TEAMCENTER

Audit Logs Management

Teamcenter 2412

SIEMENS

About Siemens Digital Industries Software

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

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

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

Contents

1. Getting started

Before you begin

Prerequisites	You need Teamcenter administrator privileges to use the Audit Manager application. You need Microsoft Excel to export audit logs to Excel. Before working with Microsoft Office datasets, verify that your computer has the required software installed.
Enable Audit Manager	• **TC_audit_manager** Ensure the preference is set to **ON** (default). • **TC_audit_manager_version** Ensure the preference is set to **3** (default).
Configure Audit Manager	There are additional steps you must perform to **configure Audit Manager** before you can use it.

Basic concepts for using audit functionality

System administrators use Audit Manager to create audit logs. Audit logs track what information has changed and who has changed the information.

To use audit functionality, you must first define audit logs by creating audit definition objects in Business Modeler IDE. You can then 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.

Defining audit logs

Audit logs are created based on the information specified in the **audit definition objects**. These define the information that should be captured about a particular object when an event occurs.

If you want to capture audit logs for events that are not available for logging, you can **create new events**.

When you create new events, you must **associate the event with an object**. Subsequently, you can create audit definition objects for that event and object type.

Accessing and viewing audit information

- Use Business Modeler IDE for creating audit definitions, events, and event mappings.
- Use the **Summary** view of the following Teamcenter applications to access or view audit logs in the **Audit logs** tab.
  - My Teamcenter
  - ADA License
  - Manufacturing Process Planner
  - Multi-Structure Manager
  - Organization
  - Schedule Manager
  - Structure Manager
  - Workflow Viewer
- You can run predefined **audit reports** or create new reports, using the Report Builder application.
- You can **create queries of audit logs**, using the Query Builder application.
- You can run predefined audit queries, using the Teamcenter advanced search functionality.

Audit objects

The audit functionality in Teamcenter uses the following objects:

Audit configuration objects

- **Fnd0AuditDefinition**

  Specifies the audit definition object. Audit definition allows you to define what information to log for a particular object and event type combination.

- **Fnd0AuditDefProperty**

  Specifies the logged properties object. This object stores logged properties information.

- **Fnd0DigitalSignatureAudit**

Specifies digital signature audit logs.

- **Fnd0EventTypeMapping**

  Specifies the event type mapping object. This object maps an event to an object, following which you can create audit definitions for the mapped object and event type combination.

- **ImanEventType**

  Specifies the event type object. This object stores events.

Audit log objects

- **Fnd0WorkflowAudit**

  Specifies workflow audit log object. This object stores process and signoff history audit logs.

- **Fnd0LicenseChangeAudit**

  Stores the license change audit logs.

- **Fnd0LicenseExportAudit**

  Stores the license export audit logs.

- **Fnd0FileAccessAudit**

  Stores file access audit logs.

- **Fnd0OrganizationAudit**

  Stores organization audit logs.

- **Fnd0StructureAudit**

  Stores structure audit logs.

- **Fnd0ScheduleAudit**

  Stores schedule audit logs.

- **Fnd0SecurityAudit**

  Stores security audit logs.

- **Fnd0GeneralAudit**

Stores the audit logs that are not stored in other audit logs.

- **Fnd0SecondaryAudit**

  Stores additional information or secondary information about objects such as attachments and attachment properties.

- **Fnd0AuditLink**

  Stores the link between primary and secondary audit logs.

2. Configuring Audit Manager

Enable Audit Manager

- Ensure that the **TC_audit_manager** preference is set to **ON** (default value).
- Ensure that the **TC_audit_manager_version** preference is set to **3** (default value).

> Note:
>
> If you update these preferences:
>
> - Restart Teamcenter server manager if you are on a four-tier installation.
> - Log off and log on to Teamcenter if you are on a two-tier installation.

Configuring access rule for deleted objects

For viewing the audit logs of deleted objects, Teamcenter provides access control rules that restrict access to audit logs of deleted objects to administrators.

You view this access rule as follows:

In Access Manager, under **Has Class (POM_Object)**

Condition = Has Class

Value = Fnd0AuditLog

ACL Name = AuditLog Access

This access rule gives administrators read privileges to audit logs of deleted objects and denies read privileges to the others.

> Note:
>
> If you are installing a new database, this rule is automatically created.
>
> If you are upgrading an existing database, you must upgrade Access Manager rules.

Post-upgrade steps required for importing custom event types into a template project

If you upgrade a Business Modeler IDE template project from a version earlier than Teamcenter 10, then after you upgrade the Business Modeler IDE template project, you must add its legacy custom event types. This is because Audit Manager objects are now managed using the Business Modeler IDE.

To help you import these custom event types, the system identifies the custom event types definitions during the upgrade process and writes them to a **custom_audit_configurations.xml** file generated under the *TC_DATA***model** directory. At the end of the upgrade process, Teamcenter Environment Manager (TEM) issues a warning if there are any custom event types.

Postupgrade, import these custom event type definitions into your custom template project before deploying any changes to the upgraded database. If not, the next TEM update process or Business Modeler IDE deployment tries to delete these event types, which may or may not pass based on whether there are references to it in the database.

Perform the following steps in the Business Modeler IDE immediately after the successful upgrade to Teamcenter and before deploying any data model changes:

1. Import the **custom_audit_configurations.xml** file from the *TC_DATA***model** directory into your custom template project by choosing **File→Import→Business Modeler IDE→Import template file**.

2. In the **BMIDE** view, right-click the project and choose **Reload Data Model**. Make sure there are no model errors reported in **Console** view.

3. Populate the appropriate display names to the custom event types wherever necessary.

4. Package and deploy the template to the Teamcenter database.

You can configure Audit Manager using the Business Modeler IDE.

Configuring Audit Manager business object constants

You must configure the following business object constants to work with Audit Manager.

- **Fnd0ObjectIDToAudit**

  Specifies the property that holds the object ID for the business object type. The object ID property differs across business object types. For example, on the **Item** business object type, the value for this constant is **item_id** and on the **ADA_License** business object type, the value is **id**. When an audit log is written for an instance of the business object, the property in this constant is used to obtain the object's ID for the audit log (and is written to the **fnd0PrimaryObjectID** property on the **Fnd0GeneralAudit** business object). For custom objects that have their own property for the object

ID, change this constant to the property that holds the object ID so that the ID of the business object is captured when the audit log is written.

This constant is placed on the **POM_object** business object and its children. There is no default value. Type a value in the **Value** box to assign an object ID.

This constant is provided by the **foundation** template file.

- **Fnd0ObjectNameToAudit**

  Specifies the property that holds the object name for the business object type. The object name property is different for different business object types. For example, on the **Workspace** business object type, the value for this constant is **object_name** and on the **User** business object type, the value is **user_name**. When an audit log is written for an instance of the business object, the property in this constant is used to obtain the object name for the audit log (and is written to the **name** property on the **AuditLog** business object). For custom objects that have their own property for the object name, change this constant to the property that holds the object name so that the name of the business object is captured when the audit log is written.

  This constant is placed on the **POM_object** business object and its children. There is no default value. Type a value in the **Value** box to assign an object name.

  This constant is provided by the **foundation** template file.

- **Fnd0ObjectRevIDToAudit**

  Specifies the property that holds the revision ID for the business object type. The revision ID property differs across business object types. For example, on the **ItemRevision** business object type, the value for this constant is **item_revision_id**. When an audit log is written for an instance of the business object, the property in this constant is used to obtain the object's revision ID for the audit log. For custom objects that have their own property for the revision ID, change this constant to the property that holds the ID so that the revision ID of the business object is captured when the audit log is written.

  This constant is placed on the **POM_object** business object and its children. There is no default value. Type a value in the **Value** box to assign an object revision ID.

  This constant is provided by the **foundation** template file.

- **Fnd0AuditRecordAccessLevel**

  Controls how access to a workflow audit record is evaluated. This constant is placed on the **Fnd0WorkflowAudit** business object. The default value is **1**. Click the arrow on the **Value** box to select from the following available values:

  - **1**

Checks workflow audit record based on the read access to the objects referred by the **fnd0Object** property (primary object) and the **fnd0SecondaryObject** property (secondary object). If the primary and secondary objects are deleted, only the administrator user is provided with read access to the audit record.

- **2**

  Checks workflow audit record based on the read access to all attachments for that workflow. Access to the primary object is not evaluated. If any of the secondary objects are deleted, only the administrator user is provided with read access.

This constant is provided by the **foundation** template file.

Configuring access controls for audit logs

You can view audit logs only if you have access to the relevant object. Audit logs can contain primary and secondary objects.

For viewing **audit logs of deleted objects**, Teamcenter provides access control rules that restrict access to audit logs of deleted objects to administrators.

The following table shows the conditions under which you can view audit logs, based on the access you provide to objects:

Primary object exists	Secondary object exists	Read access to primary object	Read access to secondary object	Show audit record	Comment
Yes	No	Yes	–	Yes	
Deleted	No	–	–	No	Display logs only to the administrator.
Yes	Yes	Yes	Yes	Yes	
Yes	Deleted	Yes	–	No	Display logs only to the administrator.
Deleted	Yes	–	Yes	No	Display logs only to the administrator.
Deleted	Deleted	–	–	No	Display logs only to the administrator.

Configuring access controls for workflow audit logs

Set access for workflow objects, using Access Manager and update the value of the **Fnd0AuditRecordAccessLevel** constant as follows:

- **Fnd0AuditRecordAccessLevel=1**

  The access check on the workflow audit records is evaluated based on the read access to the primary and secondary objects. Only administrators can view the audit records if the primary or secondary objects are deleted.

Primary object exists	Secondary object exists	Read access to primary object	Read access to secondary object	Show audit record	Comment
Yes	No	Yes	–	Yes	
Deleted	No	–	–	No	Display logs only to the administrator.
Yes	Yes	Yes	Yes	Yes	
Yes	Deleted	Yes	–	No	Display logs only to the administrator.
Deleted	Yes	–	Yes	No	Display logs only to the administrator.
Deleted	Deleted	–	–	No	Display logs only to the administrator.

- **Fnd0AuditRecordAccessLevel=2**

  The access check on the workflow audit records is evaluated based on the read access to the current list of attachments for that workflow. Access to the primary object is not evaluated.

Primary object exists	Read access to attachment	Show audit record	Comment
Yes	Yes	Yes	Teamcenter checks access to the current list of attachments.
Deleted	Yes	Yes	Teamcenter checks access to the current list of attachments.
Yes/Deleted	Yes At least one attachment is deleted without removing it from the workflow.	No	Show logs only to the administrator.
Yes/Deleted	Yes At least one attachment is deleted and removed from the workflow	Yes	Teamcenter checks access to the current list of attachments. The attachments that are deleted and removed from the workflow are not evaluated.

Configuring audit log archiving

Update the following business object constants to configure the audit log archiving:

- **Fnd0ArchiveLocation**

  Specifies the location of the audit logs (for example, **c:\archive**).

- **Fnd0RetentionPeriod**

  Specifies the retention period of the audit log archives in days (for example, 90).

Append audit log display value with real value for LOV-attached properties

> Note:
>
> This functionality is specific to string properties.

There are times when the *real value* is different from the *value display name*. It is important for the audit record that the *value display name* appear. To append an audit log *value display name* with a *real value* for LOV-attached properties on audit logs, set the **TC_audit_append_lov_display_value** preference to **true**. The display value for the LOV-attached property logged on audit log is in master locale.

The following examples illustrate appending the *value display name* to the *real value* recorded on audit record.

Example 1 - Appending the associated vendor value display name while auditing the Vendor LOV-attached properties

In the **General Audit** log, you want the *real value* of the vendor property for wheel component base (**77229 Wheel Component Base**) to show with the *value display name* of the associated vendor (**Acme Wheel Base Component**). Therefore, the *property value on the audit record* is **77229 Wheel Component Base:Acme Wheel Base Component**.

Real value	Value display name	Property value on audit record
77229 Wheel Component Base	**Acme Wheel Base Component**	77229 Wheel Component Base:Acme Wheel Base Component
89534 Wheel Component Base	**Beta Wheel Base Component**	89534 Wheel Component Base:Beta Wheel Base Component
12809 Wheel Component Base	**Cabbs Wheel Base Component**	12809 Wheel Component Base:Cabbs Wheel Base Component

Example 2 - Appending the associated organization value display name while auditing the Organization/Department LOV-attached properties

In the **General Audit** log, you want the *real value* of the organization/department codes (**77.909.990**/**7.909.991**) to show with the *value display name* of the associated organization/department (**Industrial Technology**/**Engineering**). Therefore, the *property value on the audit record* is **77.909.990:Industrial Technology,7.909.991: Engineering**.

Real value	Value display name	Property value on audit record
77.909.990,7.909.991	**Industrial Technology**, **Engineering**	77.909.990:Industrial Technology,7.909.991: Engineering

Configure the display of audit logs in the summary view

The **TC_audit_number_of_logs_to_load** preference configures the number of audit logs to load in the summary view. The default number of audit logs shown in the summary view is 100.

> Note:
>
> The time needed to load audit logs increases as you increase the value of this preference. After updating the value of this preference, ensure that you test the summary view for the time required to load audit logs.

Creating custom log handlers

Creating custom log extensions

A log extension allows you to log data beyond that captured by audit logs. To capture the data you require, you can create custom log extensions in addition to those provided by Teamcenter.

You can create custom log extensions by using the Business Modeler IDE extensions mechanism.

Teamcenter provides the following **log extensions**:

- **Fnd0CICO_auditloghandler**

  Logs checkin and checkout information, change ID, and the reason to audit. Applies this information to checkin and checkout events.

- **Fnd0OCC_track_position_orientation_audithandler**

  Logs the occurrence position and orientation changes of the components in structures.

- **Fnd0PROJInfo_audithandler**

  Logs project names that are assigned to the project. The project names are separated using commas.

- **Fnd0USER_get_additional_log_info**

  Logs workflow information to audit logs. For example, for the **__Assign** event, this handler logs information such as the process name, task type, user comments, and the user ID and user name the workflow is assigned to.

- **Fnd0WriteSignoffDetails**

  Logs the workflow signoff history.

- **Fnd0WriteSecondaryProperties**

  Logs secondary properties.

These extension points have the following values:

Business object	**Fnd0AuditDefinition**
Operation name	**Fnd0writeAuditLog**
Extension point	**PostAction**

Custom log extension example

In this example, a log extension named **P3_CustomAuditLog** is attached to the **EPMTask:_Reject:isTrue** audit definition object. You can create log extensions, using Business Modeler IDE.

1. In Business Modeler IDE, create a template project if you have not already created one.
2. Define the **P3_CustomAuditLog** extension.
   a. Create a library named **P3_custom**.
   b. Expand the project and the **Rules→Extensions** folders.
   c. Right-click the **Extensions** folder and choose **New Extension Definition**.

      The New Extension Definition wizard runs.
   d. Perform the following in the **Extension** dialog box:
      A. In the **Name** box, type **P3_CustomAuditLog**.
      B. In the **Language** box, select **CPlusPlus**.
      C. In the **Library** box, select **P3_custom**.
      D. Click **Add** to the right of the **Availability** table and perform the following in the **Extension availability** dialog box:
         i. In the **Business Object Name** box, select **Fnd0AuditDefinition**.
         ii. In the **Operation Name** box, select **fnd0writeAuditLog**.
         iii. In the **Extension Point** box, select **PostAction**.

iv. Click **Finish** in the **Extension availability** dialog box

The extension appears.

3. Attach the **P3_CustomAuditLog** extension to the **EPMTask:_Reject:isTrue** audit definition object.

a. From the **Extensions** view, open **Audit Manager→Audit Definitions**. Right-click the **EPMTask:_Reject:isTrue** audit definition and choose **Open**.

The **Audit Definition: EPMTask:_Reject:isTrue** view is displayed.

b. Click **Add** to the right of the **Audit Extensions** table, and select the **P3_CustomAuditLog** extension from the **Find an Audit Extension** dialog box.

c. To save the changes to the data model, choose **BMIDE→Save Data Model**.

4. Implement the **P3_CustomAuditLog** extension.

a. Right-click the **P3_CustomAuditLog** extension and choose **Generate extension code**.

The extension boilerplate code is generated into a **P3_CustomAuditLog.cxx** C++ file and a **P3_CustomAuditLog.hxx** header file. To see these files, open the project in the **Navigator** view and browse to the **src\server\P3_custom** directory.

> Note:
>
> You might need to right-click in the view and choose **Refresh** to see the files that were generated.

b. Open the **P3_CustomAuditLog.cxx** file in a C/C++ editor and add your custom business logic.

Generated code has method signature as **int P3_CustomAuditLog (METHOD_message_t *msg, va_list args)**

Replace the method signature as below in **P3_CustomAuditLog.cxx** and **P3_CustomAuditLog.hxx**

```
int P3_CustomAuditLog
( tag_t  /*targetObjTag*/,
int /*secondaryObjectCount*/,
tag_t* /*secondaryObjectTags*/,
char** /*secondaryQualifiers*/,
tag_t /*eventType*/,
int paramCount,
char** paramNames,
char** paramValues,
int /*errorCode*/,
```

```
const char* /*errorMessage*/,
tag_t primaryAuditBOTag )
```

The following is a sample file:

```
#include <P3_custom/P3_CustomAuditLog.hxx>
int P3_CustomAuditLog
( tag_t  /*targetObjTag*/,
  int /*secondaryObjectCount*/,
  tag_t* /*secondaryObjectTags*/,
  char** /*secondaryQualifiers*/,
  tag_t  /*eventType*/,
  int    paramCount,
  char** paramNames,
  char** paramValues,
  int /*errorCode*/,
  const char* /*errorMessage*/,
  tag_t primaryAuditBOTag )
{
    int ifail = ITK_ok;

    //tag_t     job_tag         = NULLTAG;
    char*       signoff         = NULL;
    tag_t       signoffTag      = NULLTAG;
    tag_t       auditClassId    = NULLTAG;
    tag_t       signoffClassId  = NULLTAG;
    char*       audit_class_name    = NULL;
    char*       signoff_class_name  = NULL;
    logical     isProp          = false;
    char*       propVal         = 0;


    if ( paramCount ==0 )
    {
        return ITK_ok;
    }
    for ( int i=0; i<paramCount; i++ )
    {
        if(strcmp(paramNames[i],"signoff") == 0)
        {
            signoff=paramValues[i];
            break;
        }
    }
    if (signoff == 0)
    {
        return ITK_ok;
    }

ifail = POM_class_of_instance( primaryAuditBOTag, &auditClassId );
ifail = POM_name_of_class  (auditClassId, &audit_class_name );
//ifail = EPM_ask_job(targetObjTag, &job_tag);
ifail = POM_string_to_tag(signoff, &signoffTag);

ifail = POM_class_of_instance( signoffTag, &signoffClassId );
ifail = POM_name_of_class  (signoffClassId, &signoff_class_name );

    isProp = false;
    ifail = POM_attr_exists("p3_decisioncode", signoff_class_name, &isProp);
```

```
    if (isProp)
    {
        ifail = AOM_ask_value_string  ( signoffTag,
"p3_decisioncode" ,&propVal);
        isProp=false;
        ifail = POM_attr_exists("p3_decisioncode", audit_class_name, &isProp);
        if (isProp)
        {
            ifail = AOM_set_value_string(primaryAuditBOTag, "p3_decisioncode",
propVal);
         }
    }
    if (audit_class_name)
    {
        MEM_free(audit_class_name);
    }
    if (signoff_class_name)
    {
        MEM_free(signoff_class_name);
    }
    if(propVal)
    {
        MEM_free(propVal);
    }

    return ITK_ok;

}
```

c. Open the **P3_CustomAuditLog.hxx** file and update it with the header information.

5. Build the library (**P3_Custom.dll** file) for the **P3_CustomAuditLog** extension.

6. (For hot deploy only) Copy the **P3_Custom.dll** file to the *TC_ROOT***bin** directory.

> Note:
>
> In a TEM deploy the required libraries are deployed with the solution package.

3. Using Audit Manager

Audit log overview

Audit logs are created based on the information specified in the **audit definition objects**. Audit definition objects define what information to capture about a particular object when an event occurs. However, audit logs are *not* generated for propagated values.

If you want to capture audit logs for events that are not available for logging, you can **create new events**.

When you create new events, you must **assign the event to an object**. Subsequently, you can create audit definition objects for that event and object type.

Create an audit definition

The following procedure shows how to create an *audit definition* that defines information to be captured when an event occurs to a particular kind of object.

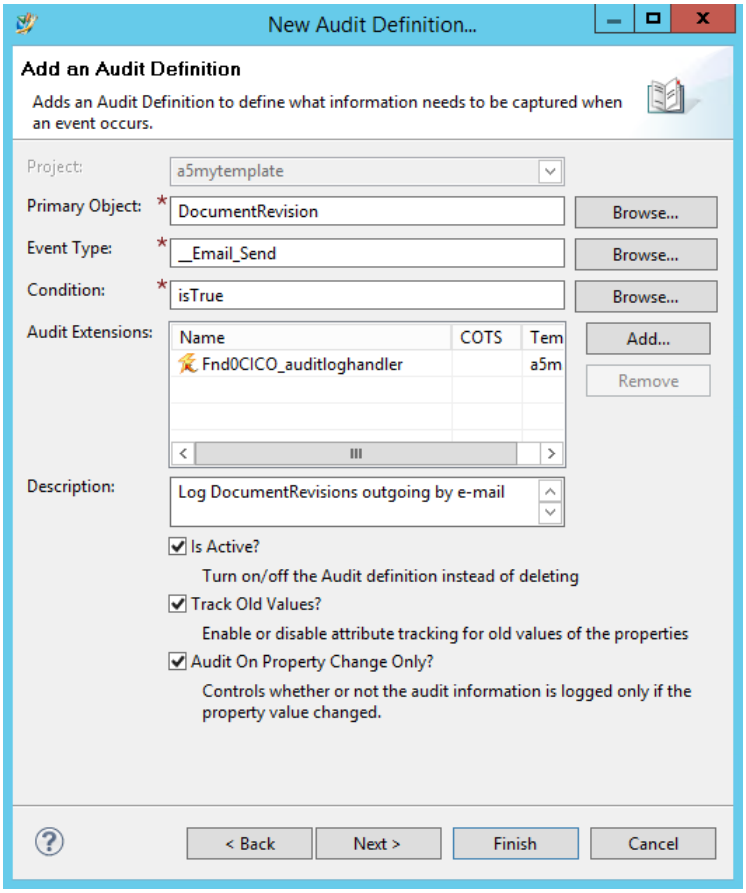
1. Ensure the following:

   - An event mapping has been created between the business object type and the event.
   - If you want to set a specific condition for activating logging of an event, then ensure that the condition is defined.

     > Example:
     >
     > You may want an audit entry to be written when a user accesses datasets that are export controlled, but not datasets that are not export controlled. Before creating the audit definition, create the condition definition.

2. Start the New Audit Definition wizard in one of these ways:

   - On the menu bar, choose **BMIDE→New Model Element**, type **Audit Definition** in the **Wizards** box, and click **Next**.
   - Open the **Extensions\Audit Manager** folders, right-click the **Audit Definitions** folder, and choose **New Audit Definition**.



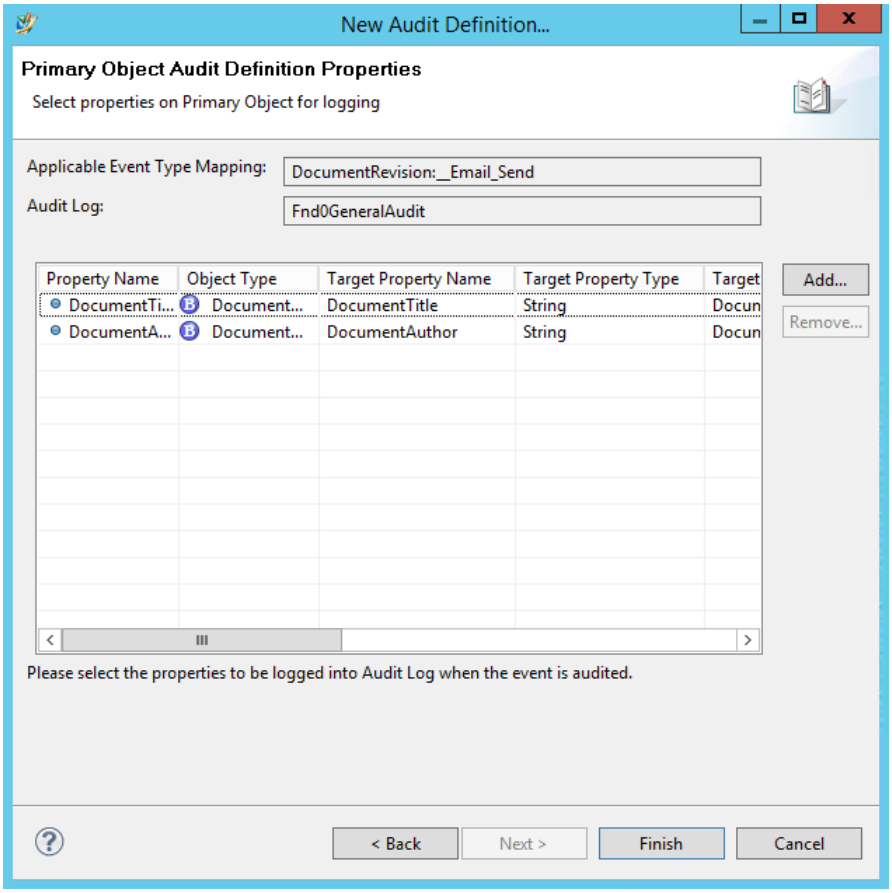
3. Specify basic parameters for the audit definition.

For this parameter	Do this
Primary Object	Click **Browse** and select the primary business object type that you want to audit.
Event Type	Click **Browse** and select the event that you want to audit for the selected business object.
Condition	Click **Browse** and select the condition under which the audit definition applies. Note: If the specified condition criteria are not matched, the audit log will always create.

For this parameter	Do this
	If you specify the **Delete** event type, then in the condition specify only the **UserSession** parameter.
Audit Extensions	Click **Add** and select the log extensions to use in the definition.
Description	Type a description of the purpose for this audit definition.
Is Active?	Select the check box to turn on the audit definition.
Track Old Values?	Select the check box to enable tracking of the old values of primary object properties. Note: Attribute tracking is available only for primary objects and is not supported for secondary objects.
Audit on Property Change Only?	Select the check box to log the information specified in this audit definition only if the property values change. This functionality is enabled only if the **Track Old Values?** check box is selected.

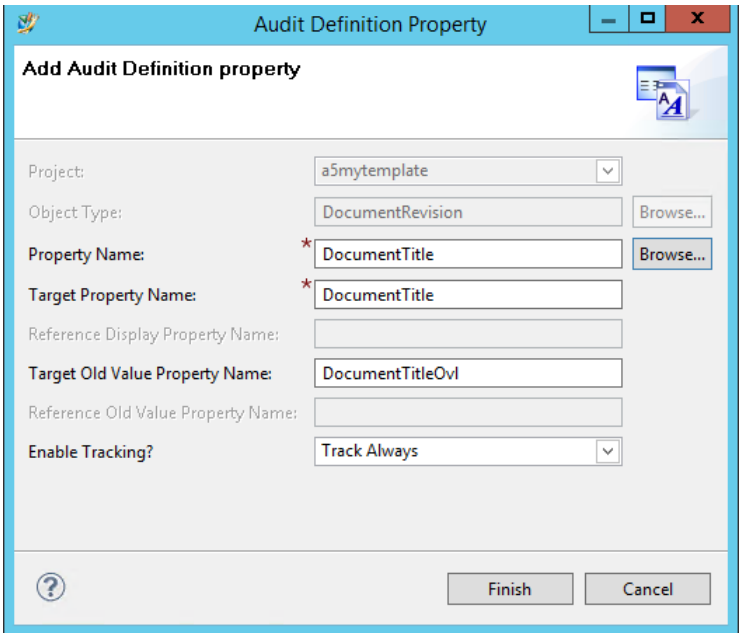
4. Click **Next**.

The **Primary Object Audit Definition Properties** dialog box is displayed.



a. In the **Primary Audit Definition Properties** dialog box, click **Add**.

The **Add Audit Definition property** dialog box is displayed.



b. In the **Add Audit Definition property** dialog box, to the right of **Property Name**, click **Browse** and select a property to track.

c. To change the display name of the property in audit logs, in **Target Property Name** type a new display name.

d. The **Target Old Value Property Name** box and the **Enable Tracking?** box are enabled if you selected the **Track Old Values?** check box in the **Add an Audit Definition** dialog box. As applicable, enter additional parameters for the audit definition property.

Parameter	Description
Target Old Value Property Name	To change the display name of the old property value in audit logs, type a new display name.
Enable Tracking?	Select the kind of tracking: **Track Always** Always tracks old and new values of properties, even if there are no changes to the property value. **No** Does not track changes to properties. **Track Different** Tracks old and new values of properties only when the property value changes.

e. Click **Finish**.

5. Add more primary object audit definition properties as needed.

6. (If the primary object can have secondary objects) To capture information from secondary objects that are related to or referenced by the main object being audited, click **Next** in the **New Audit Definition** dialog box and add secondary object audit definition properties.

7. When you are done adding properties, click **Finish** in the **New Audit Definition** dialog box.

Test the audit definition

Verify that the audit definition object can create audit logs.

Control the type of information you gather from data

You can control the type of information you gather from data by creating an audit definition. You can set conditions that log an event or change conditions on existing audit definitions. You can save time and resources by reducing:

- Maintenance costs by generating audit logs only when required
- Audit table sizes in the database
- Audit log generation per user

To set the **Condition** under which the audit definition applies, either click **Browse** or add a condition.

The **isTrue** and **isFalse** conditions are supported for all event types, including the **Delete** event type.

The **Audit Definition** display name changes from **ObjectType:__EventType:Condition** to **ObjectType:__EventType:Status**. Specify the status as active or inactive based on the **IsActive** flag on the audit definition.

If the condition does not match the above criteria, the audit log is always created.

If you specify the **Delete** event type, specify **User Session** as the only parameter in your condition.

See the out-of-the-box sample condition.

> Note:
>
> The condition signature expects three parameters. If you add a condition or audit definition, use the following allowable signature.

Condition Name	Condition Signature	Condition Expression
Fnd0AuditCondition	**Fnd0AuditCondition(UserSession us, BusinessObject bo1, BusinessObject bo2)**	As per needed in use case.

Parameters	Parameter Name	Definition
User Session	us	This parameter is User Session.
Business Object	primaryObject	This parameter can be a BusinessObject or any child of a Business Object. (This will be the

Parameters	Parameter Name	Definition
		primary object of the audition definition.)
Business Object	secondaryObject	This parameter can be a BusinessObject or any child of a Business Object. (This will be the secondary object of the audit definition.)

In the above signatures, primary object and secondary object can be ignored if it is not applicable. It can also be ignored if it is not using it in a condition signature for evaluation by just stating, POM_object in the signature, as it is required as condition signature to be in above format, i.e.

Fnd0AuditCondition(UserSession us, POM_object primaryObject, POM_object secondaryObject)

For example, if you are configuring condition on audit definition "Item:_Modify", you can write the condition this way:

Condition Name	Condition Signature	Condition Expression
Fnd0AuditCondition	**AuditCondition(UserSession us, Item it, POM_object po)**	**us.fnd0bypassflag=true**

Here Primary object(Item) is Primary object type from audit definition, you can specify secondary object as POM object if secondary object is not applicable.

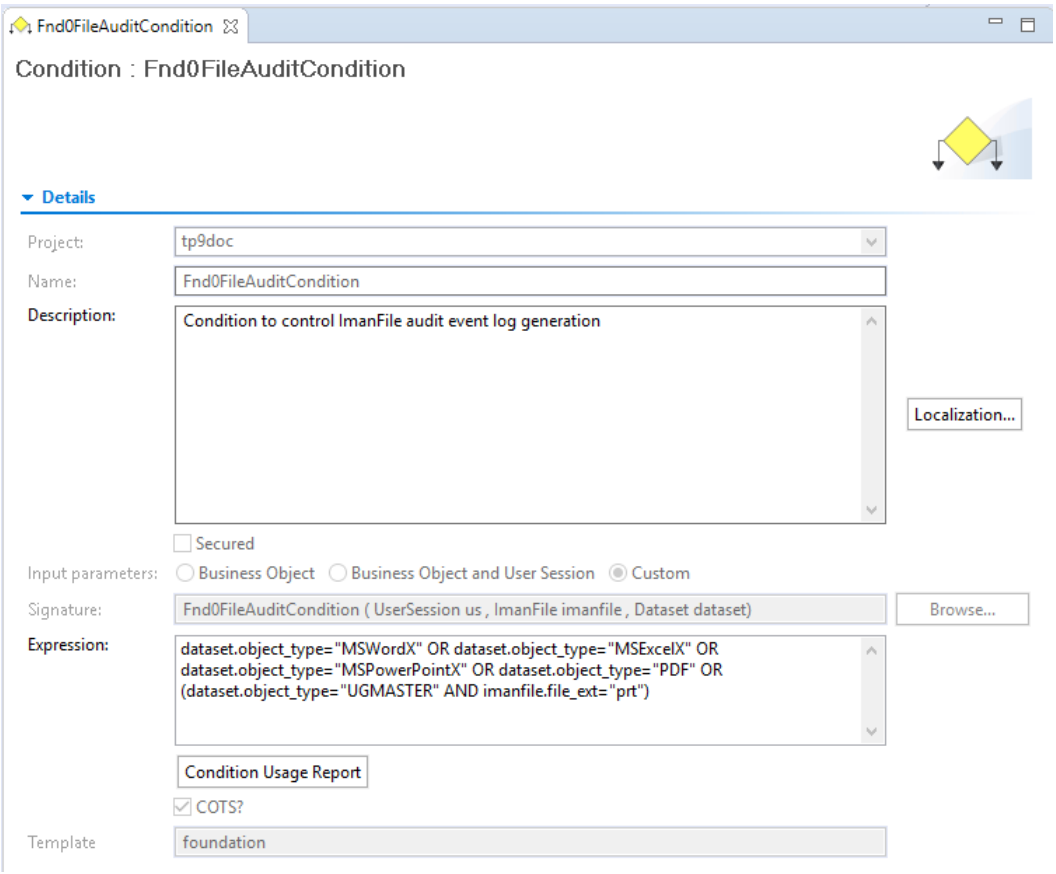
Control auditing file events using a sample condition

Creating conditions means **you can control the type of information you gather from data**. The **Fnd0FileAuditCondition** is a sample condition provided out-of-the-box for auditing **File** events.

To use this condition, configure it on **ImanFile** audit definitions and deploy the changes. Its default condition expression indicates it will audit **File** events when the dataset types are one of the following:

- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint
- PDF
- **UGMASTER** (and file extension is PRT).

Modify the expressions as needed, to provide granularity to audit **File** events.



> Note:
>
> Configure this condition on the **ImanFile** audit definition.

Condition Name	Condition Signature	Condition Expression
Fnd0FileAuditCondition	**Fnd0FileAuditCondition(User Session us, ImanFile imanfile, Dataset dataset)**	`dataset.object_type="MSWordX" OR dataset.object_type="MSExcelX" OR dataset.object_type="MSPowerPointX" OR dataset.object_type="PDF" OR (dataset.object_type="UGMASTER" AND imanfile.file_ext="prt")`

You can use the **Fnd0FileAuditCondition** condition as an alternative to the **TC_audit_disabled_dataset_types** preference, which is deprecated in Teamcenter 12.2 and will be removed in future releases. This deprecated preference was designed to specify the list of dataset types that should not be audited for **ImanFile** audit events.

Here are a few examples for using this sample condition, **Fnd0FileAuditCondition**.

- **Use case 1:** Audit only the PRT files attached to the **UGMASTER** dataset type.

  This is an example of Item Revision:

```
Item Revision
  -- UGMASTER
        -- part001.prt
        -- images_preview.qaf
        -- qafmetadata.qaf


Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile , Dataset
dataset) :=  dataset.object_type="UGMASTER" AND file.file_ext="prt"
```

This condition allows audit log generation for the PRT file of **UGMASTER** dataset type.

- **Use case 2:** Audit only the classified dataset of type **UGMASTER**.

```
Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile ,
Dataset dataset) :=  dataset.object_type="UGMaster" AND
dataset.ip_classification = "secret"
```

- **Use case 3:** Audit **UGMASTER** dataset type only.

```
Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile , Dataset
dataset) :=  dataset.object_type="UGMASTER"
```

This condition allows audit log generation of **UGMASTER** dataset type only.

- **Use case 4:** Audit only the PRT and JT files attached to the **UGMASTER** dataset type.

This is an example of Item Revision:

```
Item Revision
  -- UGMASTER
      -- part001.prt
      -- images_preview.qaf
      -- qafmetadata.qaf
      -- direct_model_data.jt

Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile , Dataset
dataset) :=  dataset.object_type="UGMASTER" AND file.file_ext="prt"
AND file.file_ext="jt"
```

This condition allows audit log generation for PRT and JT files of the **UGMASTER** dataset type.

- **Use case 5:** Do not audit thumbnail files for the **UGMASTER** dataset type.

Active Workspace renders thumbnail files for item revision, dataset object using image files under dataset named reference files.

Thumbnail files are generated for various use cases, such as view folder contents, open Item revision, open dataset, and open BOM content, wherever the object is listed.

Active Workspace uses the **TC__thumbnail_dataset_type_order** preference, which lists valid named references for previewing images as thumbnails. It also renders the content from the image files in the preview section.

For these use cases, image files are read for each thumbnail rendering and preview. Therefore, each single click may cause a file to be read multiple times.

The **ImanFile::__Read_file** audit definition, when enabled, generates an audit log for each thumbnail generation and preview. Consequently, Active Workspace generates a large number of audit logs for image files configured for thumbnail preview.

- **To control auditing of thumbnail files read events:**

  Active Workspace uses the **TC__thumbnail_dataset_type_order** preference, which lists valid named references for previewing images as thumbnails.

  Each dataset has named references and applicable file extensions listed in the Business Modeler IDE (BMIDE); use the **References** tab on each dataset to identify file extensions for each of the named references for the datasets you are using to control audit log generation.

- **To restrict thumbnail audit log generation for the UGMASTER dataset type:**

  This is an example of Item Revision:

```
Item Revision
  -- UGMASTER
      -- part001.prt
      -- images_preview.qaf
      -- qafmetadata.qaf
```

  Create a BMIDE condition, attach condition to audit definition **ImanFile:__Read_File** in BMIDE.

  Add a BMIDE condition expression:

```
Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile ,
Dataset dataset) :=  dataset.object_type="UGMASTER" AND
file.file_ext!="qaf" AND file.file_ext!="jpg" AND file.file_ext!
="bmp" AND file.file_ext!="gif"
```

> Note:
>
> This condition allows audit log generation for PRT files only for this example.

This condition expression evaluates to false for all other dataset types (other than **UGMASTER**), which may result in restricting audit log generation for other dataset types.

If you want to allow audit for other dataset types and restrict extensions only for the **UGMASTER** dataset type, you can enhance the condition expression as follows:

> Note:
>
> In addition to controlling auditing for thumbnail generation, this expression is applicable for all use cases where these files are referenced by the **UGMASTER** dataset, for example, download action.

```
Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile ,
Dataset dataset) :=  dataset.object_type!="UGMASTER" OR
(dataset.object_type="UGMASTER" AND file.file_ext!="qaf" AND
file.file_ext!="jpg" AND file.file_ext!="bmp" AND file.file_ext!
="gif")
```

- **Use case 6:** Do not audit thumbnail files.

  To restrict thumbnail audit log generation for all dataset types and all file extensions:

```
Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile , Dataset
dataset) :=  file.file_ext!="qaf" AND file.file_ext!="jpg" AND
file.file_ext!="bmp" AND file.file_ext!="gif" AND file.file_ext!="tif"
AND file.file_ext!="png"
```

> Note:
>
> In addition to controlling auditing for thumbnail generation, this expression is also applicable for all use cases where these files are read or downloaded.

- **Use case 7**: Do not audit system files.

  To restrict thumbnail audit log generation for all dataset types and all file extensions and restrict audit log generation for system dataset types: **CLIPS**, **XMLRenderingStylesheet**, **Fnd0IconResource**:

```
Fnd0FileAuditCondition ( UserSession us , ImanFile imanfile , Dataset
```

```
dataset) :=  dataset.object_type!="CLIPS" AND dataset.object_type!
="XMLRenderingStylesheet" AND dataset.object_type!="Fnd0IconResource"
```

For more configurability, you can a add business object operation as part of the condition expression.

Control ImanFile Read audit-log generation using a condition expression and BusinessObject operation

The following illustrate how to control audit-log generation using a condition expression and a BusinessObject operation:

1. Create a BusinessObject operation on a dataset (v4FileAuditValidation).

   You can also add an operation on the **ImanFile** object and achieve the same result (follow the below operation implementation and change where application for this case).

   Operation name: **v4FileAuditValidation**

   Business Object: Dataset

   ```
   int Dataset::v4FileAuditValidation( tag_t &fileObj, bool &verdict )
   ```

Name	Description	Type
`fileObj`	ImanFile object as input	Input
`verdict`	Verdict of the validation	Output

2. Update the **Fnd0FileAuditCondition** or create a new condition (**V4FileAuditCondition**)

   Use the operation in a condition expression, as follows:

   ```
   V4FileAuditCondition ( UserSession us , ImanFile imanfile , Dataset
   dataset) := dataset.v4FileAuditValidation(imanfile) = true
   ```

3. Attach an updated condition (**V4FileAuditCondition**) to **Audit Definition - ImanFile: Read File**.

4. Implement the operation:

   a. Generate the code for the Dataset operation added in the above step.

   b. Edit two files **DatasetImplExt.hxx** and **DatasetImplExt.cxx** generated in your **src** folder of the template project after you generate the code.

   c. Implement the following function:

```
int  ::v4::v4template::DatasetImpl::v4FileAuditValidationBase( ta
g_t & fileObj, bool & verdict )
```

d.

The following are examples of operation implementation:

Example 1 - Code snippet implementing the operation to audit only UGMASTER dataset and PRT file extension type

The following code segment illustrates how to implement the operation to audit only the **UGMASTER** dataset and **prt** file extension type.

The operation implementation returns **true** only when the dataset type is **UGMASTER** and the file extension is **prt**.

```
int  ::v4::v4template::DatasetImpl::v4FileAuditValidationBase( tag_t &
fileObj, bool & verdict )
{
    int ifail = ITK_ok;

    // Your Implementation
    verdict = false; // set default value as false
    tag_t dsTag = m_dsObj->getTag();
    if(dsTag == NULLTAG)
    {
        return ifail; // error handling
    }

    char* datasettype_name = 0;
    if( AOM_ask_value_string( dsTag, "object_type", &datasettype_name ) !
= ITK_ok)
    {
        return ifail; // error handling
    }

    char* strFileExt = 0;
    if( AOM_ask_value_string( fileObj, "file_ext", &strFileExt ) !=
ITK_ok)
    {
        return ifail; // error handling
    }

    if( tc_strcmp(datasettype_name, "UGMASTER") == 0 &&
tc_strcmp(strFileExt, "prt") == 0 )
    {
        verdict = true; // return true if UGMASTER dataset type and file
```

```
extension is prt.
    }
    else
    {
        verdict = false; // return false for all other dataset type and
file extensions.
    }

    MEM_free(datasettype_name);
    MEM_free(strFileExt);

    return ifail;
}
```

Example 2 - Code snippet implementing the operation to audit only UGMASTER dataset and PRT file extension type using reference name

The operation implementation returns **true** only when dataset type is **UGMASTER** and the file reference name is **UGPART**.

You can hard code the dataset type value (**UGMASTER**) and reference name (**UGPART**) within the code or you may choose to make it configurable using a preference.

The following example uses the **Custom_file_ref_names_to_audit** preference to store the reference name.

```
<preference name="Custom_file_ref_names_to_audit" type="String"
array="true" disabled="false" protectionScope="Site" envEnabled="false">
    <preference_description>Lists the dataset named references to audit
for ImanFile __Read_File event</preference_description>
      <context name="Teamcenter">
    <value>UGPART</value>
      </context>
</preference>
```

To locate a reference for a file extension from BMIDE:

1. In BMIDE, open any dataset, for example, the **UGMASTER** business object.

2. Open the **References** tab to see all applicable references.

3. Locate **.prt** in the **File Type** column. Note that **UGPART** is the reference name from the first column, **Reference**.

   In this example, use the **UGPART** value to validate within the operation implementation.

```
int  ::v4::v4template::DatasetImpl::v4FileAuditValidationBase( tag_t
& fileObj, bool & verdict )
{
    int ifail = ITK_ok;

    // Your Implementation
    verdict = false; // set default value as false
    tag_t dsTag = m_dsObj->getTag();
    if(dsTag == NULLTAG)
    {
        return ifail; // error handling
    }

    char* datasettype_name = 0;
    if( AOM_ask_value_string( dsTag, "object_type",
&datasettype_name ) != ITK_ok )
    {
        return ifail; // error handling
    }

    if( tc_strcmp(datasettype_name, "UGMASTER") != 0 )
    {
        MEM_free(datasettype_name);
        return ifail; // return false(default initialized value is
false) for all other dataset types.
    }
    MEM_free(datasettype_name);

    int referenceCount = 0;
    if( AE_ask_dataset_ref_count(dsTag, &referenceCount) != ITK_ok )
    {
        return ifail; // error handling
    }

    if (referenceCount <= 0)
    {
        return ifail; // no named references
    }

    // ask preference values for "Custom_file_ref_names_to_audit"
    std::vector< std::string > nameRefTypes;
    int prefValueCount = 0;
    if( PREF_ask_value_count("Custom_file_ref_names_to_audit",
&prefValueCount) != ITK_ok )
    {
        return ifail; // error handling
    }
```

```
    if (prefValueCount > 0)
    {
        char** prefValues = 0;
        if( PREF_ask_char_values("Custom_file_ref_names_to_audit",
&prefValueCount, &prefValues) != ITK_ok )
        {
            return ifail; // error handling
        }

        for (int inx = 0; inx < prefValueCount; ++inx)
        {
            std::string prefValue = std::string(prefValues[inx]);
            if (prefValue != "")
            {
                nameRefTypes.push_back(prefValues[inx]);
            }
        }

        MEM_free(prefValues);
    }

    if (nameRefTypes.size() <= 0)
    {
        return ifail; // no preference values
    }

    char* referenceName = 0;
    AE_reference_type_t  referenceType;
    tag_t referencedObject = NULLTAG;
    for (int i = 0; i < referenceCount; i++)
    {
        if (AE_find_dataset_named_ref2(dsTag, i, &referenceName,
&referenceType, &referencedObject) == ITK_ok)
        {
            if (referencedObject == fileObj)
            {
                std::vector< std::string >::iterator it =
std::find(nameRefTypes.begin(), nameRefTypes.end(), referenceName);
                if (it != nameRefTypes.end())
                {
                    verdict = true;
                }
                MEM_free(referenceName);
                break;
            }
        }
        MEM_free(referenceName);
    }
```

```
        return ifail;
    }
```

Example 3 - Restrict thumbnail audit log generation for all dataset types using reference name

Use the **Custom_file_ref_names_not_to_audit** preference to store the named reference to make it configurable. Preference values are the same as the OOTB **TC__thumbnail_dataset_type_order** preference.

Implement the operation to return **false** only when the input-file-named reference value is in the list of named references. Otherwise, the operation returns **true**.

```
<preference name="Custom_file_ref_names_not_to_audit" type="String"
array="true" disabled="false" protectionScope="Site" envEnabled="false">
    <preference_description>Lists the dataset named references not to
audit for ImanFile __Read_File event</preference_description>
      <context name="Teamcenter">
    <value>Image</value>
    <value>ThumbnailImage</value>
    <value>Fnd0IconResourceReference</value>
    <value>JPEG_Reference</value>
    <value>PNG_Reference</value>
    <value>GIF_Reference</value>
    <value>TIF_Reference</value>
    <value>UG-QuickAccess-Binary</value>
    <value>SE-QuickAccess-Binary</value>
    <value>Graphics-Interface</value>
    <value>Markup</value>
    <value>BMP</value>
    <value>JPEG</value>
    <value>Shaded-Image</value>
    <value>Mes0BMP</value>
    <value>Mes0JPG</value>
    <value>Mes0GIF</value>
    <value>Mes0PNG</value>
    <value>Mes0TIF</value>
    <value>Mes0JPG_Preview</value>
      </context>
</preference>

int  ::v4::v4template::DatasetImpl::v4FileAuditValidationBase( tag_t &
fileObj, bool & verdict )
{
    int ifail = ITK_ok;

    // Your Implementation
```

```
    verdict = true; // set default value as true
    tag_t dsTag = m_dsObj->getTag();
    if(dsTag == NULLTAG)
    {
        return ifail; // error handling
    }

    int referenceCount = 0;
    if( AE_ask_dataset_ref_count(dsTag, &referenceCount) != ITK_ok )
    {
        return ifail; // error handling
    }

    if (referenceCount <= 0)
    {
        return ifail; // no named references
    }

    // ask preference values for "Custom_file_ref_names_not_to_audit"
    std::vector< std::string > nameRefTypes;
    int prefValueCount = 0;
    if( PREF_ask_value_count("Custom_file_ref_names_not_to_audit",
&prefValueCount) != ITK_ok )
    {
        return ifail; // error handling
    }

    if (prefValueCount > 0)
    {
        char** prefValues = 0;
        if( PREF_ask_char_values("Custom_file_ref_names_not_to_audit",
&prefValueCount, &prefValues) != ITK_ok )
        {
            return ifail; // error handling
        }

        for (int inx = 0; inx < prefValueCount; ++inx)
        {
            std::string prefValue = std::string(prefValues[inx]);
            if (prefValue != "")
            {
                nameRefTypes.push_back(prefValues[inx]);
            }
        }

        MEM_free(prefValues);
    }

    if (nameRefTypes.size() <= 0)
```

```
    {
        return ifail; // no preference values
    }

    char* referenceName = 0;
    AE_reference_type_t  referenceType;
    tag_t referencedObject = NULLTAG;
    for (int i = 0; i < referenceCount; i++)
    {
        if (AE_find_dataset_named_ref2(dsTag, i, &referenceName,
&referenceType, &referencedObject) == ITK_ok)
        {
            if (referencedObject == fileObj)
            {
                std::vector< std::string >::iterator it =
std::find(nameRefTypes.begin(), nameRefTypes.end(), referenceName);
                if (it != nameRefTypes.end())
                {
                    verdict = false;
                }
                MEM_free(referenceName);
                break;
            }
        }
        MEM_free(referenceName);
    }

    return ifail;
}
```

Package and install customization

You can also use this operation to filter just by the dataset type or by file extensions. Or this condition expression can be combined with an operation. Also, this example can be extended to filter more dataset types and file extensions based on your business need.

```
V4FileAuditCondition ( UserSession us , ImanFile imanfile ,
Dataset dataset) :=  dataset.object_type!="UGMASTER" OR
(dataset.v4FileAuditValidation(imanfile) = true)
```

Create an event type

An *event* is an action that occurs to an object in Teamcenter, for example, when an item is checked out. Teamcenter records audit logs when certain events occur on certain types of objects.

You only need to create a new event type if there is not an existing event type that covers your needs. When you create a type, its name is only a text reminder of the type of information you are looking from

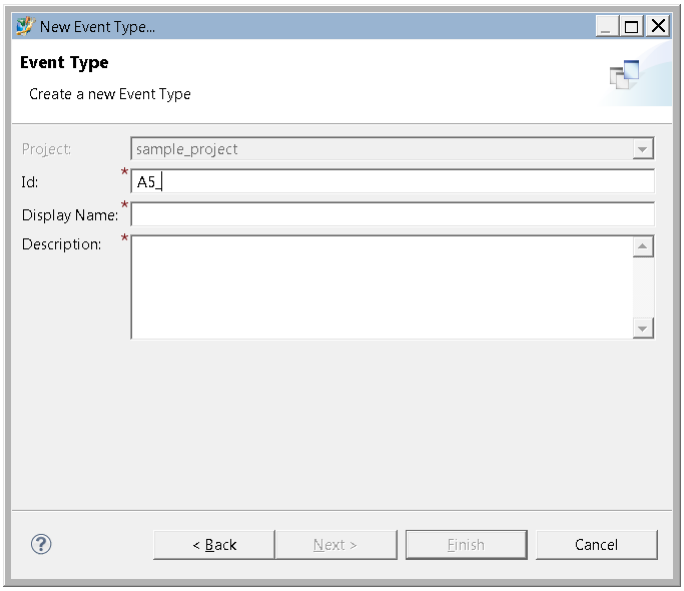
in the audit. The actual event information is captured by the audit type selected when you create the event type mapping.

In the past, the **install_event_types** utility was used to create new events. Now you create new event types using the Business Modeler IDE.

1. Choose one of these methods:

   - On the menu bar, choose **BMIDE→New Model Element**, type **Event Type** in the **Wizards** box, and click **Next**.
   - Open the **Extensions\Audit Manager** folders, right-click the **Event Types** folder, and choose **New Event Type**.

   The New Event Type wizard runs.



2. In the **Id** box, type the name of the new event.
3. In the **Display Name** box, type the name that you want the event to have in the user interface.
4. In the **Description** box, type a description of the new event so that others know what it is used for.
5. Click **Finish**.
6. **Create an event type mapping definition** to connect the event to a business object type.

Create an event type mapping

While an *event* is an action that occurs to an object in Teamcenter, *event mapping* is connecting an event to a business object type. In other words, the event mapping declares that you want to receive an audit log for a certain event on a certain kind of object.

An event mapping must be created for a business object type and event before you use that business object and event type in an audit definition. Event mapping is inherited by child business object types. For example, instances of the **Part** business object type inherit the mapping from the **Item** business object type.

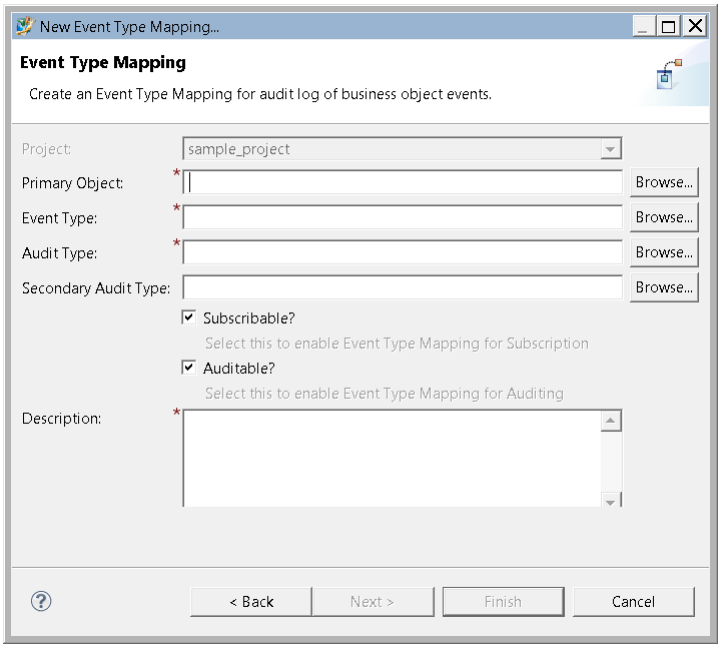
In the past, the event mapping was created using the **install_event_types** utility. Now event mapping is created using the Business Modeler IDE.

1. Choose one of these methods:

   - On the menu bar, choose **BMIDE→New Model Element**, type **Event Type Mapping** in the **Wizards** box, and click **Next**.

   - Open the **Extensions\Audit Manager** folders, right-click the **Event Types Mappings** folder, and choose **New Event Type Mapping**.

   The New Event Type Mapping wizard runs.



2. Click the **Browse** box to the right of the **Primary Object** box to select the type of business object you want to audit.

3. Click the **Browse** box to the right of the **Event Type** box to select the event you want to audit for the selected business object.

4. Click the **Browse** box to the right of the **Audit Type** box to select the type of audit to use for this mapping. The audit types are represented by business objects that are children of the **Fnd0AuditLog** business object.

5. Click the **Browse** box to the right of the **Secondary Audit Type** box to select the **Fnd0SecondaryAudit** business object. This **Secondary Audit** object stores information and properties about the secondary objects that are related to the main object being audited.

6. Select the **Subscribable?** check box to specify that the event type mapping can be subscribed to.

7. Select the **Auditable?** check box to specify that the event type mapping can be audited.

8. In the **Description** box, type a description for this mapping so that others know what it is used for.

9. Click **Finish**.

Recommendations for managing audit logs

Siemens Digital Industries Software recommends the following actions for managing audit logs:

- Opening objects in Teamcenter having audit records takes significant time. Siemens Digital Industries Software recommends, prior to upgrading Teamcenter or after installing Teamcenter, that you configure indexes on **AuditManager Solution III** audit tables by executing following commands:

```
install -add_func_index -u=<user> -p=<password>|-pf=<pwfile> -g=<group>
     wfaudit_object_inx 0 substr Fnd0WorkflowAudit fnd0Object
install -add_func_index -u=<user> -p=<password>|-pf=<pwfile> -g=<group>
     wfaudit_job_inx 0 substr Fnd0WorkflowAudit fnd0Job
install -add_func_index -u=<user> -p=<password>|-pf=<pwfile> -g=<group>
     generalaudit_object_inx 0 substr Fnd0GeneralAudit fnd0Object
install -add_index -u=<user> -p=<password>|-pf=<pwfile> -g=<group>
     fileaudit_secobject_inx 0 Fnd0FileAccessAudit fnd0SecondaryObject
```

- Avoid activating audit definitions you do not require.

- Do not deactivate the **EPMTask-Add Attachment** audit definition as some workflow queries are dependent on the **Add Attachment** event.

- Frequently purge or archive old audit data.

4. Relating audit objects, audit logs, and subscriptions

Audit objects allow administrators to define what information to log for Teamcenter object and event combinations. For example, when:

- Specified item types are checked out.
- Specified forms are created, modified, or deleted.
- Users approve specified target types in specified workflows.
- The **Complete** action on any workflow task fails.
- Access requests are denied.

Audit logs record events that occur to objects in Teamcenter. Administrators can create audit logs for virtually any event that occurs to a Teamcenter object. For example, audit logs can be created to track when:

- Any change is made to a specific type of document revision.
- A specific assembly is checked out.
- A Word document is added as a dataset to a specific type of item revision.
- Status is changed on specified parts.

Audit logs can be searched and viewed from Teamcenter. End users view audit logs to see the history of actions performed on objects.

Subscriptions are created by end users. Subscriptions are requests to be notified when data is modified by another user or when the release status of an item revision changes.

Users are notified by Teamcenter mail when the specified events occur on the specified objects.

Administration tools related to auditing events

Audit Manager	Administrators use Audit Manager to manage audit logs.
Subscription Administration	In order to manage all the different subscriptions created by users, administrators use Subscription Administration to generate subscription reports and to manage subscription tables.

5. Archiving and restoring data

Nearline archiving and restoring audit logs

Over time, as large volumes of data accumulate in Teamcenter, you may find reasons for archiving data.

- Managing database size.
- Archiving and restoring data on demand.
- Complying with legal and regulatory requirements.
- Reducing the window of time required to back up your production Teamcenter instance.

When you archive data, you can also restore it at any time. The **audit_purge** utility is enhanced with two arguments to support archive and restore of audit logs:

- Use the **nearline_archive** argument to move the production data to the archive site.
- Use the **nearline_restore** argument to move the archive data from the archive site to the production site.

> Note:
>
> Ensure these **prerequisites** are met before enabling archiving and restoring.

In addition, there are many arguments in the **audit_purge** utility that enable you to archive data based on given criteria. For example, which type of audit log, for which retention period, and so on.

Recovery of audit logs for nearline archive and restore operation failures

Audit log archive and restore operations may fail due to Integrated Distributed Services Manager (IDSM) and Object Directory Services (ODS) connection failures.

You can perform recovery and cleanup of partially created data for these types of failures in the archive and restore operation of audit logs using the **ar_recover** utility at the production site.

During the archive and restore critical operations, a recovery text dataset containing the operation ID and object Unique Identifier (UID) is created in the current logged on user's **NewStuff** folder. After a successful operation, the recovery dataset is automatically deleted. However, if a critical failure occurs, the dataset is not deleted and you can use the **ar_recover** utility to attempt recovery of the data.

Note:

Before you can perform an archive or restore operation of audit logs or reissue, you must first use the **ar_recover** utility to perform a recover operation from the previous failure.

Examples

For audit logs, only the recover operation (**-f=recover**) functionality is supported in the **ar_recover** utility.

Note:

To perform the recover operation, you must execute the **ar_recover** utility at the production site.

- To recover from a failed *archive* operation for audit logs:

```
ar_recover -u=tcadmin -p=tcadmin -g=dba -f=recover
   -ds_name="__Recover_archive_00003f58_5e731606_Fnd0AuditArchive-19-Mar-2020
   12:19:35"
```

- To recover from a failed *restore* operation for audit logs:

```
ar_recover -u=userid -p=password -g=group -f=recover
   -ds_name="__Recover_restore_00001dc0_5e7340c5_Fnd0AuditArchive-19-Mar-2020
   15:21:22"
```

6. Archiving and purging audit logs

You can make smart, cost-effective decisions while managing audit records, with greater flexibility in purging and archiving audit data.

The **audit_purge** utility enables you to archive or purge audit logs. This gives greater flexibility to:

- Control data growth.
- Maintain smaller database size, providing faster access to the database.
- Save time extracting useful audit data from the database.
- Easily clean the audit log in the database.

7. Viewing audit information

Audit reports

The following audit reports are available when you choose the **Tools→Reports→Report Builder Reports** menu command in My Teamcenter:

- **Audit - Digital Signature Report**

  Displays digital signatures associated with the specified object.

- **Audit - File Access Report**

  Displays access records of files for the specified object.

- **Audit - General Report**

  Displays general audit logs for the specified object.

- **Audit - License Change Report**

  Displays the ADA License changes for the specified license.

- **Audit - License Export Report**

  Displays the ADA License export for the specified license and object.

- **Audit – Organization Report**

  Displays the changes to an organization.

- **Audit - Security Report**

  Displays audit log events against a specified object.

- **Audit - Schedule Report**

  Displays the audit records for the specified schedule objects.

- **Audit - Structure Report**

  Displays the structure records for the specified objects.

- **Audit - Workflow Attachment Report**

Displays workflow logs for the specified workflow attachment.

- **Audit - Workflow Detailed Report**

  Displays workflow logs for the specified workflow process.

- **Audit - Workflow Signoff Report**

  Displays the signoff results and comments for the specified workflow process.

- **Audit - Workflow Summary Report**

  Displays the start, complete, approved, rejected, assign status, demote, promote, fail, and update actions for the specified workflow process.

Creating and running audit queries

You can create custom search queries for audit logs, using the Query Builder application. Saved queries identify the search criteria that are used to find information in Teamcenter.

> Note:
>
> Ensure that audit definitions exist for the objects for which you have created saved queries.

Teamcenter provides the following predefined audit queries:

- **Audit - Digital Signature Logs**
- **Audit - File Access Logs**
- **Audit - General Logs**
- **Audit - License Change Logs**
- **Audit - License Export Logs**
- **Audit - Organization Logs**
- **Audit - Project Based Logs**
- **Audit - Schedule Logs**
- **Audit - Security Logs**
- **Audit - Structure Logs**

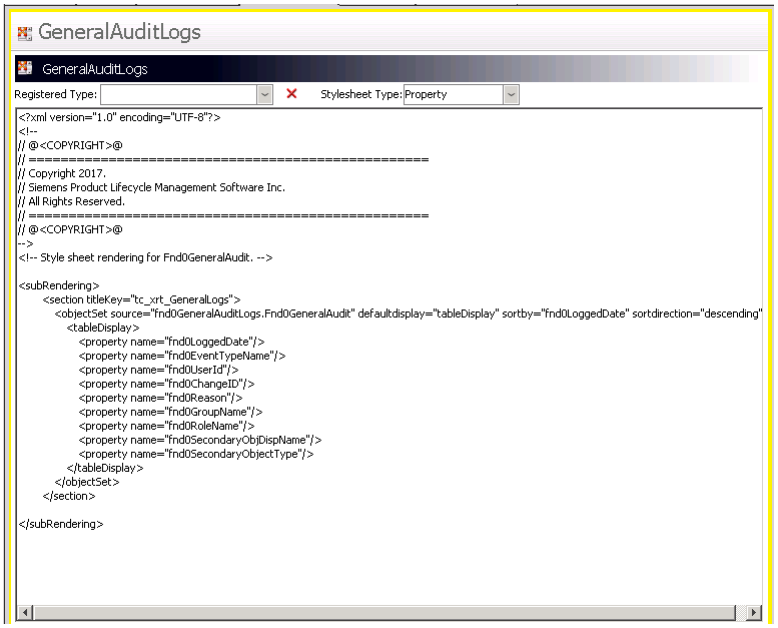
- **Audit - Workflow Attachment Logs**
- **Audit - Workflow Detailed**
- **Audit - Workflow General**
- **Audit - Workflow Signoff**
- **Audit - Workflow Summary**

Customize audit logs

You can customize which fields are displayed in each audit log.

Procedure

1. In the rich client, search for the audit log file you want to edit. For example, select the **GeneralAuditLogs** file to remove a field name in the **General Logs** audit log.

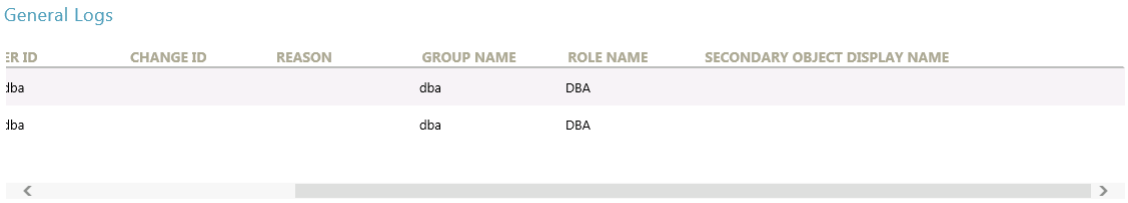


2. Delete the line associated with the field you want to remove. For example, delete the following line to remove the **Secondary Object Type** field:

```
<property name="fnd0SecondaryObjectType"/>
```

3. Save the **GeneralAuditLogs** file.

4. Using Active Workspace, verify the **Secondary Object Type** field was successfully removed.



Audit project events and view assigned or removed data

You can audit **Project** events.

Activate these to create an audit definition to audit **Project** events.

- The **TC_Project:Fnd0Assign_Data_To_Project** out-of-the-box audit definition.
- The **TC_Project:Fnd0Remove_Data_From_Project** out-of-the-box audit definition.

You can view audit logs for **Project** events.

If someone has assigned or removed data from a project, you can view it in the **Security** section of the **Audit Logs** tab in the **Project** summary page. You can customize viewing these logs by configuring the **Summary** stylesheet of the respective object.

Configure the **Summary** stylesheet to enable the out-of-the-box **Security** section under the **Audit Logs** tab in the rich client, so the audit log gets recorded under security audit logs.

For example, to add the **Security Logs** section for item revision, update the applicable stylesheet for the item revision with the following content.

```
        <section title="Security Logs" titleKey="tc_xrt_SecurityLogs"
 initialstate="collapsed">
            <objectSet source="fnd0SecurityAuditLogs.Fnd0SecurityAudit"
 defaultdisplay="tableDisplay" sortby="fnd0LoggedDate"
sortdirection="ascending">
                <tableDisplay>
                    <property name="fnd0LoggedDate"/>
                    <property name="fnd0EventTypeName"/>
                    <property name="fnd0PrimaryObjectID"/>
                    <property name="object_type"/>
                    <property name="object_name"/>
                    <property name="fnd0UserId"/>
                    <property name="fnd0GroupName"/>
                    <property name="fnd0RoleName"/>
                    <property name="fnd0SecondaryObjectID"/>
                    <property name="fnd0SecondaryObjectType"/>
                    <property name="fnd0SecondaryObjectName"/>
```

```
                    <property name="fnd0SecondaryObjDispName"/>
                </tableDisplay>
            </objectSet>
            <customPanel java="com.teamcenter.rac.auditmanager.
SecuritySecondaryAuditPanel" js="displaySecurityCustomPanel" />
        </section>
```

Export audit logs from the Summary view

The **Export to Excel** option is available on Windows clients. Microsoft Excel must be installed on your computer to export audit logs to Excel.

You can export audit logs to Microsoft Excel or csv formats from the Summary view of an object.

1. Select an object whose audit logs you want to export and click the **Summary** tab.

   In the Summary view, select the **Audit Logs** tab.

2. To export the audit logs in Microsoft Excel format, click the **Export to Excel** button.

   To export the audit logs in csv format, click the **Export to CSV** button.

   Depending on the export option you select, the **Export to Excel** or **Export to CSV** dialog boxes appear.

3. In the **Export to Excel** or **Export to CSV** dialog box, select the audit log type you want to export.

4. Click **OK**.

Depending on the export format you choose, Teamcenter exports the audit logs in Microsoft Excel or csv format.

Export audit logs to Microsoft Excel

The **Export to Excel** option is available on Windows clients. Microsoft Excel must be installed on your computer to export audit logs to Excel.

1. Run a saved query and choose the audit logs you want to export from the **Details** tab.

2. Choose **Tools→Export→Objects To Excel**.

   Teamcenter displays the **Export To Excel** dialog box.

3. Under **Object Selection**, click one of the following:

- Click **Export Selected Objects** to export the selected rows in the view.
- Click **Export All Objects in View** to export all rows.

4. Under **Output Template**, select one of the following:
   - Select **Export All Visible Columns** to export all the columns in the view.
   - Select **Use Excel Template** to activate the template list.

     In the list, select the **AUDIT_log_export_template_new** template.
5. Under **Output**, click **Static Snapshot.**
6. Click **OK** to generate the export Excel file.

Microsoft Excel opens a temporary file. You can create a permanent file by choosing **File→Save As** in Excel to display the **Save As** dialog box.

If you save a live Excel file, you can open it later in My Teamcenter to reconnect it to the database. Values that you cannot change in Teamcenter are unavailable in the cells of the live Excel file.

Export and import audit logs associated with items

To include audit logs associated with items when you export and import items from one site to another using high-level data transfer, update the options in the **TIEUnconfiguredExportDefault** transfer option set as follows:

- To export audit logs, set the value of the **Opt_exp_auditrec** option to **True**. The default value is **False**.
- To include audit logs associated with items when you export and import items from one site to another using high-level data transfer, update the **TIEUnconfiguredExportDefault** transfer option set as follows:
  - To export audit logs, set the value of the **Opt_exp_auditrec** option to **True**. The default value is **False**.
  - To export workflow audit logs, set the value of the **Opt_exp_workflow** option to **True**. The default value is **False**.
- To include audit logs associated with items when you export and import items from one site to another using low-level data transfer, update the options in the **SiteConsolidationDefault** transfer option set as follows:
  - To export audit logs, set the value of the **Opt_exp_auditrec** option to **True**. The default value is **True**.

- To export workflow audit logs, set the value of the **Opt_exp_workflow** option to **True**. The default value is **False**.

Best practices for managing audit logs

Following are best practices for managing audit logs:

- Avoid activating audit definitions you do not require.
- Make sure **EPMTask-Add Attachment, EPMJob:__Start, EPMJob:__Process_Initiated** audit definitions are active, as some workflow queries are dependent on these events.
- Frequently purge or archive old audit data.

A. Audit log extensions

Teamcenter provides the following log extensions:

- **Fnd0WriteSecondaryProperties**

  This log extension writes secondary object properties to the primary object. The following secondary properties are written to primary business objects:

Audit business objects	Fnd0WorkflowAudit business object	Fnd0StructureAudit, Fnd0ScheduleAudit, Fnd0OrganizationAudit, Fnd0GeneralAudit, and Fnd0LicenseExportAudit business objects
Secondary business object properties	fnd0SecondaryObject	fnd0SecondaryObject
	fnd0SecondaryObjectType	fnd0SecondaryObjectType
	fnd0SecondaryObjectName	fnd0SecondaryObjectName
	fnd0SecondaryObjectID	fnd0SecondaryObjectID
	fnd0SecondaryObjectRevID	fnd0SecondaryObjectRevID
	fnd0SecondaryObjDispName	fnd0SecondaryObjDispName
	fnd0SecondaryObjQualifier	

  By default, this log extension is attached to the following audit definitions:

  - **EPMTask:__Add_Attachment**
  - **EPMTask:__Remove_Attachment**
  - **ADA_License:__Attach_License**
  - **ADA_License:__Detach_License**
  - **BOMView Revision:__Component_Add**

- **BOMView Revision:__Component_Remove**
- **ScheduleTask:__ResourceAssignment_Create**
- **ScheduleTask:__ResourceAssignment_Modify**
- **ScheduleTask:__ResourceAssignment_Delete**
- **ScheduleTask:__TaskDependency_Create**
- **ScheduleTask:__TaskDependency_Modify**
- **ScheduleTask:__TaskDependency_Delete**

- **Fnd0OCC_track_position_orientation_audithandler**

  This log extension enables the logging of occurrence position and orientation changes of the components in structures. It is attached to the **PSOccurrence:__Modify** audit definition.

  > Note:
  >
  > You must not use this log extension in any audit definition except for audit definitions that log position and orientation changes of components based on occurrence properties.

- **Fnd0CICO_auditloghandler**

  This log extension applies to checkin and checkout events. It logs the change ID and reason why the object is checked in or out.

- **Fnd0PROJInfo_audithandler**

  This log extension logs information about objects that are assigned to projects. The project names are separated by commas.

  By default, this extension is not available on any audit definition.

- **Fnd0USER_get_additional_log_info**

  This log extension logs workflow information. For example, for the **__Assign** event, this log extension logs information like the process name, task type, user comments, user ID, and the user name the workflow is assigned to.

  This log extension applies to workflow-related events such as **__Abort**, **__Add_attachment**, **__Approve**, **__Assign**, **__Assign_Approver**, **__Complete**, **__Demote**, **__Fail**, **__Notify**, **__Perform**, **__Reject**, **__Stand_In**, **__Remove_Attachment**, **__Resume**, **__Start**, **__Suspend**, and **__Update_Process**.

- **Fnd0WriteSignoffDetails**

  This log extension logs workflow signoff history. For example, for the **__Add_attachment** event, this log extension logs information like signoff decision, job, sign off, user ID, sign off group, and sign off role name.

  This log extension applies to workflow related events such as **__Add_attachment**, **__Approve**, **__Reject**, **__Stand_In**, and **__Remove_Attachment**.