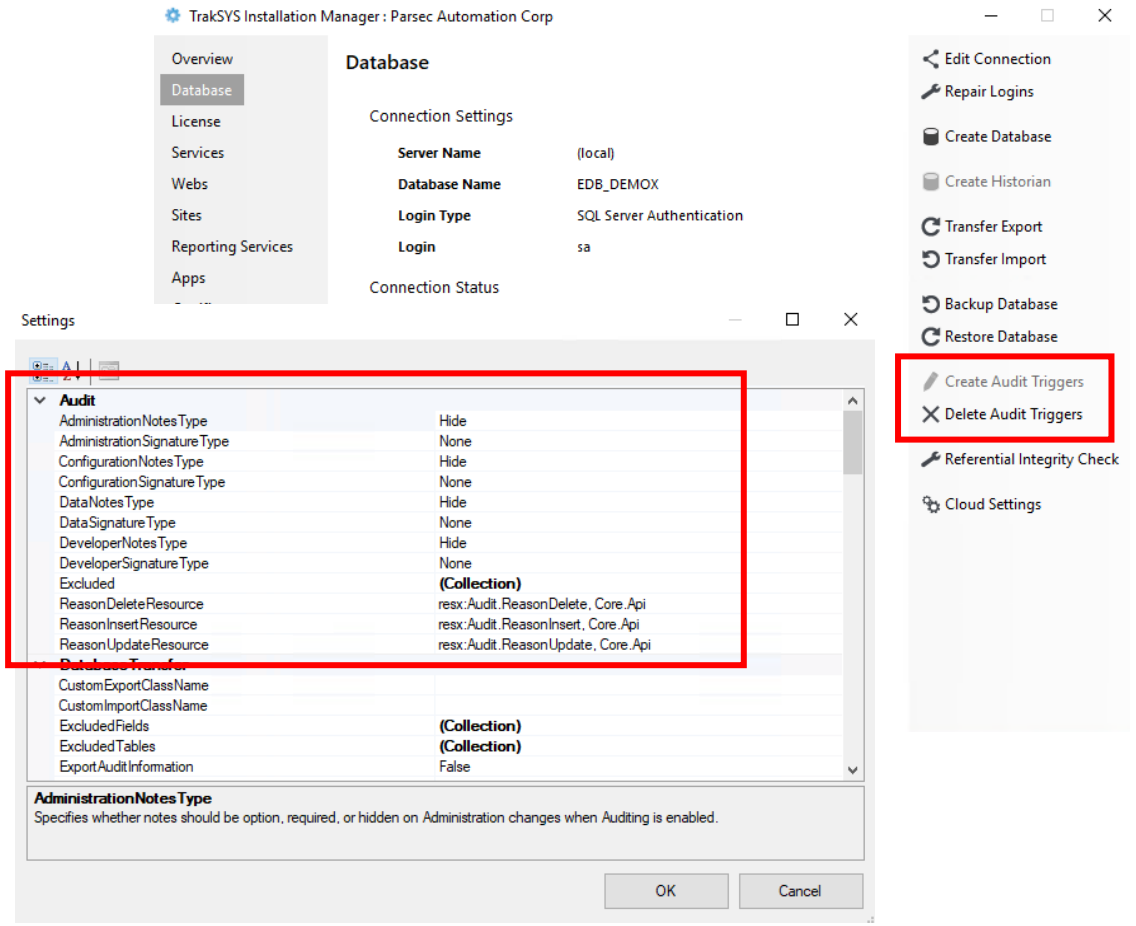
# Audit
# Overview

## Audit Features

The following are general capabilities available when the TrakSYS Audit feature is enabled...

- Requires a user digital signature any time a change is made to the Configuration or Data
- Configuration entities include an Audit link which displays related Change History
- Audit features can be configured in TrakSYS Settings within Installation Manager
- Required database Triggers can be managed from the Database section in Installation Manager



# Settings

## Audit Settings

**Control by Section**

- Administration
- Configuration
- Developer
- Data

**Signature Type**

- None
- Auto
- Name Only
- Single
- Dual

**Notes Type**

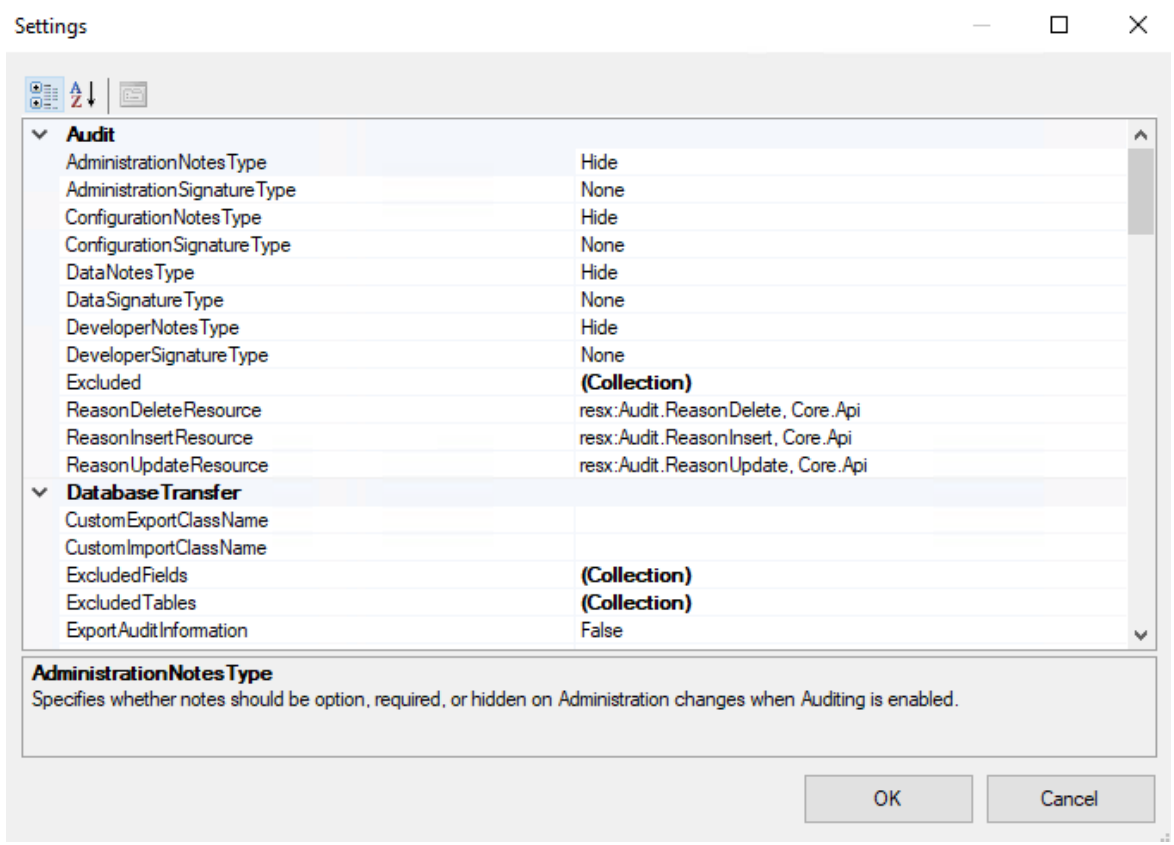
- Hide
- Optional
- Required

**Excluded Tables**

Table Names to be excluded from the Audit Triggers

**Reason Resources**

The text that is displayed when prompting for a Signature



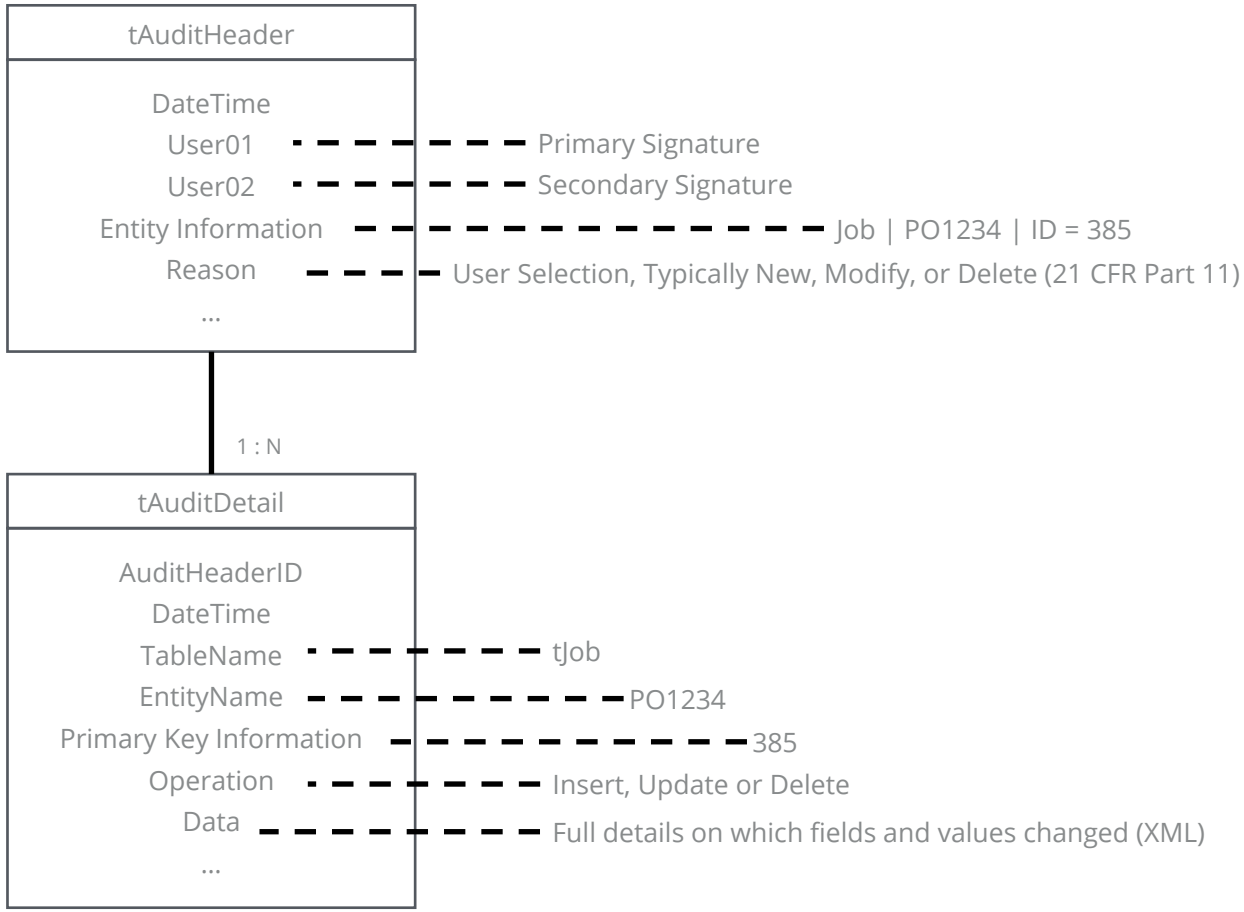
# Table Structure

## Header

An Audit Header record is created (1 per database transaction) by the TrakSYS API infrastructure when Auditing is enabled, and a change is made via the standard UI or using appropriate API techniques.

## Detail

An Audit Detail record is created (1 per insert, update or deleted row within a database transaction) by the TrakSYS Audit Triggers when they are deployed, and a change is made in any way (even direct to database).



# API Example

## Executing with Auditing

In this example, a multi-insert API call is made with support for creating the Audit Header and binding to it all resultant Audit Detail records.

The API calls shown here will create the following TrakSYS data records...

- tJob
- tJobDiscrete (related 1:1 to tJob)
- tJobSystemPlanned

The following Audit records will also be created...

- tAuditHeader
- tAuditDetail (tJob)
- tAuditDetail (tJobDiscrete)
- tAuditDetail (tJobSystemPlanned)

```
// note that proper error handling is omitted for clarity

var uow = this.Ets.Api.CreateUnitOfWork();

// load system
var sys = this.Ets.Api.Data.DbSystem.Load.ByID(23);

// create discrete job
var job = this.Ets.Api.Data.DbJobDiscreteComposite.Create.FromParentNone();
this.Ets.Api.Util.Db.SetNextSequenceToModel(job);
job.Name = "PO123";
// fill other job fields
this.Ets.Api.Data.DbJobDiscreteComposite.Save.InsertAsNew(job, uow);

// create job planned
var jsp = new ETS.Core.Api.Models.Data.DbJobSystemPlanned();
jsp.JobID = job.ID;
jsp.SystemID = sys.ID;
this.Ets.Api.Data.DbJobSystemPlanned.Save.InsertAsNew(jsp, uow);

// create header
var header = this.Ets.Api.Data.DbAuditHeader.Create.FromParentNone();
header.AuditDateTime = this.Ets.SiteNow;
header.User01 = this.Ets.User.DisplayName;
header.User01FullName = this.Ets.User.Login;
header.EntityID = job.ID;
header.EntityName = job.Name;
header.EntityTypeName = "Job";
header.Reason =
  this.Ets.Api.Util.Resource.GetEnumString(ETS.Core.Enums.AuditDetailOperationType.Insert);
header.IsDatabaseTransfer = false;

// execute with audit
uow.ExecuteAudit(header);
```

# Database Transfer

## Audit Transfer

For scenarios where Non-Production (Development) instances are in use, and the changes made there must be tracked, the ExportAuditInformation setting allows for the change details to be passed down to the Production instance.

Audit Records that are transferred will be marked in the Header as IsDatabaseTransfer = True. The original date/time, user and all details are retained.