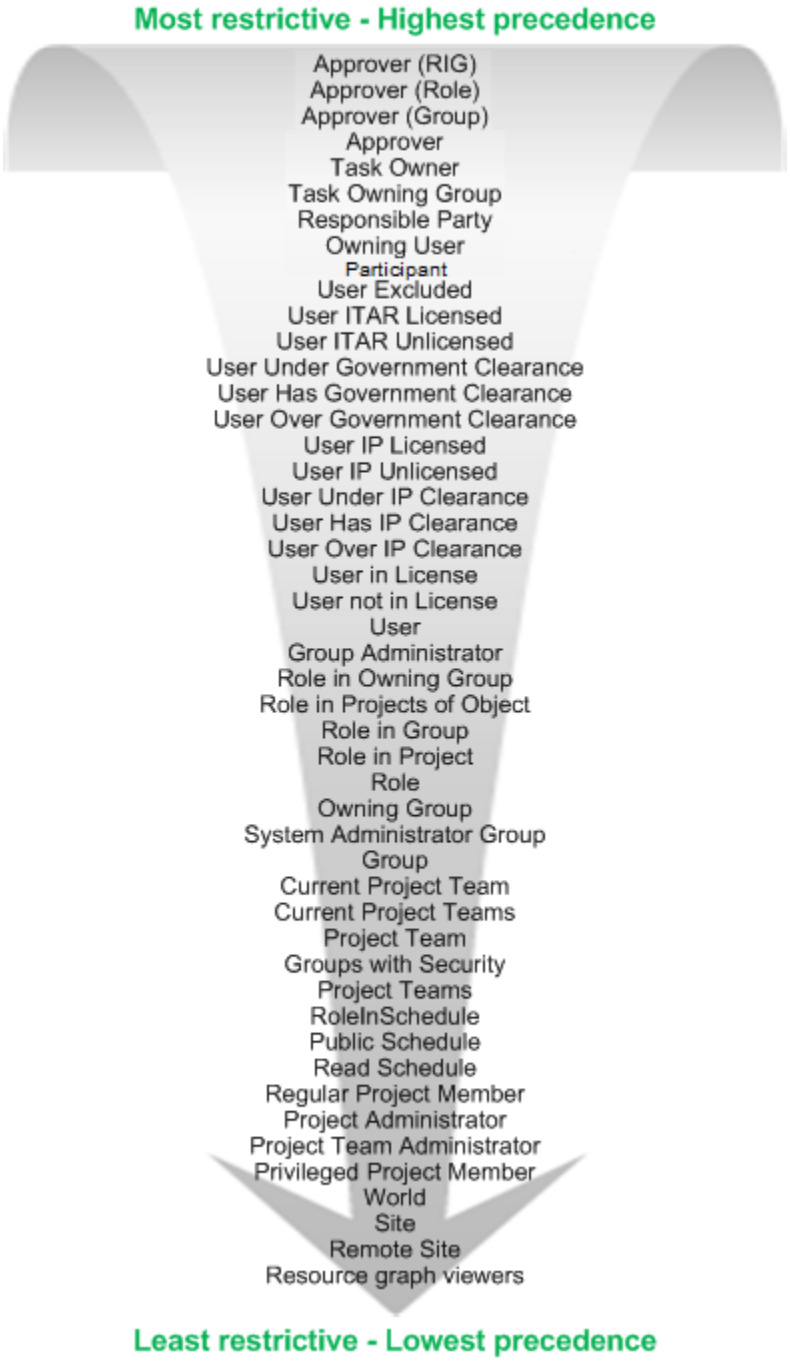
Accessor type	Accessor (input argument)	Description
	directly or by membership in a cited group	`Accessor Type=User IP Licensed and the Accessor=User1` If **User1** logs on as **User**, this accessor type evaluates to true. If **User2** logs on as **User**, this accessor type evaluates to false.
User IP Unlicensed		The user is not cited in a current license associated with the selected object.
User Has IP Clearance		Compares the user's clearance **(secret**, **super-secret**, **top-secret**) with the object classification and tests whether the user has clearance above, below, or equal to that required to access the object.
User Over IP Clearance		The user's clearance is over the level required by the object. This accessor is typically used to grant access and is only applicable when the IP clearance on the user and the IP classification on the object come from a common multi-level scheme defined by the **IP_level_list_ordering** preference.
User Under IP Clearance		The user's clearance is below the level required by the object. This accessor is typically used to revoke access and is only applicable when the IP clearance on the user and the IP classification on the object come from a common multi-level scheme defined by the **IP_level_list_ordering** preference.

Accessor precedence

An *accessor* is a user or group of users who share certain traits, such as membership in the group that owns the object or membership in the project team. The following list presents the predefined accessors delivered with Teamcenter in order of precedence, from most restrictive to least restrictive. The more restrictive the accessor, the higher precedence it has over other accessors.

Note:

- When two accessors with different precedences are added to a named ACL configuration, the highest precedence accessor is automatically moved to the top in the ACL table.
- When two accessors with the same precedence are added to a named ACL configuration, they stay in the order they are added.
- The **Role in Group**, **Role in Owning Group**, **Role in Project**, and **Role in Project of Object** accessors work on the superset of roles the user possesses in the relevant group or project, rather than on the session current role.
- When the **TC_current_role** preference is set, it affects the evaluation of the **Role in Owning Group**, **Role in Group**, and **Role** accessors. It enforces object access based on the user's current role in the current group.
- When the **AM_PROJECT_MODE** preference is set, it affects the evaluation of the **Role in Project** and **Role in Project of Object** accessors.

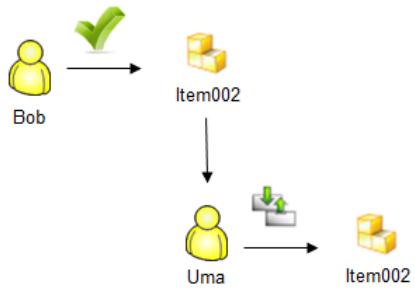


Access privileges

Symbol	Privilege	Description
	**Create**	Controls the creation of objects.

Symbol	Privilege	Description
		Note: There are best practices for ACLs to consider when creating an ACL with the **Create** privilege.
	**Read**	Controls the privilege to open and view an object.
	**Write**	Controls the privilege to check the object in/out of the database and modify it. Note: Even if a user has write permission, Teamcenter will still enforce certain rules, like the item_id property cannot be modified while the object is referenced by a release status, for example.
	**Delete**	Controls the privilege to delete the object from the database. Note: Even if a user has delete permission, Teamcenter will still enforce certain rules, like an object cannot be deleted if it is referenced by another object, for example.
	**Change**	Controls the privilege to modify object protections that override the rules-based protection for the object. You must have change privileges to apply object-based protection (object ACLs).
	**Promote**	Controls the privilege to move a task forward in a workflow process.
	**Demote**	Controls the privilege to move a task backward in a workflow process.
	**Copy**	Controls the privilege to copy an object as a new object.

Symbol	Privilege	Description
		Note: It still allows copy and paste of the object as a reference, with no new object created.
	**Change ownership**	Controls the privilege required to grant, change, or restrict ownership rights to an object. Note: **Write** access is required.
	**Publish**	Controls the publish privilege to users or groups.
	**Subscribe**	Controls the privilege to subscribe to an event on a specified workspace object.
	**Export**	Controls the privilege to export objects from the database.
	**Import**	Controls the privilege to import objects in to the database.
	**Transfer out**	Controls the privilege to transfer ownership of objects when they are exported from the database.
	**Transfer in**	Controls the privilege to assign ownership of objects when they are imported in to the database.
	**Write Classification ICO**	Controls the privilege to write Classification objects (ICOs).
	**Assign to project**	Controls the privilege to assign an object to a project. This applies to users who are not designated as privileged project team members. Note: The validation of the **Assign to project** privilege in conjunction with privileged project membership is evaluated based on the value of the **TC_project_validate_conditions** preference.
	**Remove from project**	Controls the privilege to remove an object from a project. This applies to users who are not designated as privileged project team members.

Symbol	Privilege	Description
		Note: The validation of the **Assign to project** privilege in conjunction with privileged project membership is evaluated based on the value of the **TC_project_validate_conditions** preference.
	Remote checkout	Controls the privilege to remotely check out an object.
	Unmanage	Enables users to circumvent the blocking implemented using the **TC_session_clearance** preference.
	IP Admin	Enables users to add users to manage IP licenses.
	ITAR Admin	Enables users to add administrative users to manage ITAR licenses.
	CICO	Grants a user the ability to override the checkout of an object by another user. It lets the user with the override privilege check in, transfer, or cancel the checkout of the object.
		Example: If Bob checks out an object (**item2**) and forgets to check it back in before leaving on vacation, the **CICO** privilege can be granted to the project manager, Uma, so she can check **item2** back in and the project can proceed.
		
	Translation	Controls the privilege to add translated text using the **Localization** button.

Symbol	Privilege	Description
	**View/Markup**	Controls the privilege to view and create markups.
	**Batch Print**	Controls the privilege to print multiple objects.
	**Digitally Sign**	Controls the privilege to digitally sign a document. The Commercial Off-The-Shelf (COTS) Digital Sign Dataset ACL rule grants owning user and owning group digital sign privileges for the dataset object. World users do not have digital sign privileges.
	**Void Digital Signature**	Controls the privilege to revoke or cancel an existing PKI digital signature for a business object. World users do not have void digital signature privileges.
	**Administer ADA Licenses**	Controls the privilege to create, modify, or delete ADA licenses for users in the ADA License application.
	**IP Classifier**	Controls the privilege to classify intellectual property (IP) information.
	**ITAR Classifier**	Controls the privilege to classify international traffic in arms (ITAR) information.
	**Remove Content**	Allows a user of 4G Design Management — Deployment and Rich Client Usage (4GD) to remove content from a collaborative design (CD), for example, to remove an existing design element.
	**Add Content**	Allows a user of 4G Design Management — Deployment and Rich Client Usage (4GD) to add content to a CD, for example, to create a new design element.
	**Effectivity**	Allows a user to modify effectivity on released objects.
	**Manage Variability**	Allows a user to add or remove the association between a product structure and a configurator context.
	**PDF Control**	Allows a user to add a system stamp and watermarks to an existing PDF using workflow.