TrakSYS™ Training

Day 4

Training Overview

Training Agenda

Day 1	Day 2	Day 3	Day 4	Day 5
TrakSYS Overview	Content Pages	Performance Management	API Introduction	Production Scheduling
Setup and Installation	Values Dictionary	Content Page Functionality	Logic Service	Alerts and Notifications
Configuration Basics	Visual Pages	Batching and Storage Systems	Data Management Service	Inventory Management
Navigation Introduction	Content Parts and Features	Template Systems	TrakSYS Extensibility	Statistical Process Control
Functionality and Data	Users and Permissions	Task Configuration	Sites, Translations, and Audit	Support and Resources

Introduction Training

Advanced Training

Comprehensive Training

Application Programming Interface

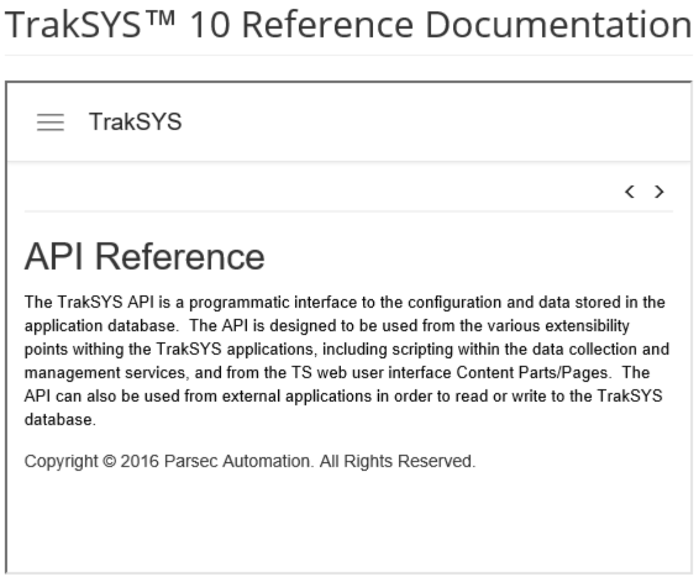
API

Training Objectives

Understand the basic structure and capabilities of the TrakSYS Application Programming Interface (API).

Explore a simple API script to load, modify and save a data entity in the TrakSYS database.

TrakSYS API



TrakSYS™ 10 Reference Documentation

API Reference

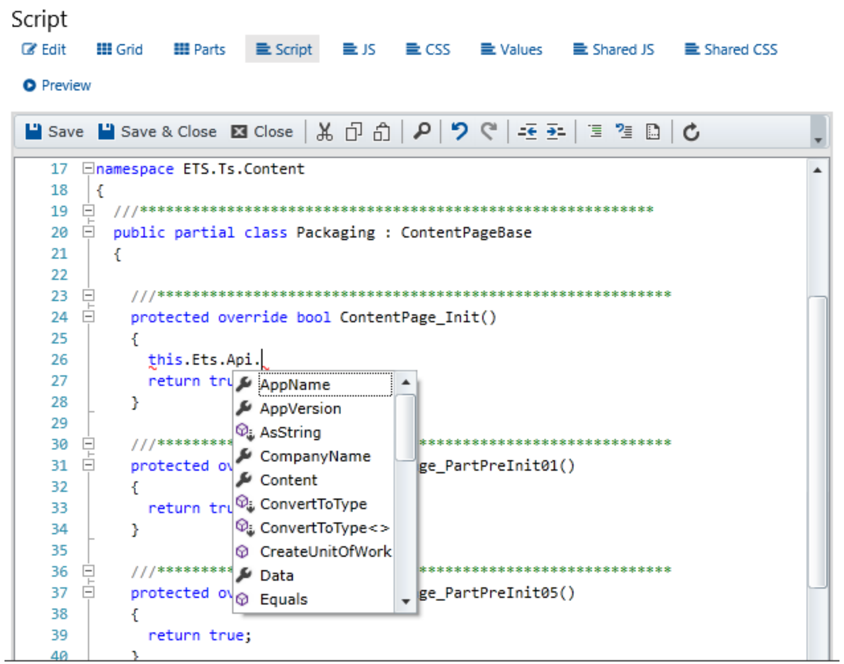
The TrakSYS API is a programmatic interface to the configuration and data stored in the application database. The API is designed to be used from the various extensibility points withing the TrakSYS applications, including scripting within the data collection and management services, and from the TS web user interface Content Parts/Pages. The API can also be used from external applications in order to read or write to the TrakSYS database.

Copyright © 2016 Parsec Automation. All Rights Reserved.

- Publicly supported set of Classes and Methods used to create custom functionality for TrakSYS and to integrate with External Applications
- Accessible through C#.NET code or via Web Services
- ETS.Core.Api namespace within the TrakSYS Core Library (Core.dll) contains all classes within the API
- ETS.Core.Api.ApiService is the primary C#.NET class used to access the API Service Objects
- DOCS available on the TrakSYS Support Site

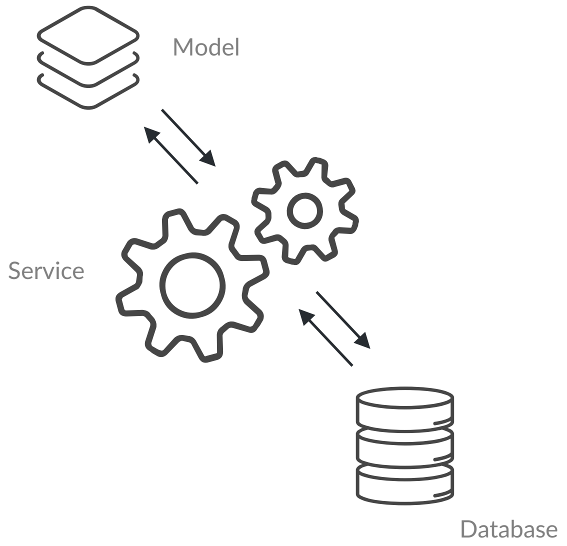
API Availability

- Logic Service
  - Advanced Script Tags
  - Logic Service Script Classes
  - SPC Rule Definitions
- Data Management Service
  - Module Script Steps
- TrakSYS Web
  - Content Pages and Parts
- Externally
  - As a .NET DLL Reference
- Instantiated and made available from within TrakSYS script Editors



```
namespace ETS.Ts.Content
{
  ///****************************************************************
  public partial class Packaging : ContentPageBase
  {

    ///****************************************************************
    protected override bool ContentPage_Init()
    {
      this.Ets.Api.
      return tru
    }

    ///*******                          ****************************
    protected ov                        ge_PartPreInit01()
    {
      return tru
    }

    ///*******                          ****************************
    protected ov                        ge_PartPreInit05()
    {
      return true;
    }
```

Models and Services



The API contains two different types of Classes

- Model classes are data structures only. They typically represent and mirror tables in the TrakSYS database.
  - tSystem = DbSystem
  - tEvent = DbEvent
- Service classes take Model classes and perform actions on them, typically inserting, updating and deleting the related records from the TrakSYS database.

Model and Service Example

Model
ETS.Core.Api.Models.Data.DbEvent

- ID
- StartDateTime
- EndDateTime
- EventDefinitionID
- …
- Notes

Services
ETS.Core.Api.Data.DbEvent

- .Load.ByID(int)
- .Save.InsertAsNew(DbEvent)
- .Save.UpdateExisting(DbEvent)
- .Delete.ByID(int)

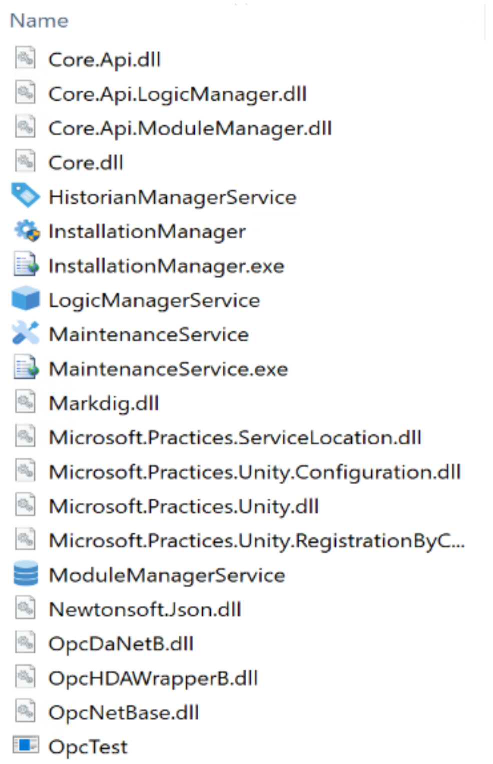
API Example

- Example of loading an Event model object, changing the Notes field, and then saving it back to the database.
- Common patterns for loading, inserting, updating and deleting can be used on any TrakSYS database entity.

```
// get a reference to the api service
ETS.Core.Api.ApiService api = ETS.Core.Api.ApiService.GetInstance();

// create a model object to hold the results of the load
ETS.Core.Api.Models.Data.DbEvent ev;

// load the entity with ID 123 from the database
ev = api.Data.DbEvent.Load.ByID(123);

// modify the properties of the model object as needed
ev.Notes = "new notes have been added";
// etc...

// create a result object to determine the success of the operation
ETS.Core.Api.Models.Result<ETS.Core.Api.Models.Data.DbEvent> result;

// update the entity in the database
result = api.Data.DbEvent.Save.UpdateExisting(ev);

// examine the results of the operation
if (result.Success)
// etc...
```

Referencing DLLs

- Support for referencing .NET assemblies written and compiled outside of the TrakSYS Environment (DLLs)
- TrakSYS .NET script can consume public classes and code from these referenced DLLs
- DLLs must be placed in specific TrakSYS folder locations (depending on the scripting application)
  - Logic Service
  - Data Management Service
  - TrakSYS Web



Name
Core.Api.dll
Core.Api.LogicManager.dll
Core.Api.ModuleManager.dll
Core.dll
HistorianManagerService
InstallationManager
InstallationManager.exe
LogicManagerService
MaintenanceService
MaintenanceService.exe
Markdig.dll
Microsoft.Practices.ServiceLocation.dll
Microsoft.Practices.Unity.Configuration.dll
Microsoft.Practices.Unity.dll
Microsoft.Practices.Unity.RegistrationByC...
ModuleManagerService
Newtonsoft.Json.dll
OpcDaNetB.dll
OpcHDAWrapperB.dll
OpcNetBase.dll
OpcTest

Common Scripting Examples

Training Objectives

To better understand use cases and examples for the different scripting services.

Be able to understand the common functionalities that are handled by the different API services.

Data Service

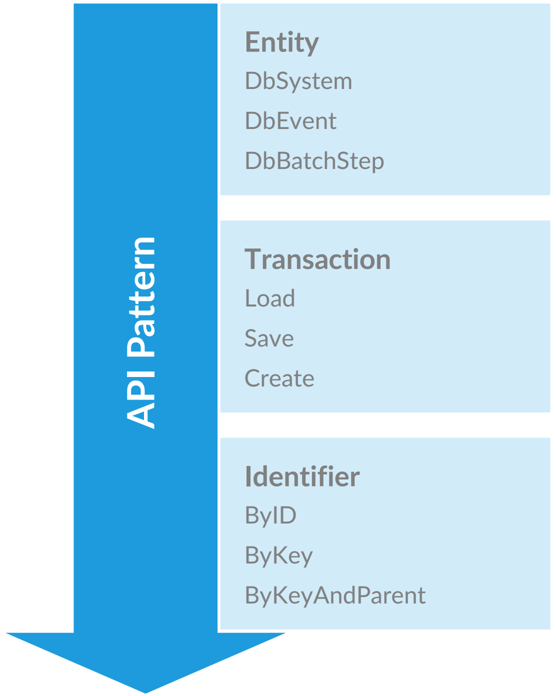
Api.Data

Facilitates structured interactions with the Database through script.

- Every table in the database has matching API and models.
- Some more complex models may have Composite objects that allow for loading and saving of two records at once.

Examples:

- Api.Data.DbEvent.Delete.ByID(int ID)
- Api.Data.DbSystem.GetList.ForAreaID(int AreaID)
- Api.Data.DbProduct.Create.FromParentProductGroup(DbProductGroup parent)
- Api.Data.DbMaterial.Load.WithSql(string Sql)
- Api.Data.DbOeeCalculation.Save.UpdateExisting(DbOeeCalculation item)
- Api.Data.DbJobDiscreteComposite.Load.ByID(int ID)



Common Entity Services

Api.Events – Api.Tags – Api.Tasks – Etc.

Services exist to assist with functionality that utilize multiple tables at once.

- Includes special models specifically for the intended functionality
- Includes specialized calls for data loading, processing, and support

Examples:

- Api.Events.CalculateJobID(DbEvent ev);
- Api.Historian.GetTagHistory([...]);
- Api.Kpi.LoadOeeDataByProductionDateRange([...]);
- Api.Notification.CreateAlertNotificationForLogin(login, title, body);
- Api.ProdSched.Schedule(settings);
- Api.Spc.Variable.CalculateStandardDeviationS(List<double> values);
- Api.Tags.GetList.ForTagNames(List<string> names);
- Api.Tasks.CreateFromParentTaskDefinitionID(int TaskDefinitionID);

Tasks

Load Task Items

Load Task Form Items

Create Task Sink

KPI

Load for Time Range

Create Adjustment

Estimate Job End

Tags

Load by Name

Get Dictionary from List

Update List of Virtual Tags

Utilities and Util Service

General and TS Web Specific Utilities

- General Util services exist to assist with troubleshooting and logging
- Based upon the scripting location, additional information is exposed
- TS Web utilities include web-specific troubleshooting and user information

Examples:

- Api.Util.Db.ExecuteSql(sql);
- Api.Util.Log.WriteInformation(message, category);
- Api.Util.LogCustom.WriteWarningsFromResultObject(result, category);
- Api.Site.GetCurrentSiteID();
- Debug.Trace(message);
- Device.Name();
- User.DisplayName();

Logic Service
Tags Collection
Post Scan Context
Execution Times

Web
Various Dictionaries
User and Device Information
Parts and Styling Access

Data Management Service
Module Information
Args Dictionary
Last Step Information

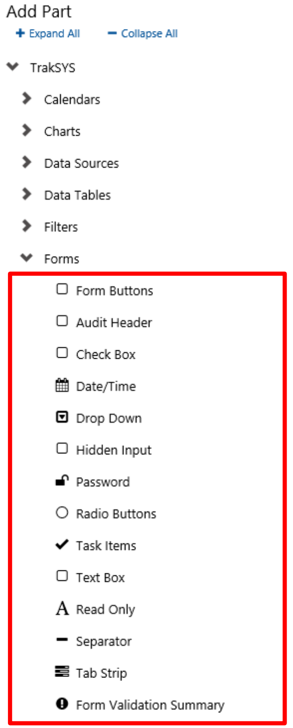
Form Development

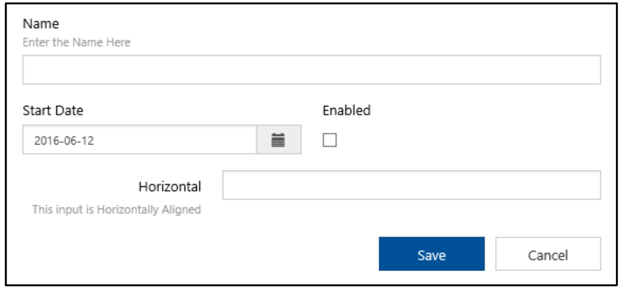
Training Objectives

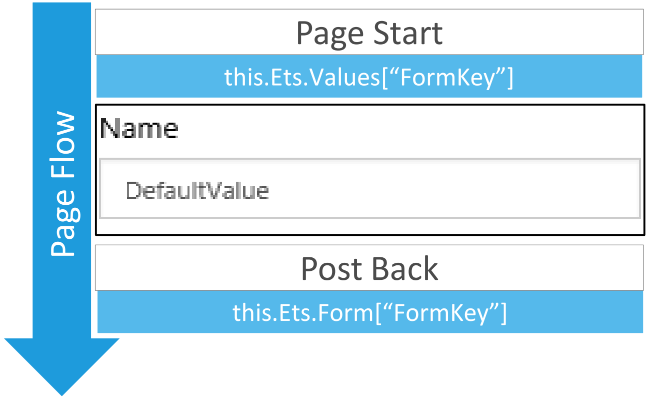
Understand the page lifecycle and key steps in creating a data entry/edit form using a Visual Page Definition.

Describe the available Page API methods for loading and saving data in a TrakSYS form.

Form Content Parts



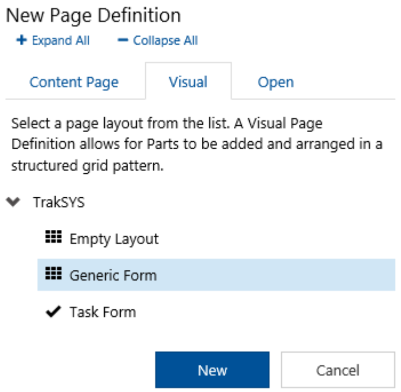




- Form Parts can be added and arranged to Visual Pages
- Configurable properties such as…
  - Caption
  - Sub-Caption
  - Widths
  - Form Key Mapping
- Inputs can be configured for Vertical or Horizontal Layout
- Buttons Part provides common Form Operations
- Standard TrakSYS Styling

Generic Form Visual Page Template

- Visual Page template for rapid development of a data entry / edit Form
- Form Layout
  - Tabs
  - Inputs
  - Validation
  - Buttons
- Script Class
  - Skeleton Script
  - Comments



New Page Definition

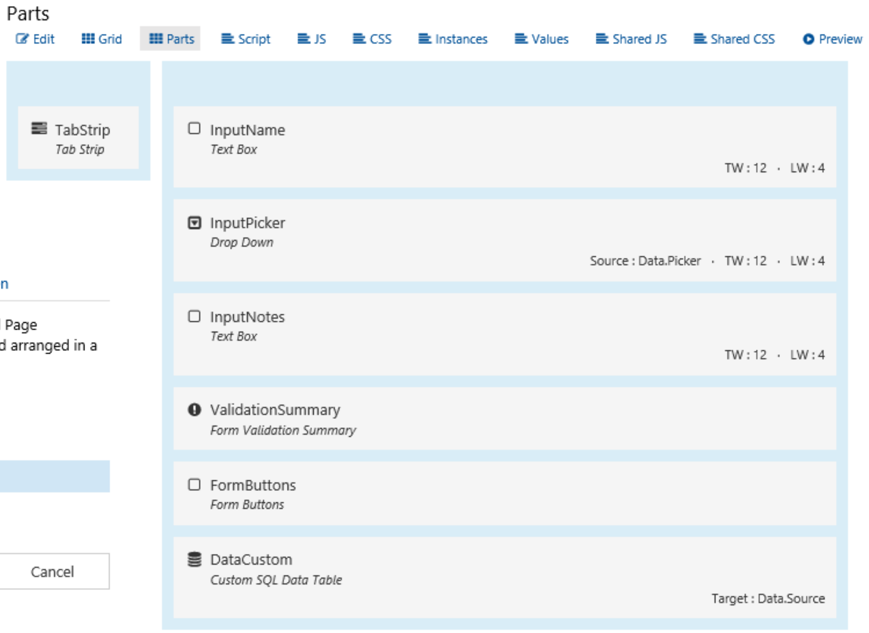
Expand All · Collapse All

Content Page | Visual | Open

Select a page layout from the list. A Visual Page Definition allows for Parts to be added and arranged in a structured grid pattern.

- TrakSYS
  - Empty Layout
  - Generic Form
  - Task Form

New · Cancel



Parts

Edit · Grid · Parts · Script · JS · CSS · Instances · Values · Shared JS · Shared CSS · Preview

TabStrip
Tab Strip

InputName
Text Box
TW : 12 · LW : 4

InputPicker
Drop Down
Source : Data.Picker · TW : 12 · LW : 4

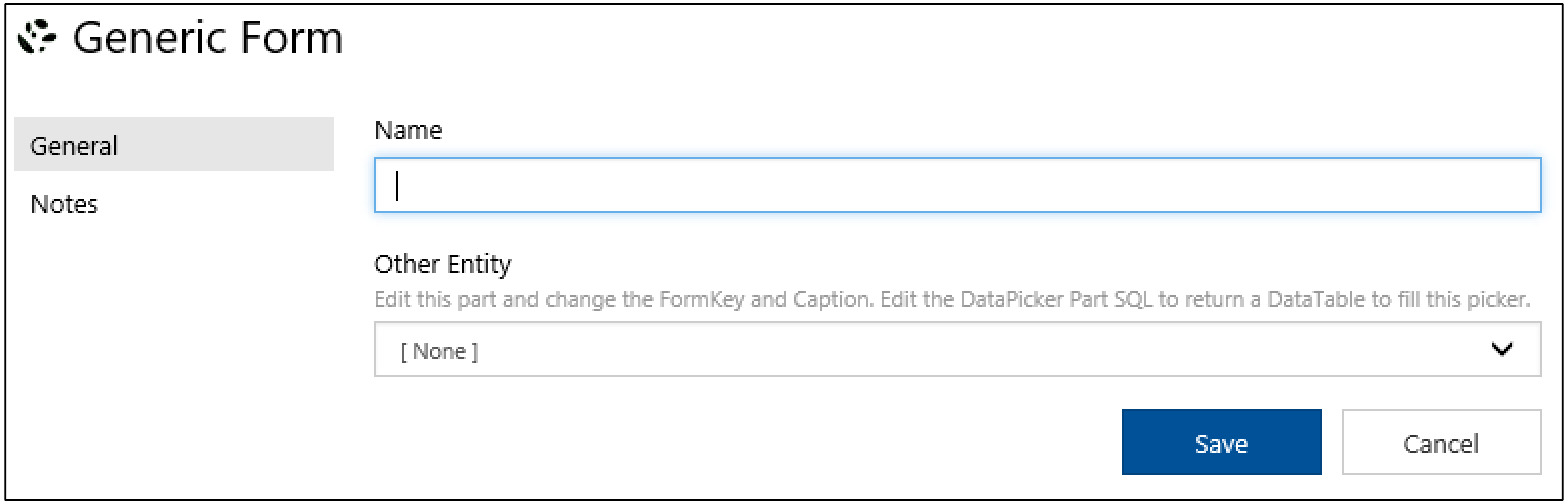
InputNotes
Text Box
TW : 12 · LW : 4

ValidationSummary
Form Validation Summary

FormButtons
Form Buttons

DataCustom
Custom SQL Data Table
Target : Data.Source

Generic Form Visual Page Template



Name

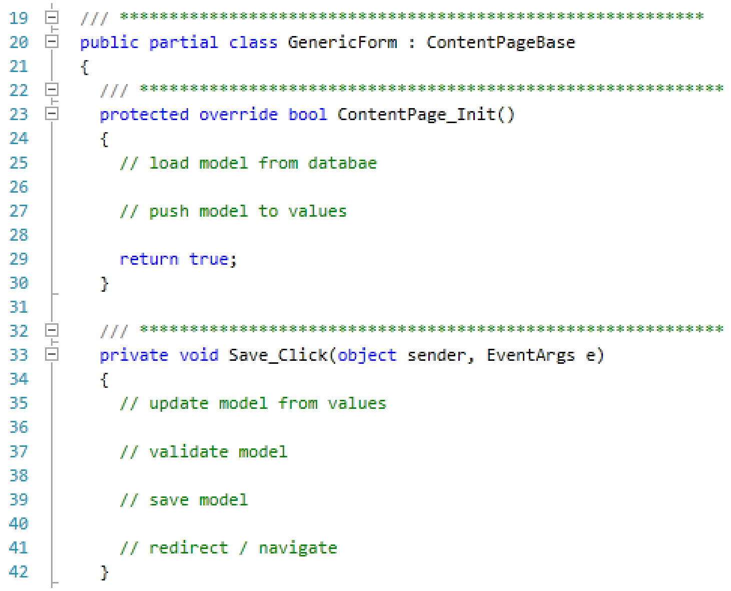
Other Entity

Edit this part and change the FormKey and Caption. Edit the DataPicker Part SQL to return a DataTable to fill this picker.

[None]

Form Lifecycle

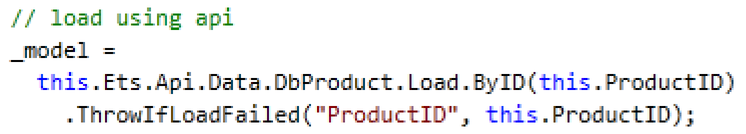
- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate



```
/// ***********************************************************
public partial class GenericForm : ContentPageBase
{
  /// ***********************************************************
  protected override bool ContentPage_Init()
  {
    // load model from databae

    // push model to values

    return true;
  }

  /// ***********************************************************
  private void Save_Click(object sender, EventArgs e)
  {
    // update model from values

    // validate model

    // save model

    // redirect / navigate
  }
```

Load Data to Model

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate



```
// load using api
_model =
  this.Ets.Api.Data.DbProduct.Load.ByID(this.ProductID)
    .ThrowIfLoadFailed("ProductID", this.ProductID);
```

Push Model to Values

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate

```
// copy model to values
if (!this.Ets.Values.CopyFromModel(_model, "Model."))
  return false;
```

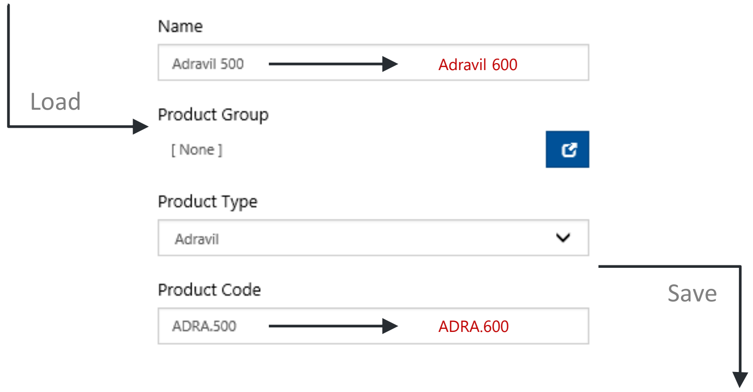
$\downarrow$

- Model.ID = 23
- Model.Name = “Adravil”;
- Model.ProductCode = “ADRA.500”;
- …

TS Web Interaction

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate

Model.Name = “Adravil 500”
Model.ProductGroupID = -1
Model.ProductTypeID = 3
Model.ProductCode = “ADRA.500”



Model.Name = “Adravil 600”
Model.ProductGroupID = -1
Model.ProductTypeID = 3
Model.ProductCode = “ADRA.600”

Update Model from Values

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate

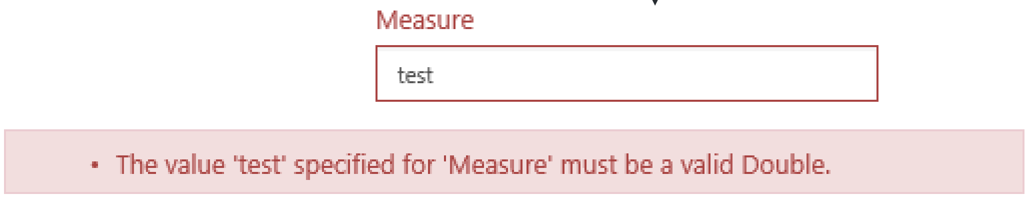
- Model.ID = 23
- Model.Name = “Adravil”;
- Model.ProductCode = “ADRA.500”;
- …

```
// update model from values
if (!this.Ets.Form.UpdateModelWithKeyPrefix(_model, "Model."))
  return;
```

Validate the Model

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate

```
// validate the model
var coreValidate =
  this.Ets.Api.Data.DbProduct.ValidateForMerge(_model, this.IsNew);

// send validation results to ui
if (!this.Ets.Form.AddResultMessagesWithPrefixIfFailed(
  coreValidate, "Model.")
  ) return false;
```



Measure

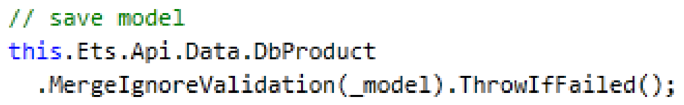
test

- The value 'test' specified for 'Measure' must be a valid Double.

Save Model

Single-Model Commit

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate



```
// save model
this.Ets.Api.Data.DbProduct
  .MergeIgnoreValidation(_model).ThrowIfFailed();
```

Unit of Work

Multi-Model Commit

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate

```
// create unit of work
var uow = this.Ets.Api.CreateUnitOfWork();

// queue models
this.Ets.Api.Data.DbProduct
  .MergeIgnoreValidation(_model1, isNew, uow).ThrowIfFailed();
this.Ets.Api.Data.DbProduct
  .MergeIgnoreValidation(_model2, isNew, uow).ThrowIfFailed();
this.Ets.Api.Data.DbProduct
  .MergeIgnoreValidation(_model3, isNew, uow).ThrowIfFailed();

// execute saves
var result = uow.ExecuteReturnsResultObject();

// process result
if(!result.Success)
{
  this.Ets.Debug.FailFromResultMessages(result.Messages);
}
```

Redirect / Navigate

- Load data from the Database to Model
- Push Model to Values Dictionary
- Populate Controls from the Values Dictionary
- User interacts with Form and Saves
- Populate the Values Dictionary from the Controls
- Update Model from the Values Dictionary
- Validate the Model
- Save the Model to the Database
- Redirect / Navigate

```
// redirect to success
this.Ets.Pages.RedirectToSuccessUrl();

// redirect to the current "folder" page
this.Ets.Pages.RedirectToFolderUrl();

// redirect to the current "spoke" page
this.Ets.Pages.RedirectToPageUrl();

// redirect to some specific page
this.Ets.Pages.RedirectToUrl("../SomePageKey");
```

Demonstration

- Write a simple example of Loading an entity in TS Web
- View the properties of a TS entity
- Create a Generic Form
- Examine the Script
- Demonstrate Form Key Mapping

Lab 13

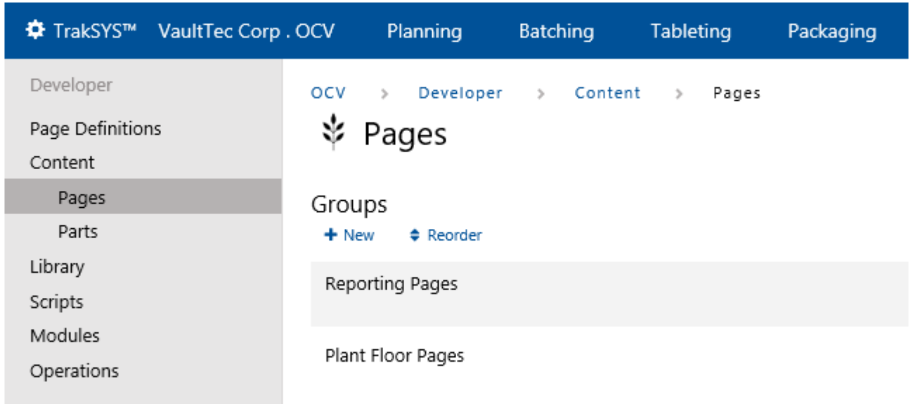
Library Pages and Parts

Training Objectives

Explain the concepts and techniques behind creating re-usable Pages and Parts using the Open Page Definition Type and Content Library.

Learn how to expose Parameters so developers can configure and direct the behavior of re-useable Pages and Parts.

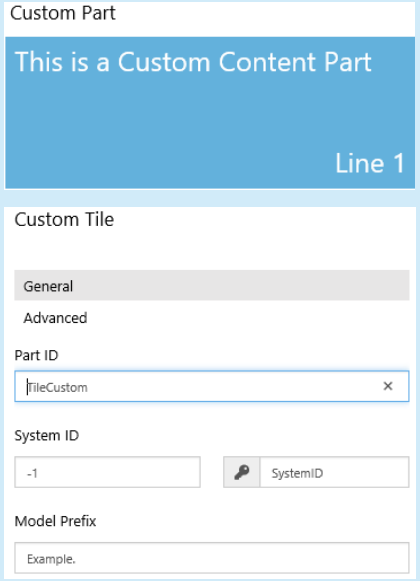
Content Library



- Re-useable Pages and Parts that can be developed for a specific Implementation
- Stored in the TrakSYS Database
- Available in the standard Page and Part Catalog Pickers
- Can be Parameterized (allowing configuration / settings)

Content Pages and Parts Use Cases

Reimagined Content Parts with New Functionality



Custom Part

This is a Custom Content Part

Line 1

Custom Tile

General

Advanced

Part ID

TileCustom

System ID

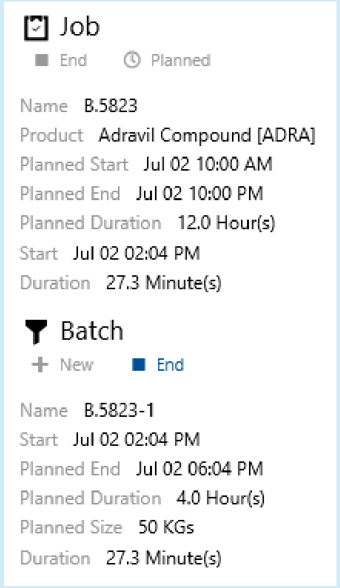
-1

SystemID

Model Prefix

Example.

Standardize Multi-Part Layout and Logic



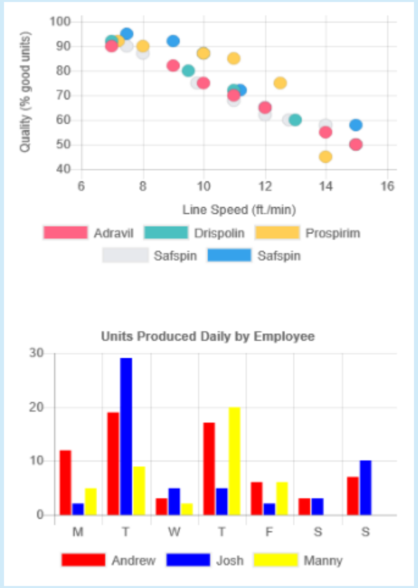
Job

Name	B.5823
Product	Adravil Compound [ADRA]
Planned Start	Jul 02 10:00 AM
Planned End	Jul 02 10:00 PM
Planned Duration	12.0 Hour(s)
Start	Jul 02 02:04 PM
Duration	27.3 Minute(s)

Batch

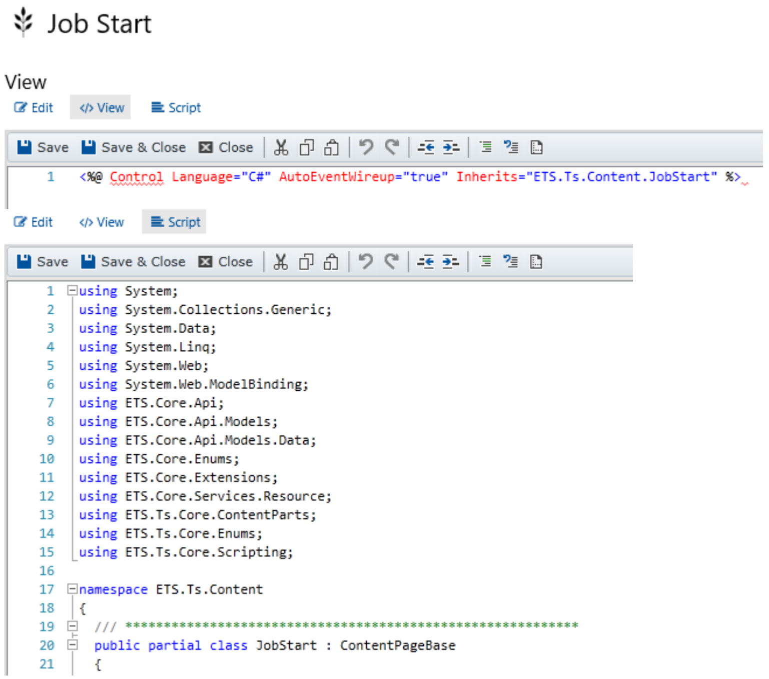
Name	B.5823-1
Start	Jul 02 02:04 PM
Planned End	Jul 02 06:04 PM
Planned Duration	4.0 Hour(s)
Planned Size	50 KGs
Duration	27.3 Minute(s)

Standardized External Components



Open Content Pages

- **Front-End (HTML)**
  Uses an open HTML editor.
  Any valid HTML Tags or ASP.NET User Controls can be arranged.
- **Back-End (Script)**
  Uses the same Script editor and page lifecycle as Visual Page Definitions.
- May be instanced as Content Page Definitions, or called directly using C Equals URL Syntax

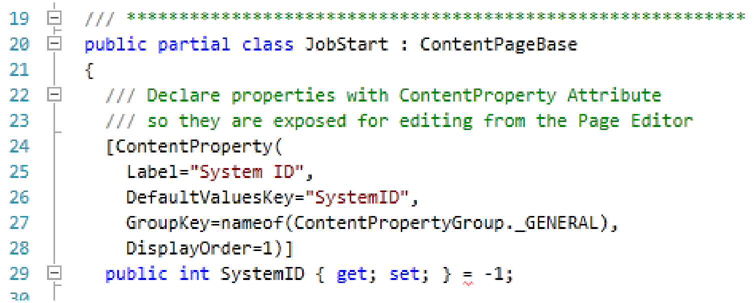


```
<%@ Control Language="C#" AutoEventWireup="true" Inherits="ETS.Ts.Content.JobStart" %>
```

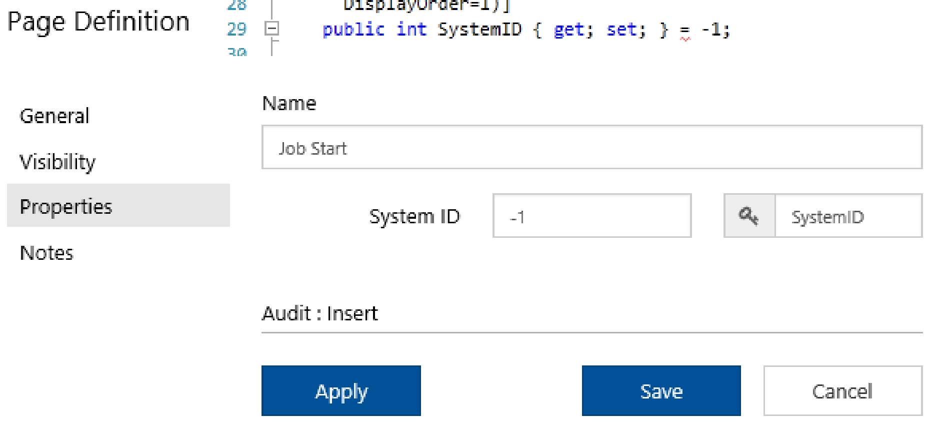
```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.ModelBinding;
using ETS.Core.Api;
using ETS.Core.Api.Models;
using ETS.Core.Api.Models.Data;
using ETS.Core.Enums;
using ETS.Core.Extensions;
using ETS.Core.Services.Resource;
using ETS.Ts.Core.ContentParts;
using ETS.Ts.Core.Enums;
using ETS.Ts.Core.Scripting;

namespace ETS.Ts.Content
{
  /// ***********************************************************
  public partial class JobStart : ContentPageBase
  {
```

Content Page Properties

- Properties defined in Script that are exposed to Developers for Configuration
- Special .NET Attribute Decoration
- Exposed via the standard Page Definition Properties Interface
- Automatically mapped from the Values Dictionary



```
/// **********************************************************
public partial class JobStart : ContentPageBase
{
  /// Declare properties with ContentProperty Attribute
  /// so they are exposed for editing from the Page Editor
  [ContentProperty(
    Label="System ID",
    DefaultValuesKey="SystemID",
    GroupKey=nameof(ContentPropertyGroup._GENERAL),
    DisplayOrder=1)]
  public int SystemID { get; set; } = -1;
```



Page Definition

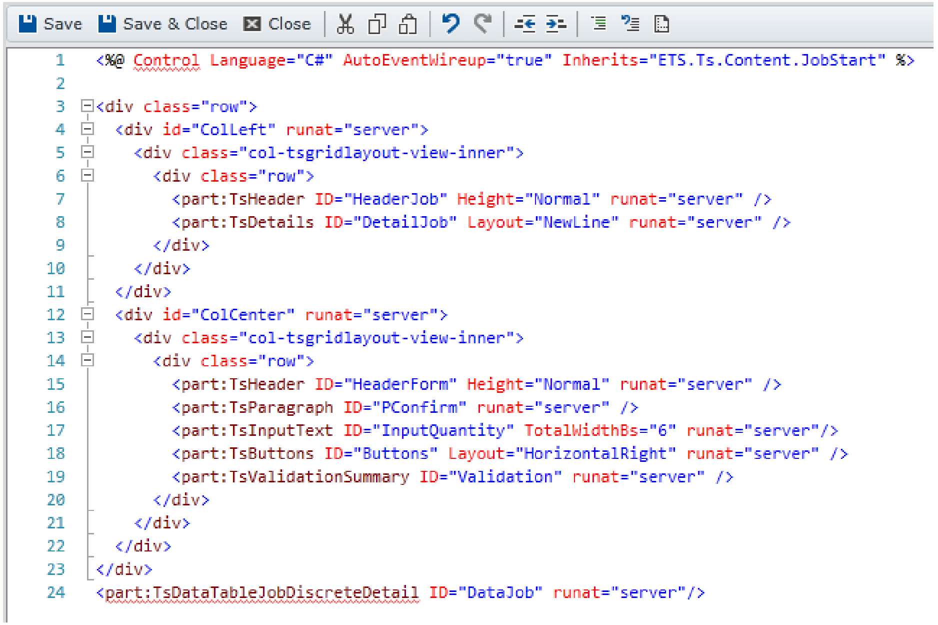
Name: Job Start

System ID: -1 SystemID

Audit : Insert

Open Content Page Layout

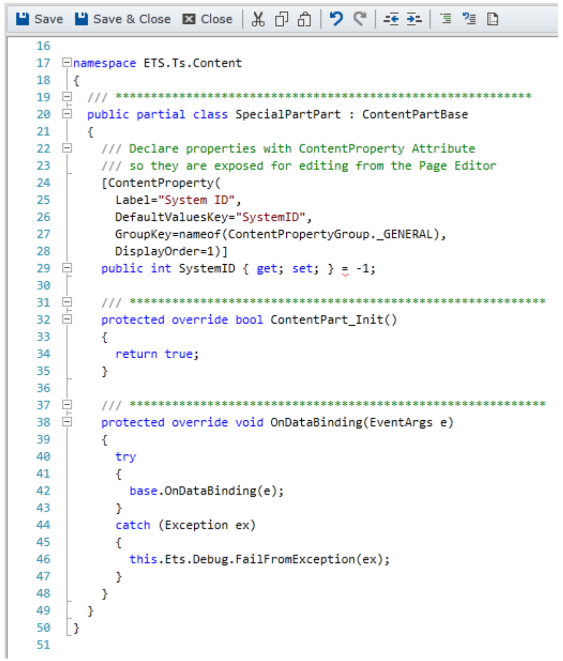
- Supports HTML (5) Syntax
- Supports ASP.NET Controls
- Support TrakSYS Parts
    - <part:TsPartName ID="PartID" Property="Value" … />
- Recommend creating a Grid layout using a Visual Page Definition



```
<%@ Control Language="C#" AutoEventWireup="true" Inherits="ETS.Ts.Content.JobStart" %>

<div class="row">
  <div id="ColLeft" runat="server">
    <div class="col-tsgridlayout-view-inner">
      <div class="row">
        <part:TsHeader ID="HeaderJob" Height="Normal" runat="server" />
        <part:TsDetails ID="DetailJob" Layout="NewLine" runat="server" />
      </div>
    </div>
  </div>
  <div id="ColCenter" runat="server">
    <div class="col-tsgridlayout-view-inner">
      <div class="row">
        <part:TsHeader ID="HeaderForm" Height="Normal" runat="server" />
        <part:TsParagraph ID="PConfirm" runat="server" />
        <part:TsInputText ID="InputQuantity" TotalWidthBs="6" runat="server"/>
        <part:TsButtons ID="Buttons" Layout="HorizontalRight" runat="server" />
        <part:TsValidationSummary ID="Validation" runat="server" />
      </div>
    </div>
  </div>
</div>
<part:TsDataTableJobDiscreteDetail ID="DataJob" runat="server"/>
```

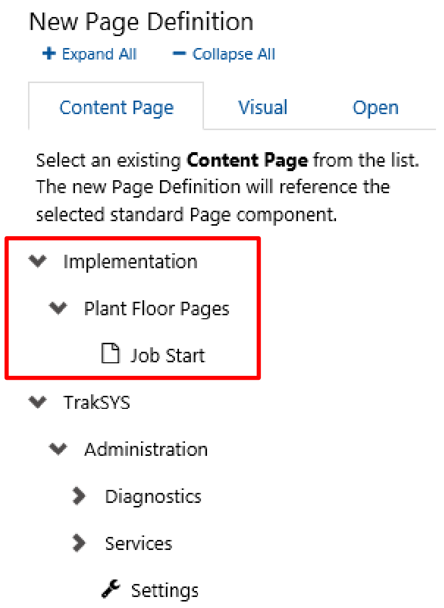
Re-Useable Content Parts

- Similar to Open Pages
- Includes Content Property Mapping
- **Front-End (HTML)**
  Uses an open HTML editor. Any valid HTML Tags or ASP.NET User Controls can be arranged.
- **Back-End (Script)**
  Define script that is run when the Part is loaded and rendered. A slightly different lifecycle that the Page Definitions.



```

namespace ETS.Ts.Content
{
  /// ****************************************************************
  public partial class SpecialPartPart : ContentPartBase
  {
    /// Declare properties with ContentProperty Attribute
    /// so they are exposed for editing from the Page Editor
    [ContentProperty(
      Label="System ID",
      DefaultValuesKey="SystemID",
      GroupKey=nameof(ContentPropertyGroup._GENERAL),
      DisplayOrder=1)]
    public int SystemID { get; set; } = -1;

    /// ****************************************************************
    protected override bool ContentPart_Init()
    {
      return true;
    }

    /// ****************************************************************
    protected override void OnDataBinding(EventArgs e)
    {
      try
      {
        base.OnDataBinding(e);
      }
      catch (Exception ex)
      {
        this.Ets.Debug.FailFromException(ex);
      }
    }
  }
}

```

Using Library Pages and Parts

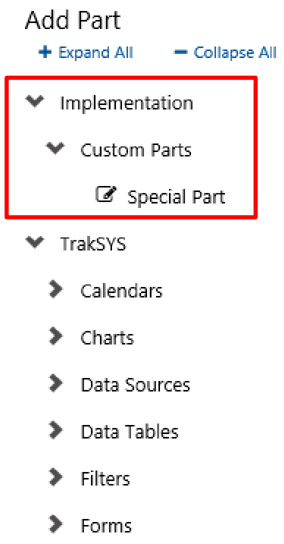

New Page Definition

+ Expand All − Collapse All

Content Page | Visual | Open

Select an existing **Content Page** from the list. The new Page Definition will reference the selected standard Page component.

- Implementation
  - Plant Floor Pages
    - Job Start
- TrakSYS
  - Administration
    - Diagnostics
    - Services
    - Settings


Add Part

+ Expand All − Collapse All

- Implementation
  - Custom Parts
    - Special Part
- TrakSYS
  - Calendars
  - Charts
  - Data Sources
  - Data Tables
  - Filters
  - Forms

- Content Library Pages and Parts are available in the standard Page and Part Catalog Pickers
- Displayed at the top of the Picker under the Implementation Group
- Can be added to the Page Definition hierarchy or called directly using "C Equals"
- Catalog Picker Tree includes Compiled Pages and Parts

Custom Types, Properties and Permissions

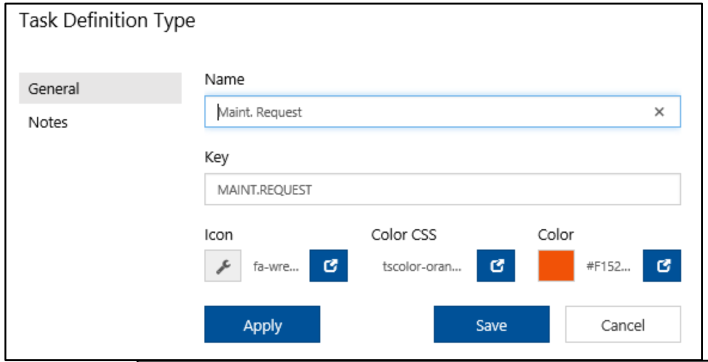
Training Objectives

Explore the extensibility of the TrakSYS configuration.

Understand the configuration and use of Custom Types, Custom Properties and Custom Permissions, and how to reference them within the API.

Custom Types

- A Custom Type is an entity-specific, user-defined Enum list.
- Custom Types have an ID and key that can be referenced when writing implementation-specific logic.
- Once configured, Custom Types can be selected from their entity's standard property page.
- Loading and filtering by Custom Types is supported in standard Data Providers and API.



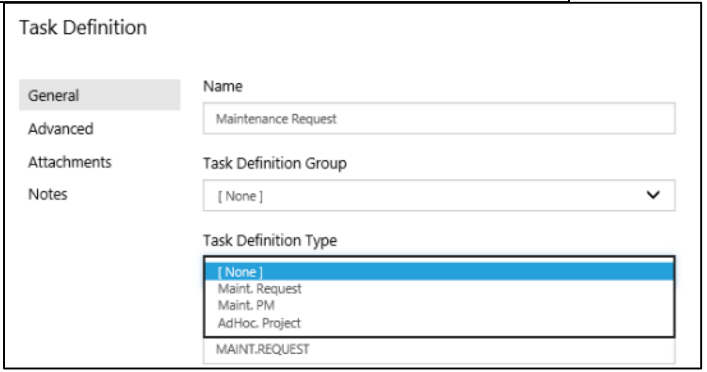
Task Definition Type

General
Notes

Name: Maint. Request

Key: MAINT.REQUEST

Icon: fa-wre... Color CSS: tscolor-oran... Color: #F152...



Task Definition

General
Advanced
Attachments
Notes

Name: Maintenance Request

Task Definition Group: [None]

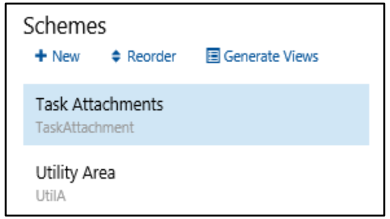
Task Definition Type:
[None]
Maint. Request
Maint. PM
AdHoc. Project

MAINT.REQUEST

Custom Properties

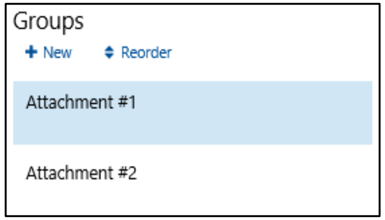
Once configured, Custom Properties appear as additional tabs in the assigned Property Pages.

Schemes identify the Name that will be used in the configuration and the Key Prefix that will be used in the API



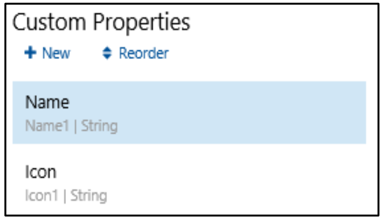
Task Attachments
TaskAttachment
Utility Area
UtilA

Groups identify the header for a collection of Custom Properties within the Scheme



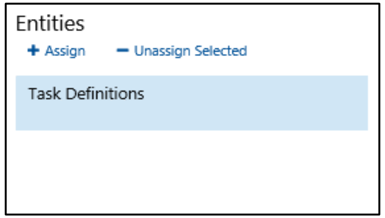
Attachment #1
Attachment #2

Custom Properties identify the new attributes as well as acceptable data structures, and the Key Suffix for the API



Name
Name1 | String
Icon
Icon1 | String

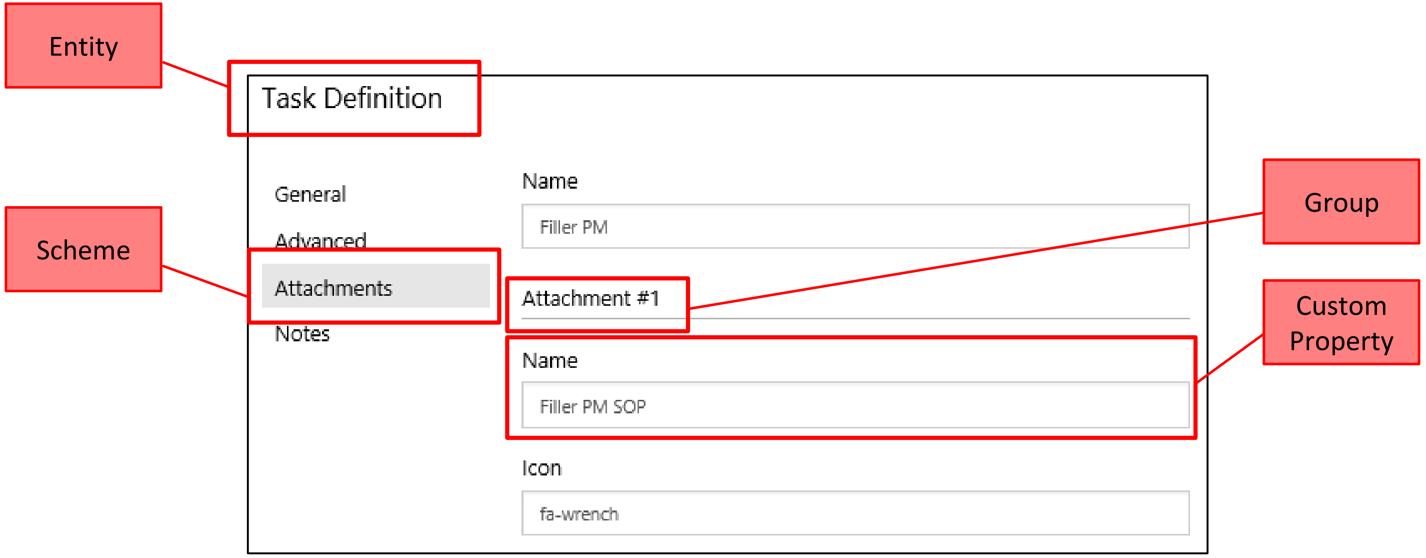
Entities identify where the scheme will appear in the standard configuration pages.



Task Definitions

Custom Properties

A Custom Property is set of configured attributes that are associated with one or more entities



Task Definition

General
Advanced
Attachments
Notes

Name
Filler PM

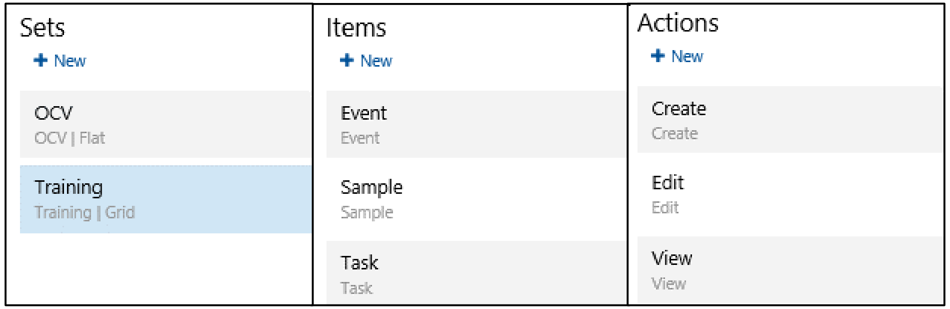
Attachment #1

Name
Filler PM SOP

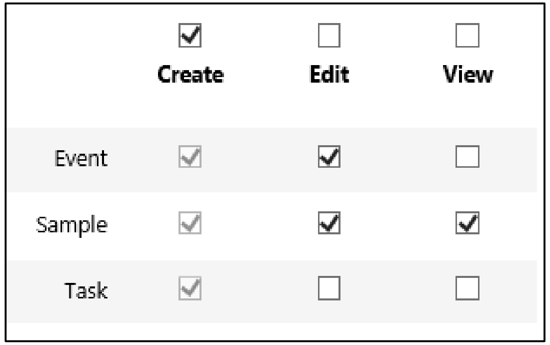
Icon
fa-wrench

Custom Permission Sets

- A Custom Permission Set is a user-defined collection of Role capabilities.
- Custom Permissions can be either Flat (with only Items) or Grid (with Items and Actions)
- Once configured, Custom Permissions appear as an additional tab when configuring Roles.
- Evaluating Custom Permissions is supported with the API.



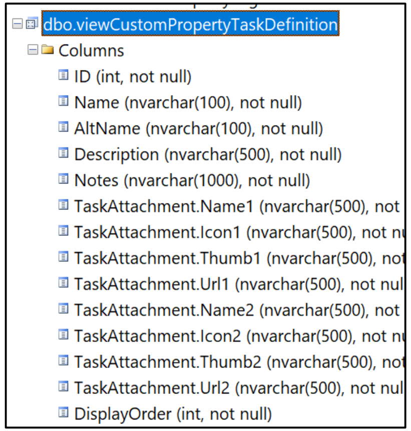
Sets	Items	Actions
+ New	+ New	+ New
OCV OCV | Flat	Event Event	Create Create
Training Training | Grid	Sample Sample	Edit Edit
	Task Task	View View



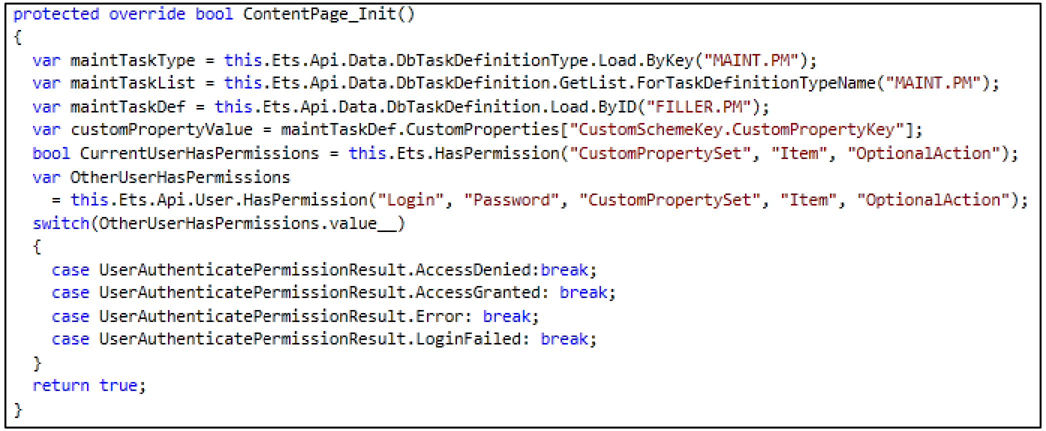
	✓ Create	☐ Edit	☐ View
Event	✓	✓	☐
Sample	✓	✓	✓
Task	✓	☐	☐

Referencing Custom Configuration

Custom configuration can be referenced with both the API and with SQL



```
⊟ dbo.viewCustomPropertyTaskDefinition
  ⊟ Columns
      ID (int, not null)
      Name (nvarchar(100), not null)
      AltName (nvarchar(100), not null)
      Description (nvarchar(500), not null)
      Notes (nvarchar(1000), not null)
      TaskAttachment.Name1 (nvarchar(500), not
      TaskAttachment.Icon1 (nvarchar(500), not nu
      TaskAttachment.Thumb1 (nvarchar(500), not
      TaskAttachment.Url1 (nvarchar(500), not nul
      TaskAttachment.Name2 (nvarchar(500), not
      TaskAttachment.Icon2 (nvarchar(500), not nu
      TaskAttachment.Thumb2 (nvarchar(500), not
      TaskAttachment.Url2 (nvarchar(500), not nul
      DisplayOrder (int, not null)
```



```
protected override bool ContentPage_Init()
{
  var maintTaskType = this.Ets.Api.Data.DbTaskDefinitionType.Load.ByKey("MAINT.PM");
  var maintTaskList = this.Ets.Api.Data.DbTaskDefinition.GetList.ForTaskDefinitionTypeName("MAINT.PM");
  var maintTaskDef = this.Ets.Api.Data.DbTaskDefinition.Load.ByID("FILLER.PM");
  var customPropertyValue = maintTaskDef.CustomProperties["CustomSchemeKey.CustomPropertyKey"];
  bool CurrentUserHasPermissions = this.Ets.HasPermission("CustomPropertySet", "Item", "OptionalAction");
  var OtherUserHasPermissions
    = this.Ets.Api.User.HasPermission("Login", "Password", "CustomPropertySet", "Item", "OptionalAction");
  switch(OtherUserHasPermissions.value__)
  {
    case UserAuthenticatePermissionResult.AccessDenied:break;
    case UserAuthenticatePermissionResult.AccessGranted: break;
    case UserAuthenticatePermissionResult.Error: break;
    case UserAuthenticatePermissionResult.LoginFailed: break;
  }
  return true;
}
```

Views that contain Custom Properties are generated for each Entity

Types can be loaded by key, or used when loading their parent entities. Custom Properties Dictionaries are available as a property of the parent entity. Custom Permissions can be checked against a user with a single-line API call.

Demonstration

- Create an open Content Page
  - Add a Property
  - Show in the Catalog
- Create a Content Part
  - Add a Property
  - Show in the Catalog

- Configure a Custom Type
- Configure a Custom Property
- Configure a Custom Permission
- Show all three in Configuration
- Reference all three in a web script

Lab 14

Logic Service Scripting

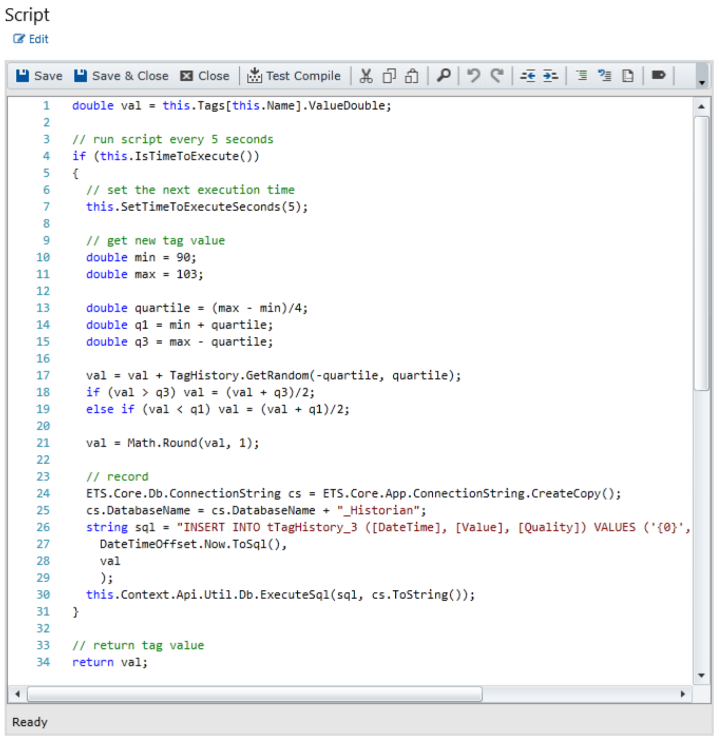
Training Objectives

Describe the Advanced Scripting opportunities available within the Logic Service execution.

Understand how Advanced Scripting can be used to create solution-specific business rules and extend the standard TrakSYS capabilities.

Advanced Script Tags

- Produce a value using an embedded .NET Script
  - **Simple**
    Single-Line expression written in VB.NET
  - **Advanced**
    Multi-Line function written in C#.NET
- Must return a value of the Data Type assigned to the Tag
- May reference other Tags in the Configuration
- this.Tags
  - Collection of objects for all Tags loaded and evaluated by the Logic Service
- this.Context
  - Includes methods for creating Tag History or Sample Sub-Group data, writing values to Virtual Tags, and retrieving Data Tables from the TrakSYS Database



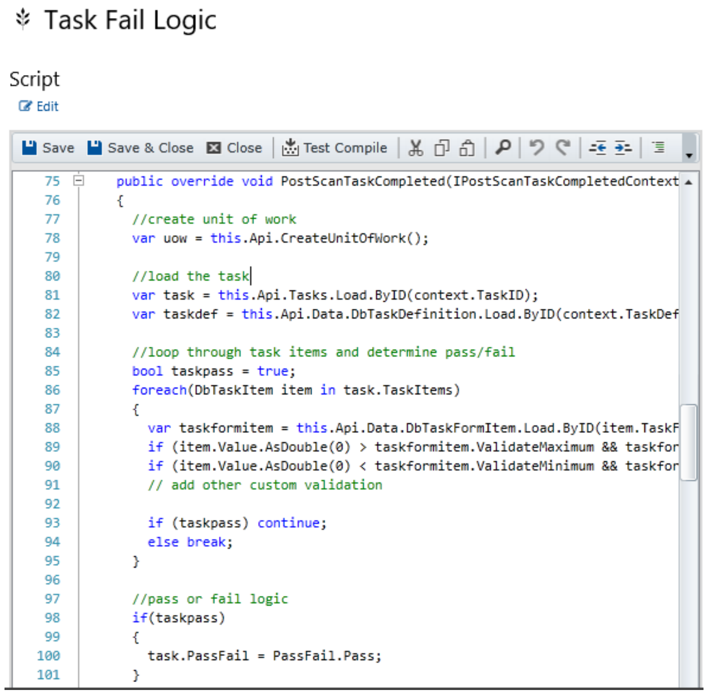
```
double val = this.Tags[this.Name].ValueDouble;

// run script every 5 seconds
if (this.IsTimeToExecute())
{
  // set the next execution time
  this.SetTimeToExecuteSeconds(5);

  // get new tag value
  double min = 90;
  double max = 103;

  double quartile = (max - min)/4;
  double q1 = min + quartile;
  double q3 = max - quartile;

  val = val + TagHistory.GetRandom(-quartile, quartile);
  if (val > q3) val = (val + q3)/2;
  else if (val < q1) val = (val + q1)/2;

  val = Math.Round(val, 1);

  // record
  ETS.Core.Db.ConnectionString cs = ETS.Core.App.ConnectionString.CreateCopy();
  cs.DatabaseName = cs.DatabaseName + "_Historian";
  string sql = "INSERT INTO tTagHistory_3 ([DateTime], [Value], [Quality]) VALUES ('{0}',
    DateTimeOffset.Now.ToSql(),
    val
    );
  this.Context.Api.Util.Db.ExecuteSql(sql, cs.ToString());
}

// return tag value
return val;
```

Advanced Script Tag Example

- Example of a Tag that will return 1 if a Line…
    - Job is NOT Running
    - Is within a scheduled Period
    - And is waiting on Tasks to be Completed

```
// set variables from current tag values
string jobName = this.Tags["P1.RUN.JOB"].ValueString;
string productCode = this.Tags["P1.RUN.PRODUCT_CODE"].ValueString;
bool isScheduled = !this.Tags["P1.EVENT.NOT_SCHEDULED"].ValueBoolean;

// determine if a job is running
bool isJobRunning = (jobName != "") && (productCode != "");

// determine if there are open tasks
int taskCount = CustomTaskUtil.GetOpenTaskCountForLine("P1");

// make final calculation
if ( !isJobRunning && isScheduled && (taskCount > 0) )
{ return true; }
else
{ return false; }
```

Logic Service Script Classes



Task Fail Logic

Script

```
public override void PostScanTaskCompleted(IPostScanTaskCompletedContext
{
  //create unit of work
  var uow = this.Api.CreateUnitOfWork();

  //load the task
  var task = this.Api.Tasks.Load.ByID(context.TaskID);
  var taskdef = this.Api.Data.DbTaskDefinition.Load.ByID(context.TaskDef

  //loop through task items and determine pass/fail
  bool taskpass = true;
  foreach(DbTaskItem item in task.TaskItems)
  {
    var taskformitem = this.Api.Data.DbTaskFormItem.Load.ByID(item.TaskF
    if (item.Value.AsDouble(0) > taskformitem.ValidateMaximum && taskfor
    if (item.Value.AsDouble(0) < taskformitem.ValidateMinimum && taskfor
    // add other custom validation

    if (taskpass) continue;
    else break;
  }

  //pass or fail logic
  if(taskpass)
  {
    task.PassFail = PassFail.Pass;
  }
```

Allows for development of C#.NET Classes

- Standard Scripts
  - C# classes for implementing utility Functions
  - Instanced and called from Script Tags / Script Classes
- Entity Scripts
  - Implement special functionality for select configuration entities (Systems, Event Definitions, Task Definitions, etc…)
  - Methods triggered by entity behavior (Logic Service Scan, Job Start/End, Event Start/End, etc…)
  - Classes must be explicitly assigned to Entities

Standard Script Class Example

- Example of a Task Utility Class that will lookup the number of Open (incomplete) Tasks that are currently in progress for a specified Line (specified by System Key).

```
/// *******************************************************************
public class CustomTaskUtil
{
  /// *******************************************************************
  public static int GetOpenTaskCountForLine(string systemKey)
  {
    // get api
    var api = ETS.Core.Api.ApiService.GetInstance();

    // get the system id
    string sqlSys =
"SELECT * FROM tSystem WHERE [Key] = {0}".FormatWith(systemKey.ToSql());
    var sys = api.Data.DbSystem.Load.WithSql(sqlSys)
        .ThrowIfLoadFailed("Key", systemKey);
    int systemID = sys.ID;

    // get open task count
    string sqlTask =
"SELECT COUNT(*) FROM viewTask WHERE IsComplete = 0 AND SystemID =
{0}".FormatWith(systemID.ToSql());
    int taskCount = api.Util.Db.ExecuteScalar<int>(sqlTask).Return;

    return taskCount;
  }
}
```

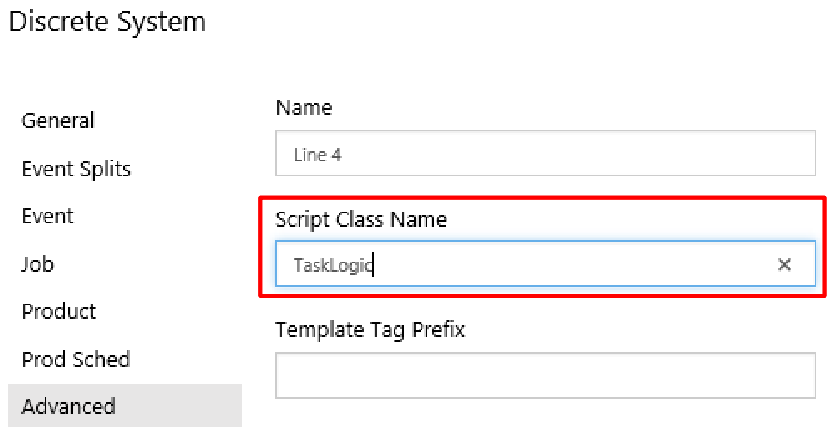
Entity Script Class Example

- Example of a System Script Class that runs whenever a Task is Late. The first two times the task is late, the Task's UserState is changed. The third time it is late, the Task automatically fails, and a notification will be sent.

```
/// ************************************************************************
public class TaskLogic : ETS.Core.Scripting.SystemScriptClassBase
{
  /// ******************************************************************
  public override void PostScanTaskLate(IPostScanTaskLateContext context)
  {
    var task = context.Api.Data.DbTask.Load.ByID(context.TaskID);
    switch(task.UserState)
    {
      case 0:
        task.CompleteByDateTime = task.CompleteByDateTime.AddMinutes(2);
        task.UserState = 1;
        break;
      case 1:
        task.CompleteByDateTime = task.CompleteByDateTime.AddMinutes(3);
        task.UserState = 2;
        break;
      case 2:
        task.PassFail = PassFail.Fail;
        context.Api.Notification.CreateNotification("TaskMissed", Supervisors",
        task.ShiftHistoryID, null);
        break;
      default:
        break;
    }
    context.Api.Data.DbTask.Save.UpdateExisting(task);
  }
}
```

Entity Script Class Assignment

- Script Classes are assigned using the .NET Class Name
- Class Name is added to the Script Class Name property in the Advanced Tab (for the target entity)
- Multiple Script Classes can be assigned to a single Entity (separate Class Names with a semi-colon ;)


Discrete System

Name: Line 4

Script Class Name: TaskLogic

Template Tag Prefix:

Data Management Modules

Training Objectives

Explore the Data Management Service and the configuration entities that enable its functionality.

Understand the mechanics of Modules and Module Steps, and how they can be used to orchestrate data aggregation and movement within TrakSYS.

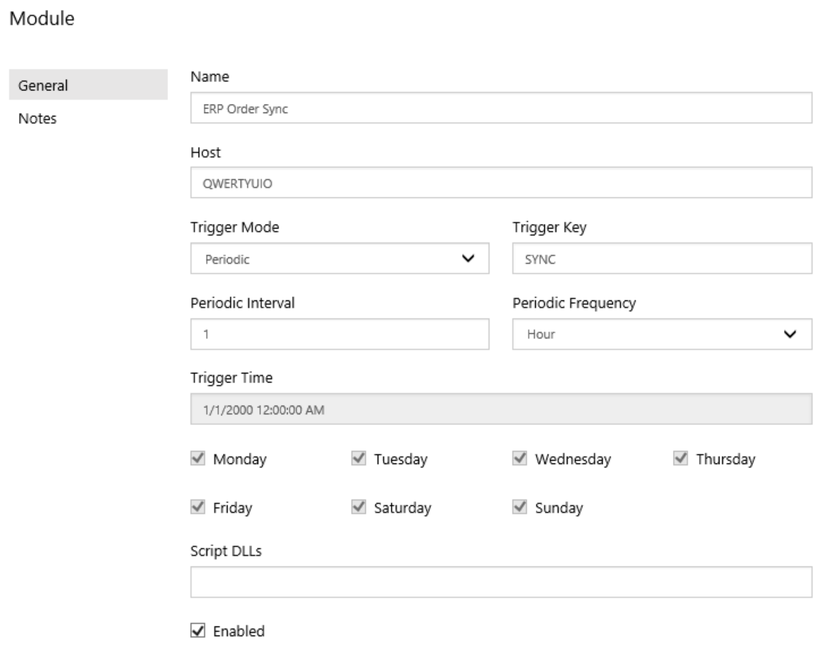
Data Management Service

- Independent multi-threaded service used for executing Non-Real Time Operations
- Facilitates processing large Data Aggregation
- Connects to external Business Systems for Import and Export of Configuration and Data
- Schedules periodic execution of scripted Modules

Supports Multiple and/or Distributed Instances

Data Management Modules

- A Module is a collection of business rules to be executed by the Data Management Service
- Modules contain one or more Module Steps that implement specific Functionality
- Module Steps are executed sequentially (no parallel processing within a Module)
- Modules can be scheduled to execute periodically or triggered externally by API Calls

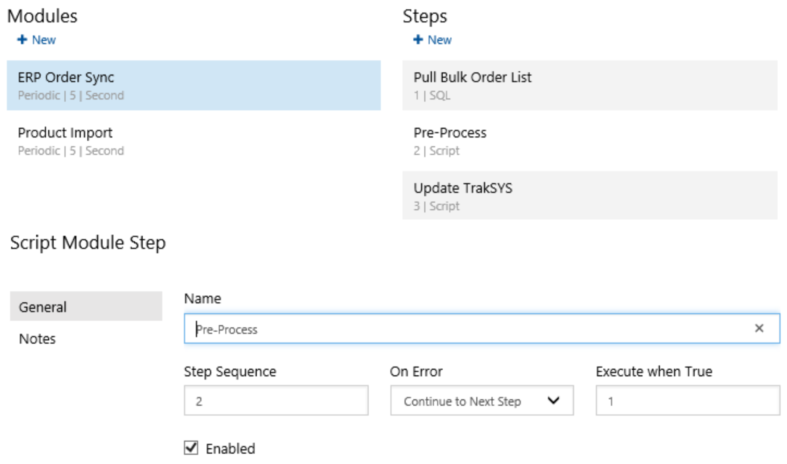

Module

Name	ERP Order Sync
Host	QWERTYUIO
Trigger Mode	Periodic
Trigger Key	SYNC
Periodic Interval	1
Periodic Frequency	Hour
Trigger Time	1/1/2000 12:00:00 AM
Script DLLs	

☑ Monday ☑ Tuesday ☑ Wednesday ☑ Thursday
☑ Friday ☑ Saturday ☑ Sunday

☑ Enabled

Module Steps



Modules

+ New

ERP Order Sync
Periodic | 5 | Second

Product Import
Periodic | 5 | Second

Steps

+ New

Pull Bulk Order List
1 | SQL

Pre-Process
2 | Script

Update TrakSYS
3 | Script

Script Module Step

General

Notes

Name: Pre-Process

Step Sequence: 2

On Error: Continue to Next Step

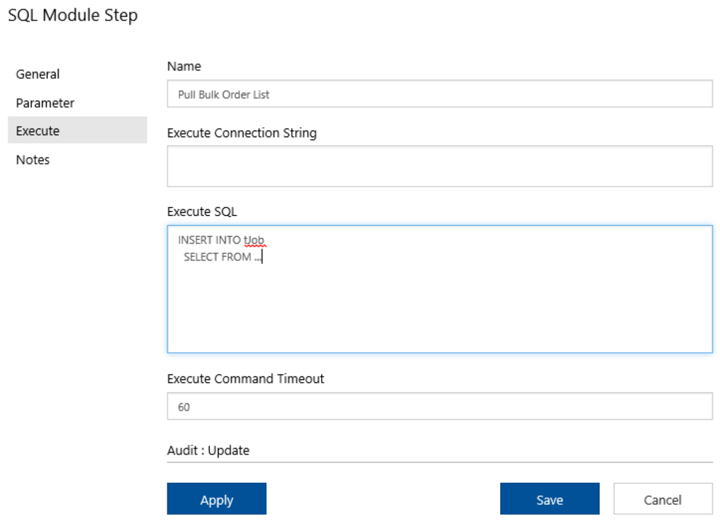
Execute when True: 1

☑ Enabled

- A Module Step is a distinct operation to be run from within a parent Module
- Module Steps contain configured options for Conditional Execution and responses to unhandled Errors/Exceptions
- Module Step Types
  - SQL Module Step
    Database Read/Writes
  - Script Module Step
    API Calls, File Read/Writes, Web Services
  - Metric Calculator Module Step
    * Legacy

SQL Module Steps

- Step functionality is implemented using Structured Query Language (SQL)
- The Execute SQL setting contains all T-SQL to be executed against an OLEDB-compliant database Connection
  - Built-in handling of database Connections, Timeouts, and Errors
  - Connection String may be left blank to connect to the TrakSYS Database
- The Parameter SQL setting creates a record set over which to run the Execute SQL
  - Allows for multiple executions of the main SQL query over a list of identifiers or parameters
  - Can execute insert/update/delete statements against a different database than the Parameter SQL



Name: Pull Bulk Order List

Execute Connection String:

Execute SQL:

```
INSERT INTO tJob
  SELECT FROM ...
```

Execute Command Timeout: 60

Audit : Update

Example: Parameter and Execute SQL Queries

Parameter Query

```
SELECT
  UpcCode,
  ValidatedRate
FROM
  [ERP].[dbo].[ProductData]
```

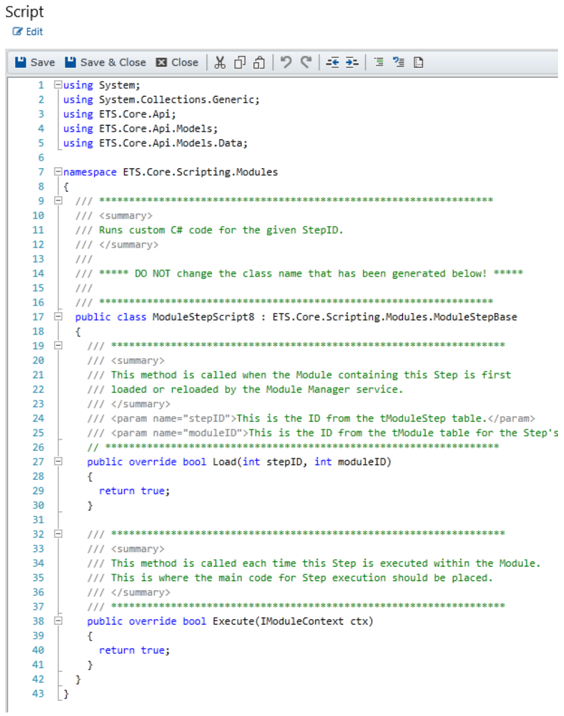
Execute Query

```
UPDATE tProduct
SET
  Attribute01 = '{param.sql|ValidatedRate}'
WHERE
  (ProductCode = '{param.sql|UpcCode}')
```

UpcCode	ValidatedRate
FRP-1000167	210
FRZ-1000283	240
FRX-1000552	225
FRX-1000629	240

Executed for each record returned by the Parameter SQL, where {param} values are updated with the values from each record

Script Module Steps

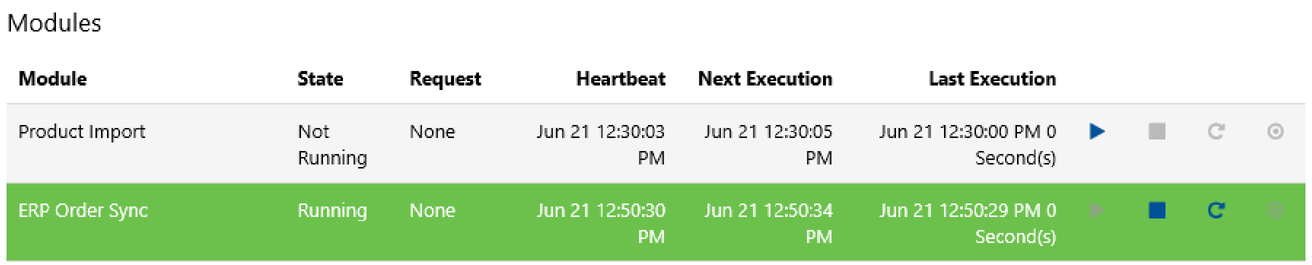

Script
Edit

```
using System;
using System.Collections.Generic;
using ETS.Core.Api;
using ETS.Core.Api.Models;
using ETS.Core.Api.Models.Data;

namespace ETS.Core.Scripting.Modules
{
  /// ******************************************************************
  /// <summary>
  /// Runs custom C# code for the given StepID.
  /// </summary>
  ///
  /// ***** DO NOT change the class name that has been generated below! *****
  ///
  /// ******************************************************************
  public class ModuleStepScript8 : ETS.Core.Scripting.Modules.ModuleStepBase
  {
    /// ******************************************************************
    /// <summary>
    /// This method is called when the Module containing this Step is first
    /// loaded or reloaded by the Module Manager service.
    /// </summary>
    /// <param name="stepID">This is the ID from the tModuleStep table.</param>
    /// <param name="moduleID">This is the ID from the tModule table for the Step's
    // ******************************************************************
    public override bool Load(int stepID, int moduleID)
    {
      return true;
    }

    /// ******************************************************************
    /// <summary>
    /// This method is called each time this Step is executed within the Module.
    /// This is where the main code for Step execution should be placed.
    /// </summary>
    /// ******************************************************************
    public override bool Execute(IModuleContext ctx)
    {
      return true;
    }
  }
}
```

- Step functionality is implemented using TrakSYS Advanced Scripting (Microsoft C#.NET)
  - The Load() method is run once when the Module is first loaded. Anything instantiated at the class level and populated here is maintained across Module execution calls
  - The Execute() method is run each time the Modules is executed when the Script Step is reached.
- The TrakSYS API is available and passed into the Execute method via the ctx Argument

Module Control

- The Data Management Service should always be Running (Windows Services Automatic)
- Modules may be Started, Stopped or Restarted from the Services Administration | Services Hub
- Difference between Running and Executing
- If the Service Restarts, Modules resume their previous running State



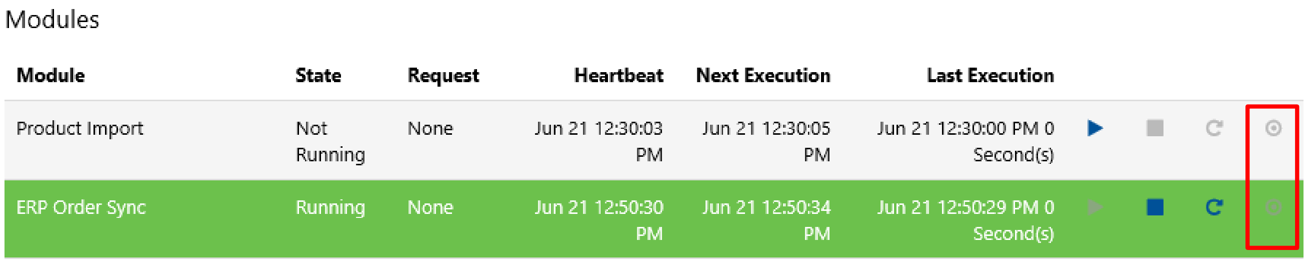
Modules

Module	State	Request	Heartbeat	Next Execution	Last Execution
Product Import	Not Running	None	Jun 21 12:30:03 PM	Jun 21 12:30:05 PM	Jun 21 12:30:00 PM 0 Second(s)
ERP Order Sync	Running	None	Jun 21 12:50:30 PM	Jun 21 12:50:34 PM	Jun 21 12:50:29 PM 0 Second(s)

Module Triggers

- Modules will trigger periodically if configured
- Modules may be Executed through the Script and Module Control (one-time run)
  - Must be Running
  - Must have a Trigger Key

```
// create module trigger
var mt = this.Ets.Api.Data
  .DbModuleTrigger.Create.FromParentNone();

// identify which module and when
mt.TriggerDateTime = this.Ets.SiteNow;
mt.TriggerKey = "ProductImport";

// save trigger
this.Ets.Api.Data.DbModuleTrigger.Save
  .InsertAsNew(mt).ThrowIfFailed();
```



Modules

Module	State	Request	Heartbeat	Next Execution	Last Execution				
Product Import	Not Running	None	Jun 21 12:30:03 PM	Jun 21 12:30:05 PM	Jun 21 12:30:00 PM 0 Second(s)	▶	■	↻	⊙
ERP Order Sync	Running	None	Jun 21 12:50:30 PM	Jun 21 12:50:34 PM	Jun 21 12:50:29 PM 0 Second(s)	▶	■	↻	⊙

Demonstration

- Configure a simple Utility Script Class
- Reference the Utility Script Class from an Advanced Script Tag
- Configure a System Entity Script Class
- Show an API Model and Service example in the Event Start Method
- Associate the Entity Script Class with a System
- Configure a Data Management Module
- Configure a SQL Module Step
- Configure a Script Module Step
- Show the Data Management Service in Windows Services
- Show the Module Control Interface
    - Start
    - Execute

Lab 15

Sites

Training Objectives

Understand the new TrakSYS Site capabilities and when multi-site implementations are appropriate.

Explain the TrakSYS Site licensing model and implications.

Why Sites? Physical Locations

Site 1	Site 2	Site 3
• Equipment	• Equipment	• Equipment
• Rules	• Rules	• Rules
• Materials	• Materials	• Materials
• User Interfaces	• User Interfaces	• User Interfaces
• Users / Roles	• Users / Roles	• Users / Roles

Installation

Why Sites? Independent Operations

Production Area 1

- Equipment
- Rules
- Materials
- User Interfaces
- Users / Roles

Production Area 2

- Equipment
- Rules
- Materials
- User Interfaces
- Users / Roles

Production Area 3

- Equipment
- Rules
- Materials
- User Interfaces
- Users / Roles

Installation

Root Site

Site 1

- Equipment
- Rules
- Materials
- User Interfaces
- Users / Roles

Site 2

- Equipment
- Rules
- Materials
- User Interfaces
- Users / Roles

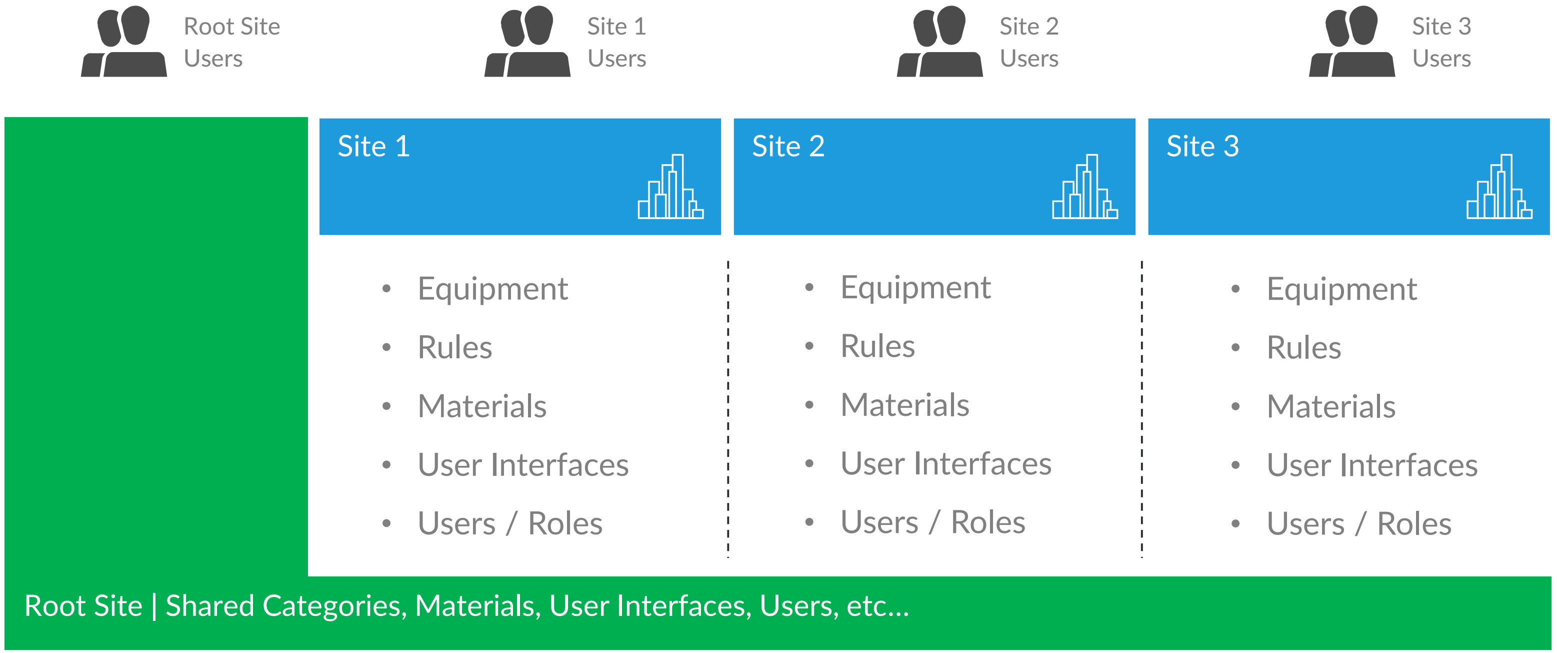
Site 3

- Equipment
- Rules
- Materials
- User Interfaces
- Users / Roles

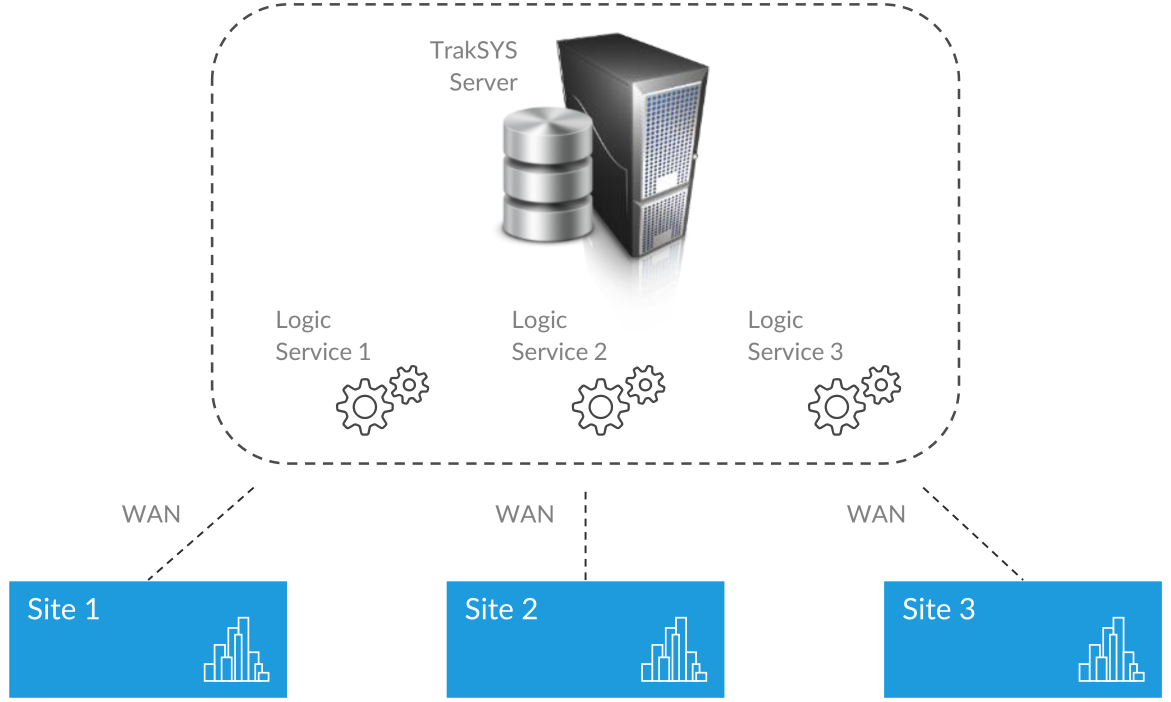
Root Site | Shared Categories, Materials, User Interfaces, Users, etc…

Installation

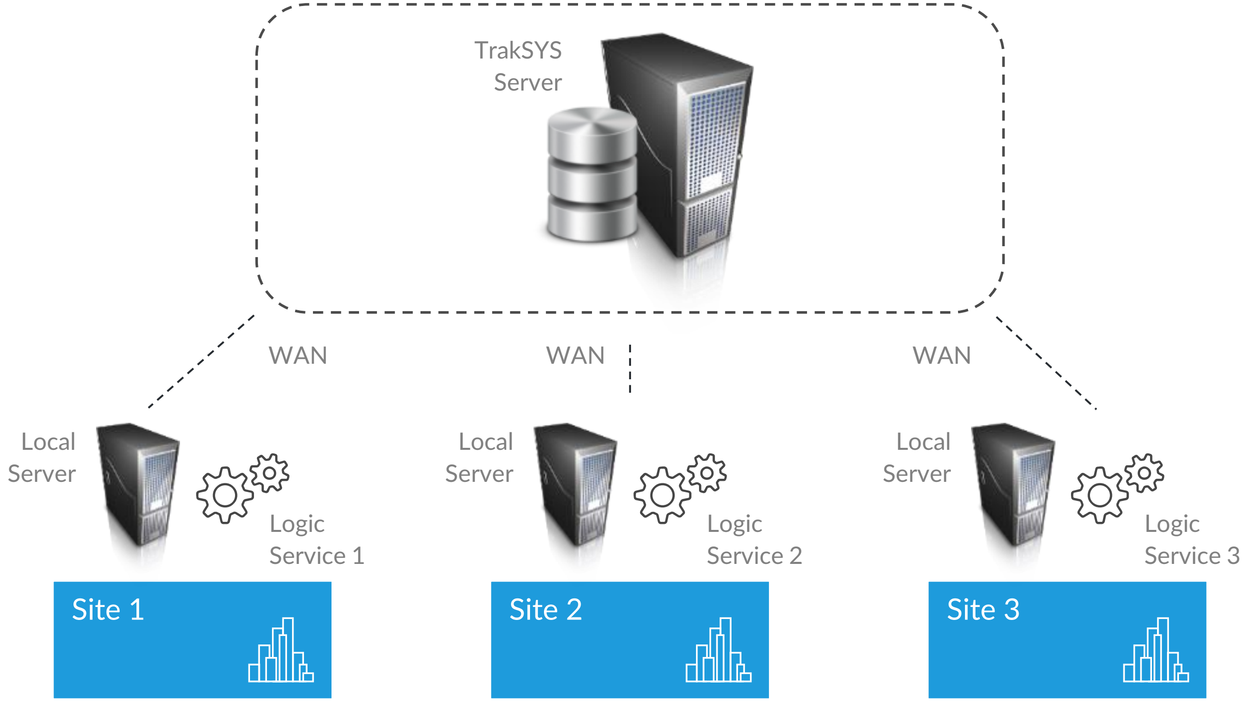
Site Access Control



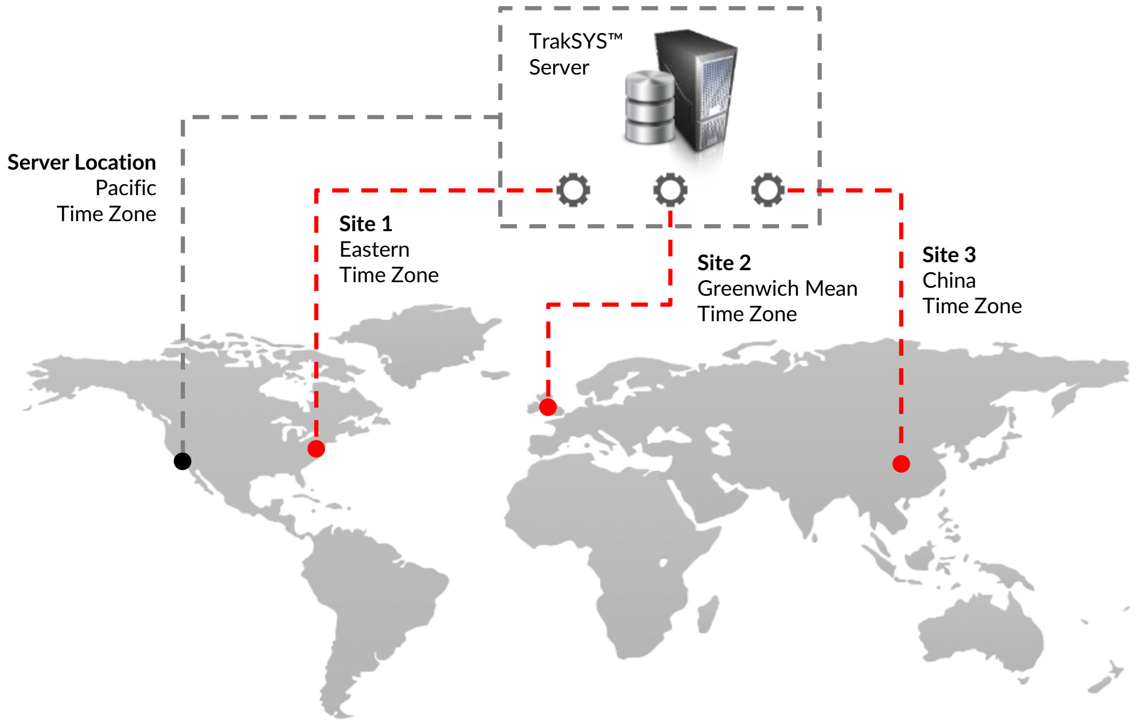
Architecture 1



Architecture 2

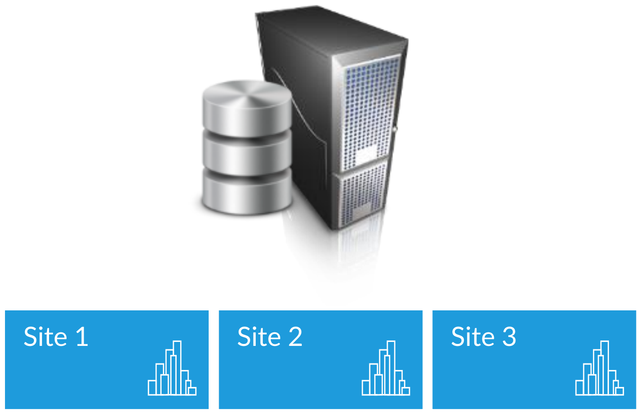


Time Zones

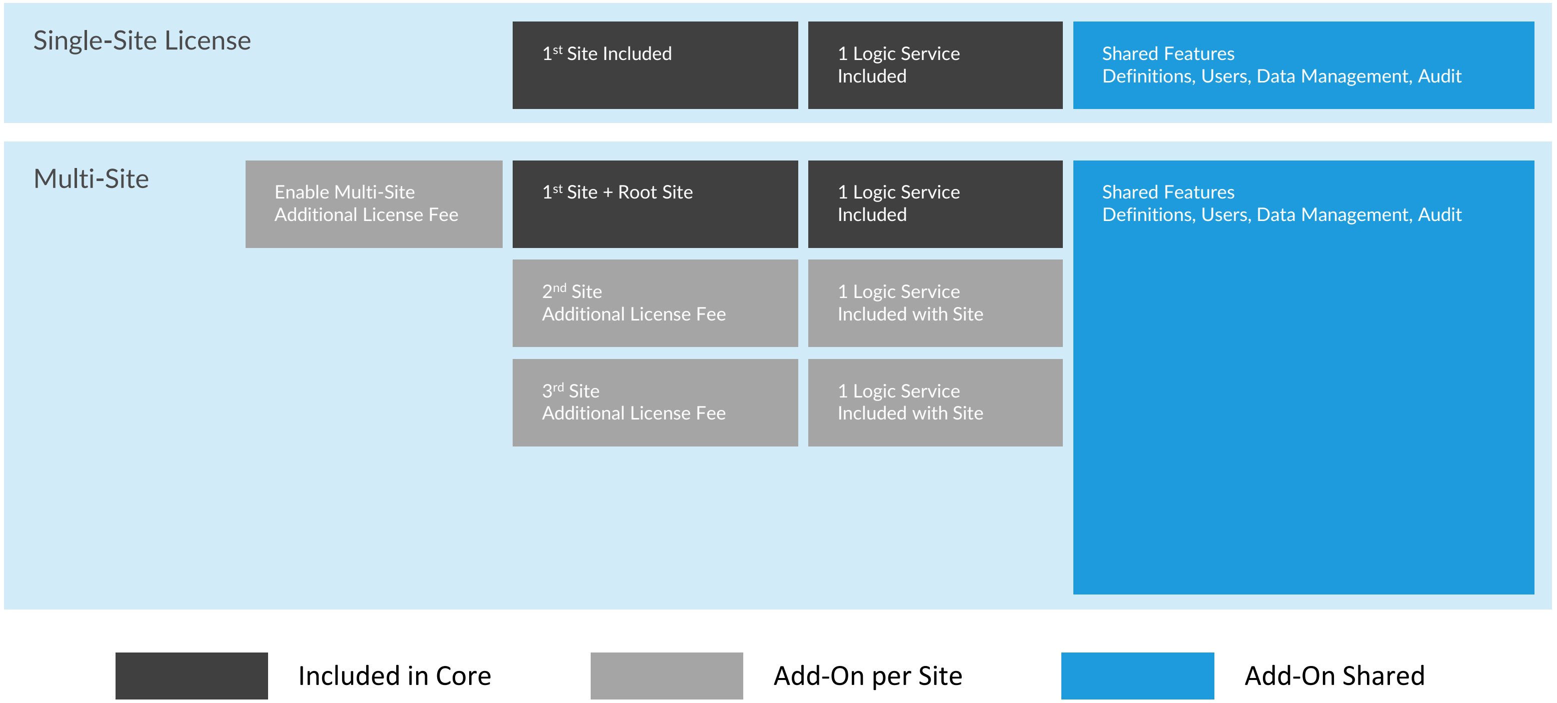

TrakSYS™ Server
Server Location Pacific Time Zone
Site 1 Eastern Time Zone
Site 2 Greenwich Mean Time Zone
Site 3 China Time Zone

Considerations

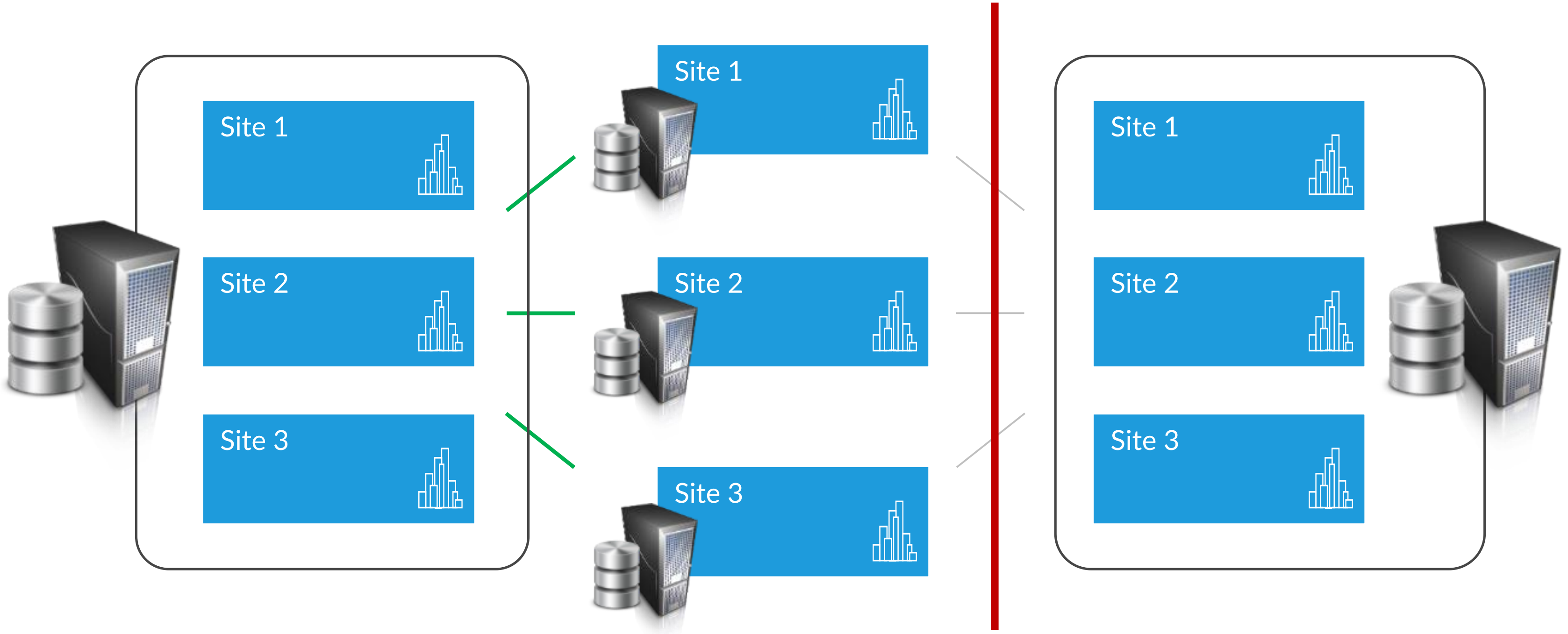
- Server Load
  While data collection Services can be distributed, the database and web server are shared.
- Remote Communication
  Network visibility/ports must be open and accessible between local automation devices and client browsers.
- Speed and Availability
  Network speeds and connection availability must be adequate for the application requirements.
- Governance
  A clear plan should be in place for who manages system upgrades, support renewals, and other global decisions.
- Upgrades and Maintenance
  Any system maintenance or upgrades will affect all Sites housed within the shared implementation.



Multi-Site Licensing Summary


Single-Site License
1st Site Included
1 Logic Service Included
Shared Features Definitions, Users, Data Management, Audit
Multi-Site
Enable Multi-Site Additional License Fee
1st Site + Root Site
1 Logic Service Included
Shared Features Definitions, Users, Data Management, Audit
2nd Site Additional License Fee
1 Logic Service Included with Site
3rd Site Additional License Fee
1 Logic Service Included with Site
Included in Core
Add-On per Site
Add-On Shared

Splitting and Combining



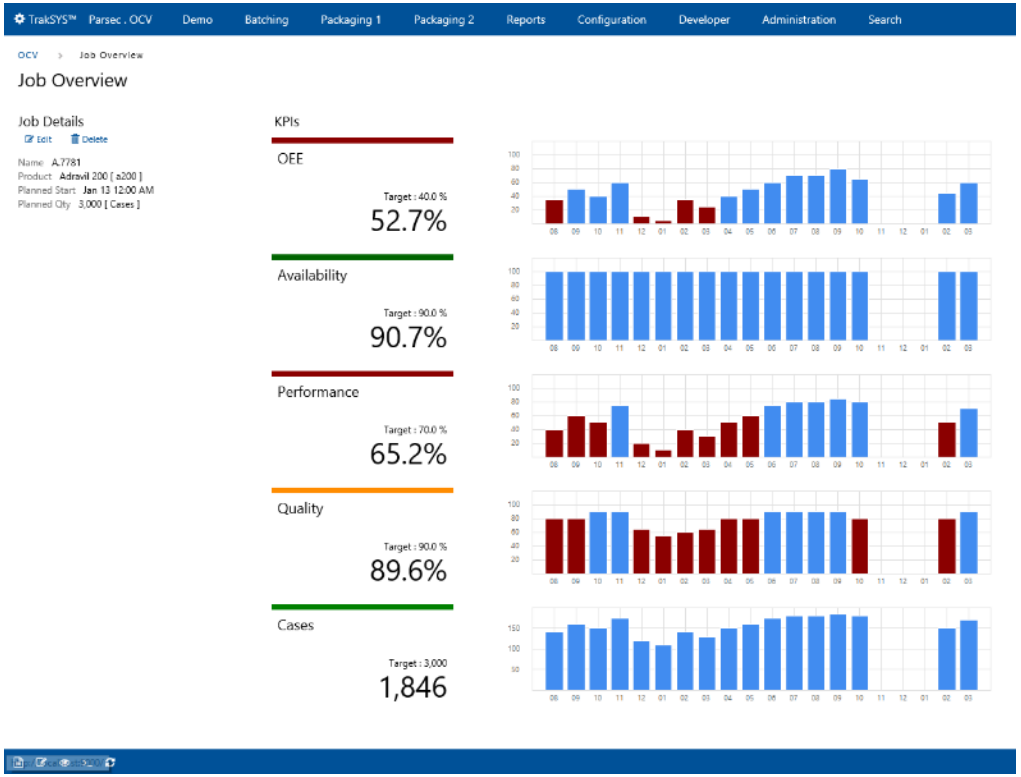
Translations

Training Objectives

Describe the TrakSYS capabilities for multi-language support for both built-in text, as well as the extensibility options for adding and integrating solution-specific Translations.

Language Translation

- Supports displaying text in the language appropriate for the User
- Supports displaying different languages from the same application Install
- Supports both built-in and solution-specific text Translations



Job Overview

Job Details

Edit Delete

Name A.7781
Product Adravil 200 [a200]
Planned Start Jan 13 12:00 AM
Planned Qty 3,000 [Cases]

KPIs

OEE
Target : 40.0 %
52.7%

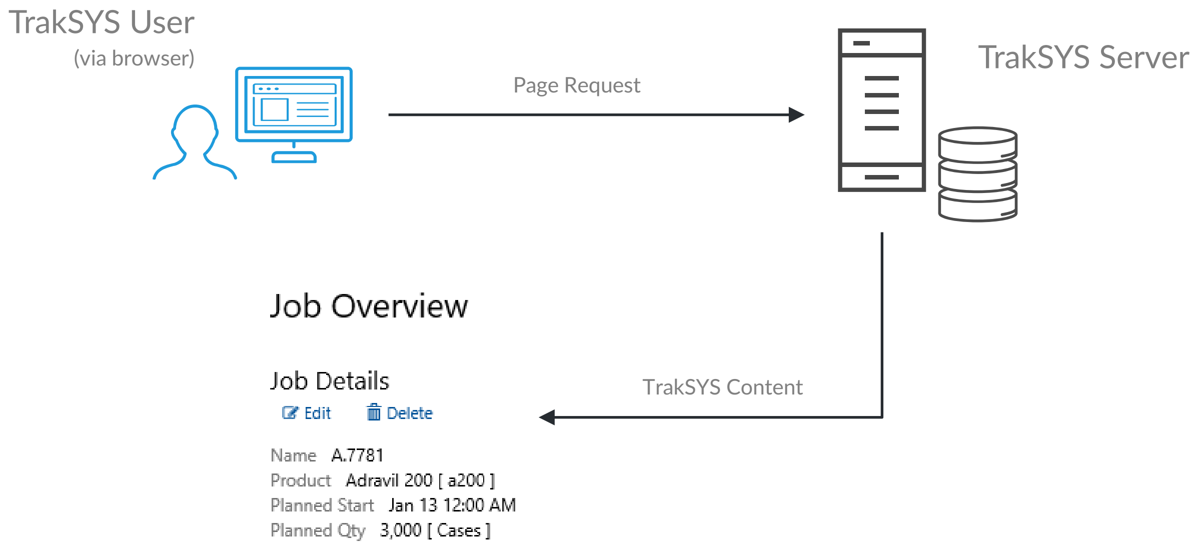
Availability
Target : 90.0 %
90.7%

Performance
Target : 70.0 %
65.2%

Quality
Target : 90.0 %
89.6%

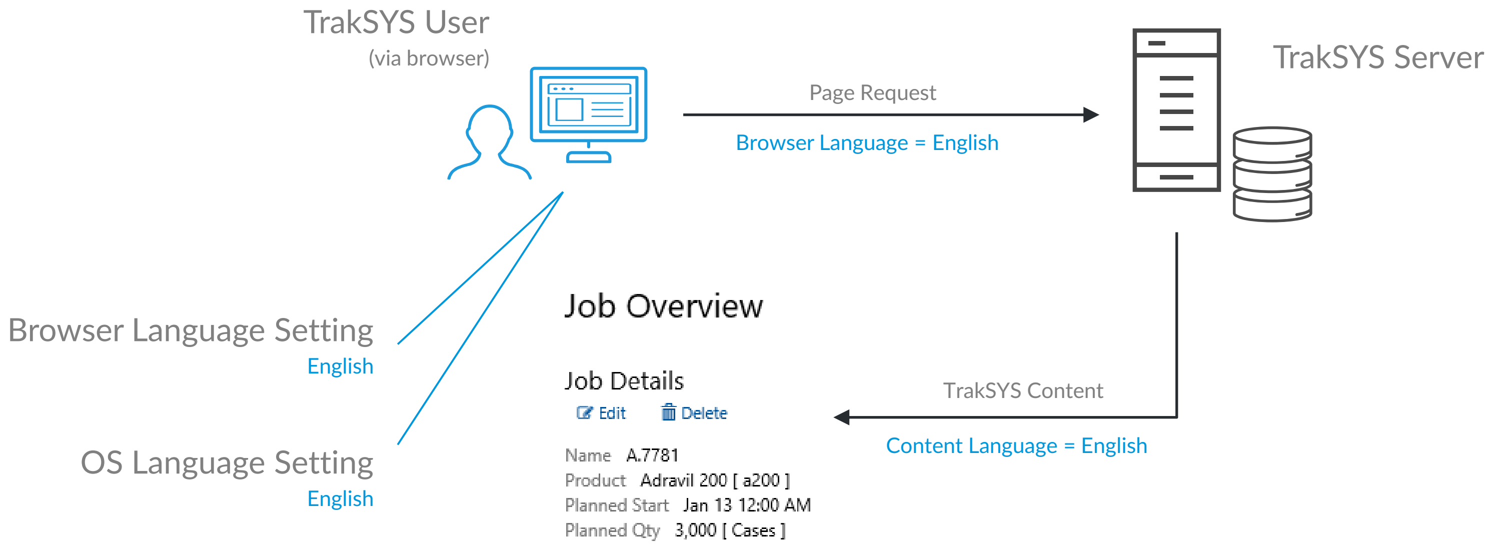
Cases
Target : 3,000
1,846

Basic Page Request



Name A.7781
Product Adravil 200 [a200]
Planned Start Jan 13 12:00 AM
Planned Qty 3,000 [Cases]

English Page Request



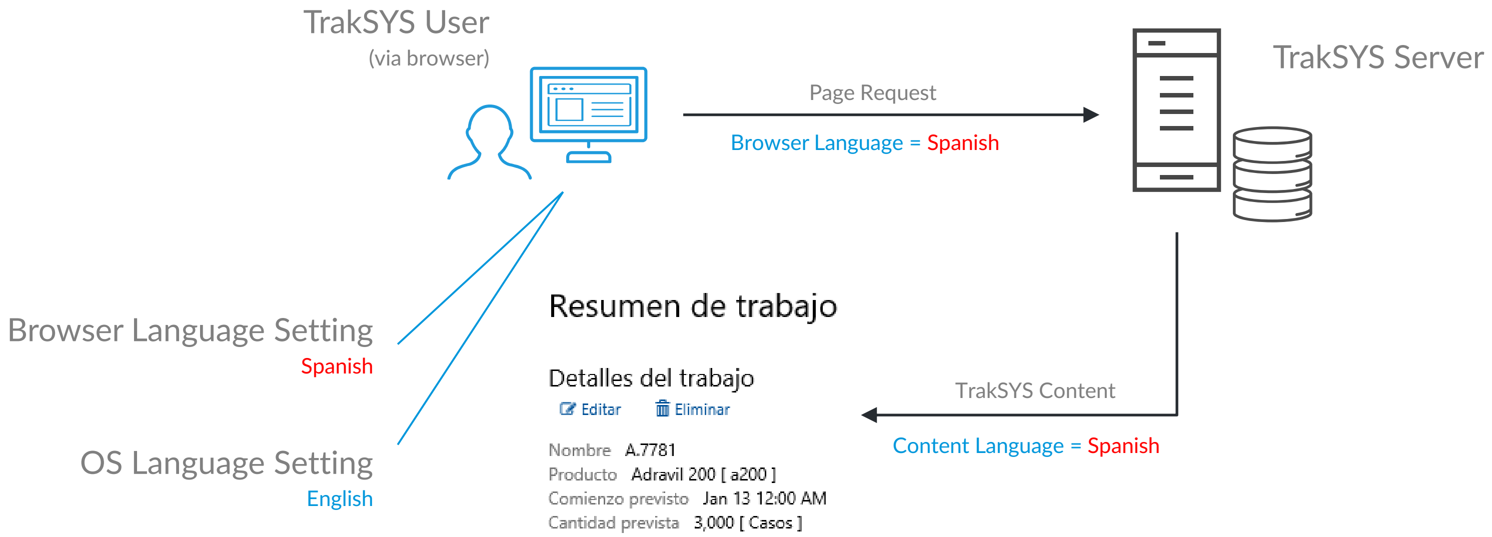
Job Overview

Job Details

Edit Delete

Name A.7781
Product Adravil 200 [a200]
Planned Start Jan 13 12:00 AM
Planned Qty 3,000 [Cases]

Spanish Page Request



Resumen de trabajo

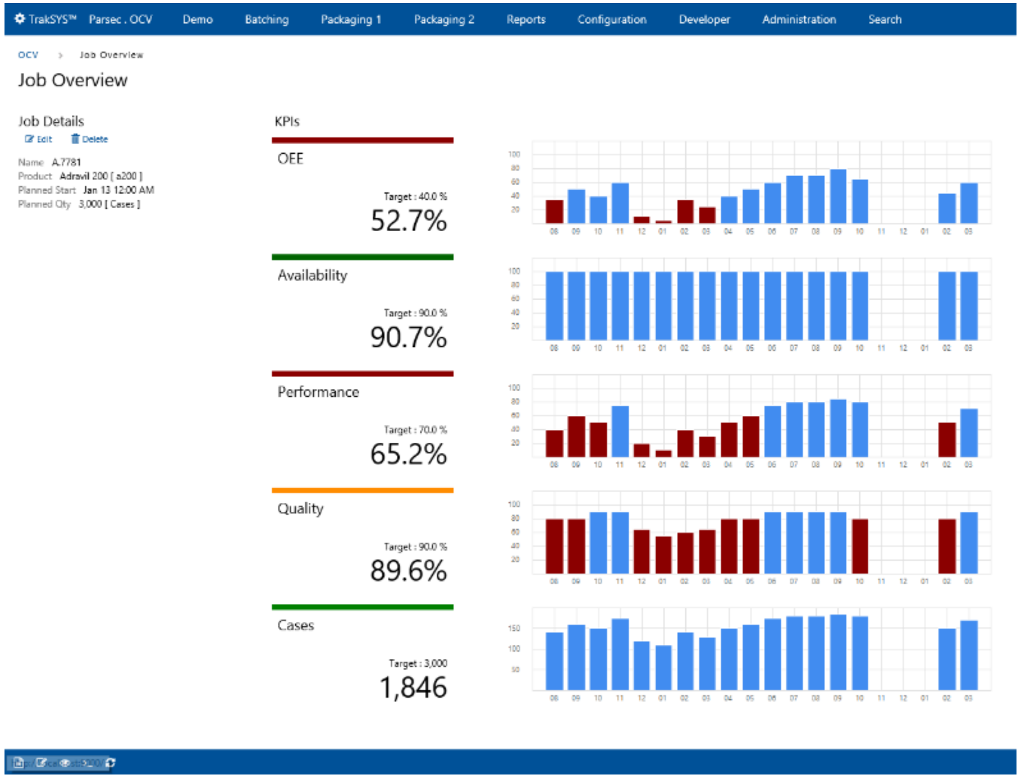
Detalles del trabajo

Nombre	A.7781
Producto	Adravil 200 [a200]
Comienzo previsto	Jan 13 12:00 AM
Cantidad prevista	3,000 [Casos]

Language Text (strings) in TrakSYS

- **Built-In**
  Menus
  Captions
  Standard Parts
  Standard Pages

- **Solution-Specific**
  Operations-Specific Terms
  Site-Specific Parts and Pages
  Anything YOU Create



Job Overview

Job Details

Name A.7781
Product Adravil 200 [a200]
Planned Start Jan 13 12:00 AM
Planned Qty 3,000 [Cases]

KPIs

OEE
Target : 40.0 %
52.7%

Availability
Target : 90.0 %
90.7%

Performance
Target : 70.0 %
65.2%

Quality
Target : 90.0 %
89.6%

Cases
Target : 3,000
1,846

Translation Entities

Locale

- A string identifier that references a specific Language
- Known as Culture Names in the Windows Environment
- Examples
  - en = US English
  - es = Spanish

Resource Group

- A grouping, context and organization mechanism for Resource Items and Translations
- Has a Key which is the prefix for the unique identifier that will be used to reference a Translation

Resource Item

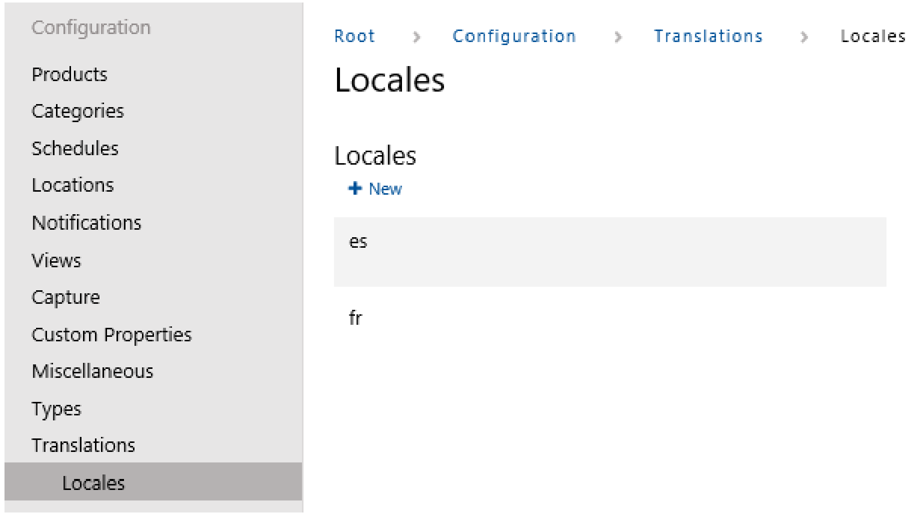
- A text string for translation and display within the TrakSYS User Interface
- Has a Key which is the suffix for the unique identifier that will be used to reference a Translation

Translation

- Each Resource Item will contain multiple Translations
- A Translation is a alternate language string for the Resource Item – for a specific Locale

Locales

- A string identifier that references a specific Language
- Known as Culture Names in the Windows Environment http://www.csharp-examples.net/culture-names/
- Examples
  - en = US English
  - es = Spanish
  - fr = French
  - de = German

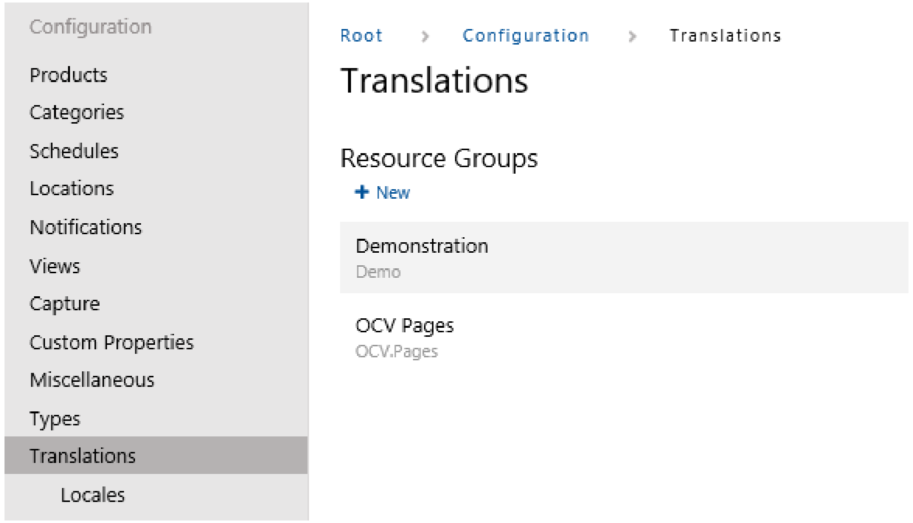


es

fr

Resource Groups

- A grouping and organization mechanism for Resource Items and Translations
- Each Resource Group is assigned a Key which is the prefix for the unique identifier that will be used to reference a Translation
- The Resource Group Key must be Unique

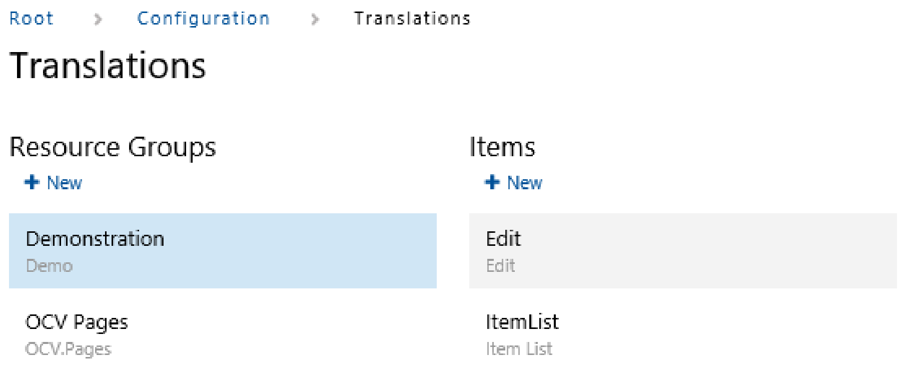


Demonstration
Demo

OCV Pages
OCV.Pages

Resource Items

- A text string that will be available for translation and display within the TrakSYS User Interface
- Each Resource Item is assigned a Key which is the suffix for the unique identifier that will be used to reference a Translation
- The Resource Item Key must be unique within the parent Group
- A Default Value must be assigned. This is essentially the English translation for the Item

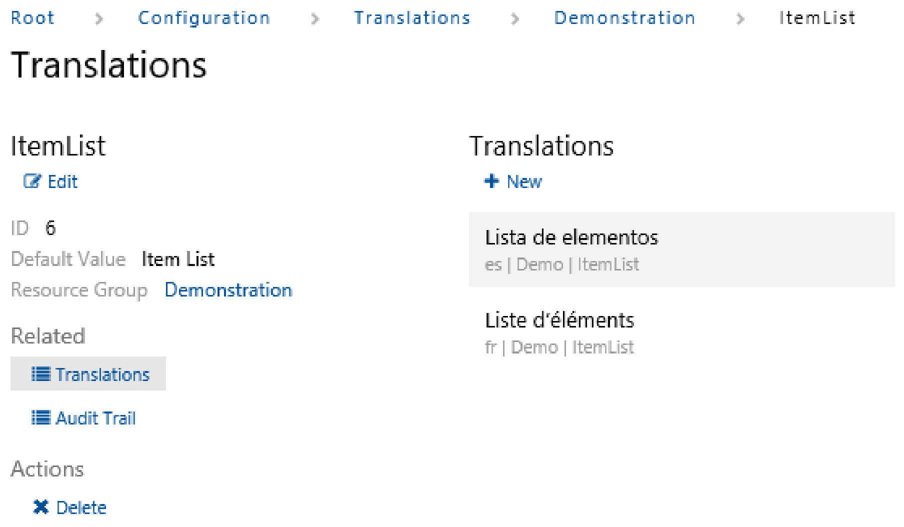

Root › Configuration › Translations

Translations

Resource Groups	Items
+ New	+ New
Demonstration Demo	Edit Edit
OCV Pages OCV.Pages	ItemList Item List

Translations

- Each Resource Item should contain multiple Translations
- A Translation is an alternate language string for the Resource Item – for a specific Locale
- If a Translation does not exist for a specific Resource Item – Locale combination, the Default Value (English) is Assumed


Root › Configuration › Translations › Demonstration › ItemList

Translations

ItemList

Edit

ID 6
Default Value Item List
Resource Group Demonstration

Related

Translations

Audit Trail

Actions

Delete

Translations

New

Lista de elementos
es | Demo | ItemList

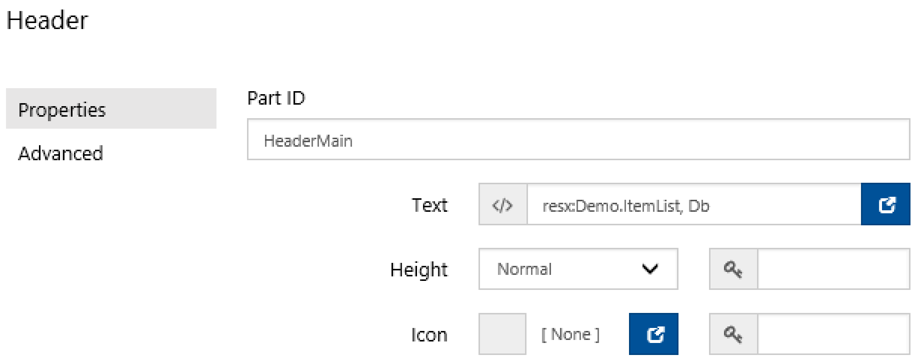
Liste d'éléments
fr | Demo | ItemList

Referencing Translations

- Translations can be referenced with a special TrakSYS Expression
- The Expression syntax uses a combination of the Resource Group and Item Keys
- Note there is NO curly braces!
- The Locale is determined by information passed from the client Browser (not the client Operating System language)

resx : ResourceGroupKey . ResourceItemKey , Db

resx : Demo . ItemList , Db


Header

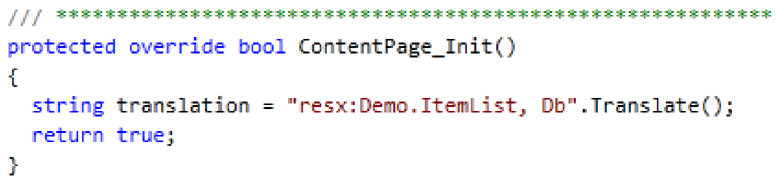
Properties
Advanced

Part ID: HeaderMain

Text: resx:Demo.ItemList, Db

Height: Normal

Icon: [None]

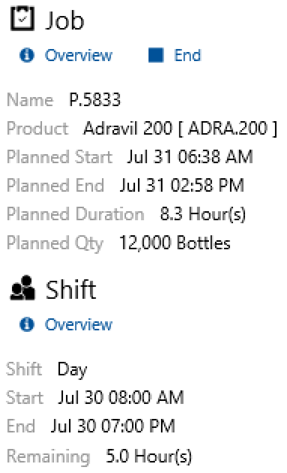


```
/// *************************************************************
protected override bool ContentPage_Init()
{
  string translation = "resx:Demo.ItemList, Db".Translate();
  return true;
}
```

Translations Example

		Locale		
Resource Group	Resource Item	Default (English)	Parsec (English)	es (Spanish)
Business Report	Job	Job	Production Order	Orden
	Product	Product	SKU	Producto
	User	User	Personnel	Personal
Factory Screens	Job	Job	Production Run	Trabajo
	Product	Product	Part	Parte
	User	User	User	Usario

Default User



Job

Overview   End

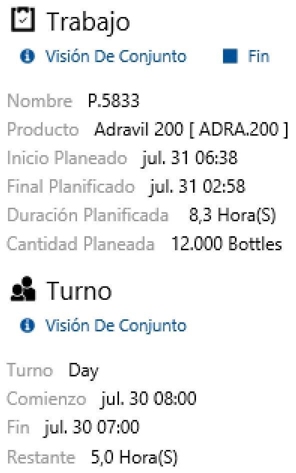
Name P.5833
Product Adravil 200 [ADRA.200]
Planned Start Jul 31 06:38 AM
Planned End Jul 31 02:58 PM
Planned Duration 8.3 Hour(s)
Planned Qty 12,000 Bottles

Shift

Overview

Shift Day
Start Jul 30 08:00 AM
End Jul 30 07:00 PM
Remaining 5.0 Hour(s)

Spanish User



Trabajo

Visión De Conjunto   Fin

Nombre P.5833
Producto Adravil 200 [ADRA.200]
Inicio Planeado jul. 31 06:38
Final Planificado jul. 31 02:58
Duración Planificada 8,3 Hora(S)
Cantidad Planeada 12.000 Bottles

Turno

Visión De Conjunto

Turno Day
Comienzo jul. 30 08:00
Fin jul. 30 07:00
Restante 5,0 Hora(S)

Audit

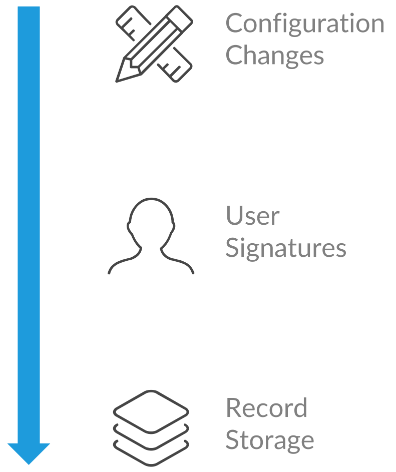
Training Objectives

Understand the basic concept of 21 CFR Part 11, the main regulatory requirements that TrakSYS can help fulfill with its Auditing features.

Explore the different configuration options, data collection capabilities, and reporting available in the TrakSYS Auditing features.

Audit and Regulatory Management

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

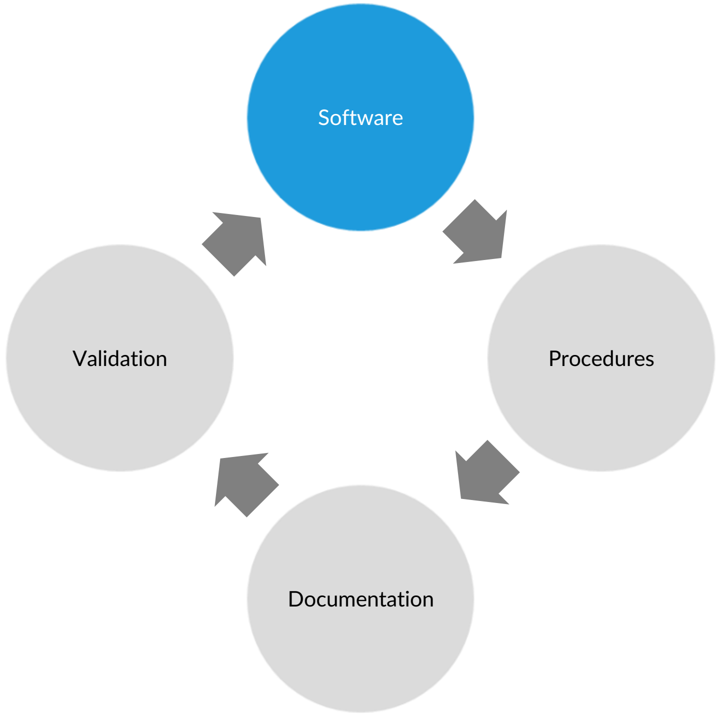


21 CFR Part 11

- **Objective**
  - To allow the industry to use electronic records and signatures alternatively to paper records and hand-written signatures.
- **Scope**
  - FDA Regulated Environments
  - When using computers to create, modify, maintain, archive, retrieve or transmit data or records.
  - Records required by predicate rules (GLP, GCP, GMP) with high impact of patient safety.
  - Applies to new and existing systems.

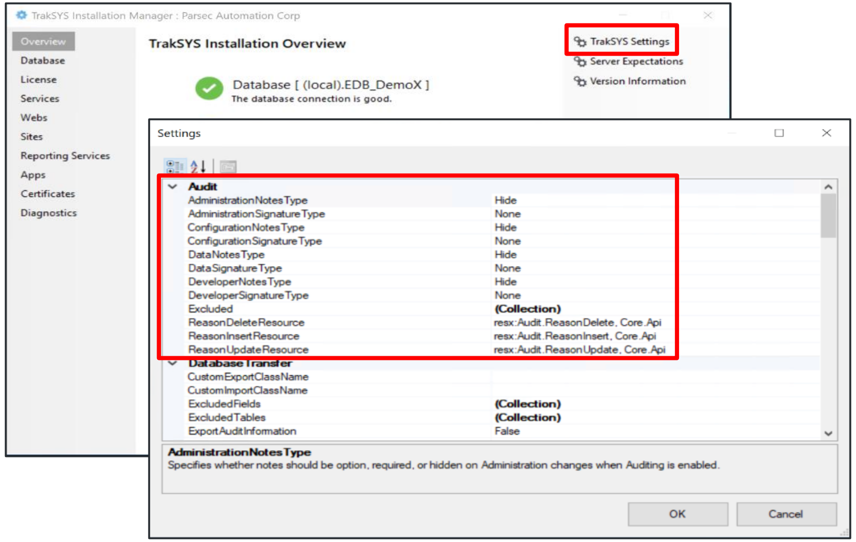
21 CFR Part 11 and TrakSYS

- Validation
- Accurate and Complete Copies
- Protection and Retrieval of Records
- Limited Access to Authorized Users
- Electronic Audit Trail
- Operational System Checks
- Authority Checks
- Device Checks
- People Qualification
- Individual Accountability
- Controls over System Documentation
- Digital Signatures for Open Systems
- Requirements for Signed Electronic Records
- Linking Signatures to e-Records



Audit Setup

- The Audit settings are controlled through the Installation Manager by navigating to the TrakSYS Settings from the Overview tab
- Changes to settings are saved to the Database, but do not immediately take effect
- Access to the Server is required to access these settings


TrakSYS Installation Manager : Parsec Automation Corp

TrakSYS Installation Overview

Database [(local).EDB_DemoX]
The database connection is good.

Settings

Audit	
AdministrationNotesType	Hide
AdministrationSignatureType	None
ConfigurationNotesType	Hide
ConfigurationSignatureType	None
DataNotesType	Hide
DataSignatureType	None
DeveloperNotesType	Hide
DeveloperSignatureType	None
Excluded	**(Collection)**
ReasonDeleteResource	resx:Audit.ReasonDelete, Core.Api
ReasonInsertResource	resx:Audit.ReasonInsert, Core.Api
ReasonUpdateResource	resx:Audit.ReasonUpdate, Core.Api
Database Transfer	
CustomExportClassName	
CustomImportClassName	
ExcludedFields	**(Collection)**
ExcludedTables	**(Collection)**
ExportAuditInformation	False

AdministrationNotesType
Specifies whether notes should be option, required, or hidden on Administration changes when Auditing is enabled.

Audit Settings

- Control by Section
  - Administration, Configuration, Developer, Data
- Signature Type
  - None

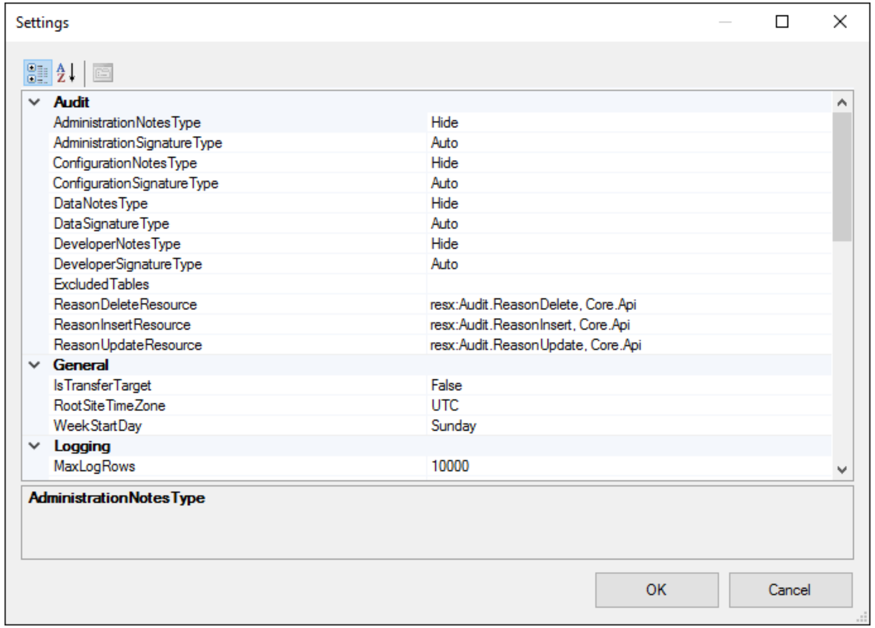
    No signature prompts, no Audit recorded.
  - Auto

    Reason prompts only. User is auto recorded.
  - Name Only

    Prompt for name only.
  - Single

    Prompt for one login and password.
  - Dual

    Prompt for 2 sets of logins and passwords.

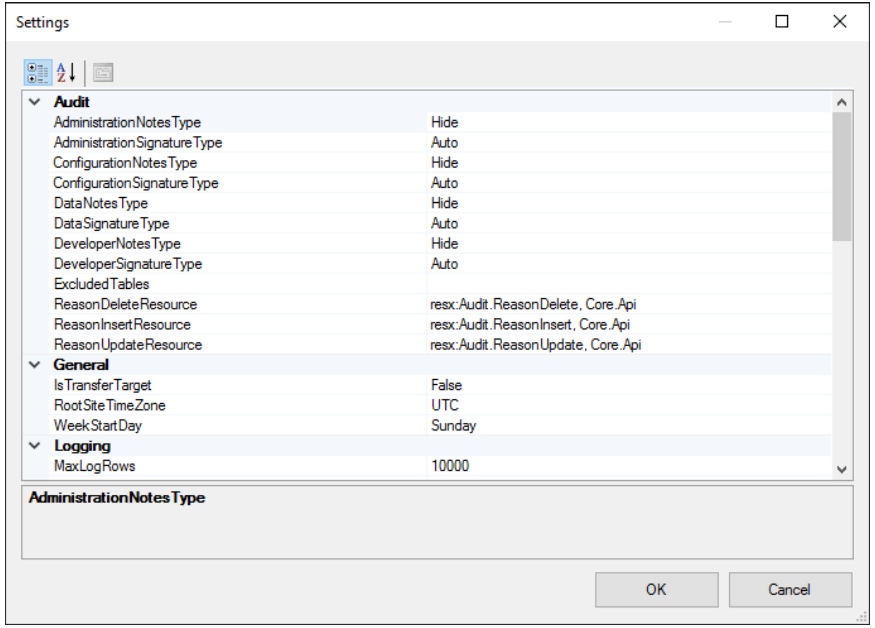


Audit	
AdministrationNotesType	Hide
AdministrationSignatureType	Auto
ConfigurationNotesType	Hide
ConfigurationSignatureType	Auto
DataNotesType	Hide
DataSignatureType	Auto
DeveloperNotesType	Hide
DeveloperSignatureType	Auto
ExcludedTables	
ReasonDeleteResource	resx:Audit.ReasonDelete, Core.Api
ReasonInsertResource	resx:Audit.ReasonInsert, Core.Api
ReasonUpdateResource	resx:Audit.ReasonUpdate, Core.Api
General	
IsTransferTarget	False
RootSiteTimeZone	UTC
WeekStartDay	Sunday
Logging	
MaxLogRows	10000

AdministrationNotesType

Audit Settings

- **Notes Type**
  - **Hide**
    Notes field is not displayed.
  - **Optional**
    Notes field is displayed, but is not required.
  - **Required**
    Notes field is displayed and is required.
- **Excluded Tables**
  Table Names to be excluded from the Audit Triggers
- **Reason Resources**
  The text that is displayed when prompting for a Signature

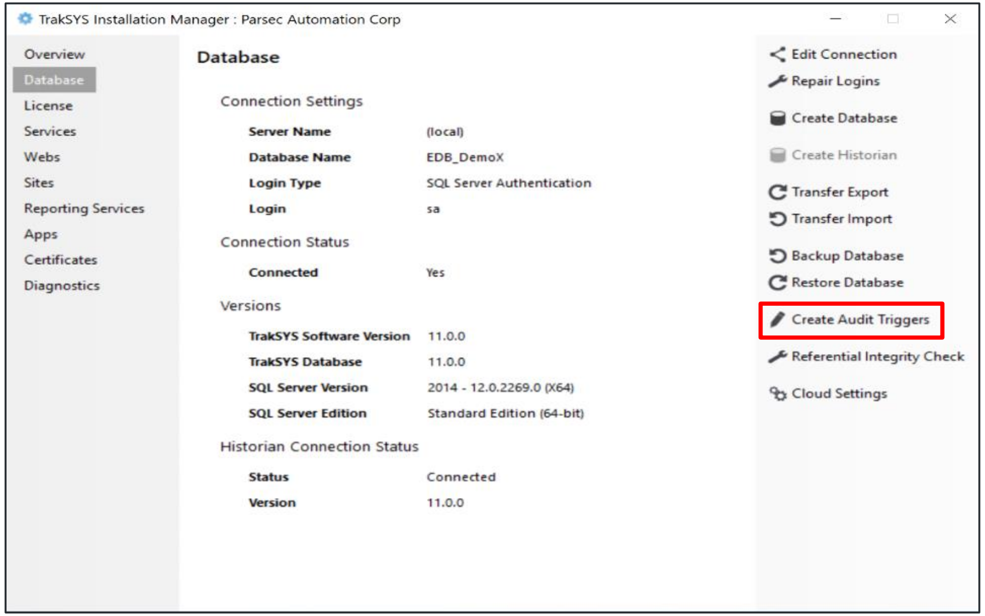


Audit	
AdministrationNotesType	Hide
AdministrationSignatureType	Auto
ConfigurationNotesType	Hide
ConfigurationSignatureType	Auto
DataNotesType	Hide
DataSignatureType	Auto
DeveloperNotesType	Hide
DeveloperSignatureType	Auto
ExcludedTables	
ReasonDeleteResource	resx:Audit.ReasonDelete, Core.Api
ReasonInsertResource	resx:Audit.ReasonInsert, Core.Api
ReasonUpdateResource	resx:Audit.ReasonUpdate, Core.Api
General	
IsTransferTarget	False
RootSiteTimeZone	UTC
WeekStartDay	Sunday
Logging	
MaxLogRows	10000

AdministrationNotesType

Audit Triggers

- Once Audit Settings are in place, the Database Triggers must be created
- Triggers can be created through the Installation Manager, under the Database tab
- Creating Triggers requires SQL credentials



TrakSYS Installation Manager : Parsec Automation Corp

Database

Connection Settings

Server Name	(local)
Database Name	EDB_DemoX
Login Type	SQL Server Authentication
Login	sa

Connection Status

Connected	Yes

Versions

TrakSYS Software Version	11.0.0
TrakSYS Database	11.0.0
SQL Server Version	2014 - 12.0.2269.0 (X64)
SQL Server Edition	Standard Edition (64-bit)

Historian Connection Status

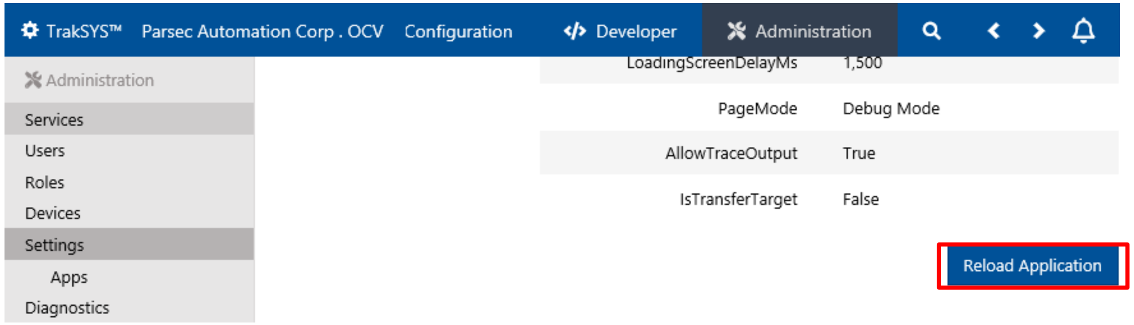
Status	Connected
Version	11.0.0

Audit Enforcement

In TS Web, Auditing may not be immediately enforced. The changes are stored in the database but are not checked on page that have been recently loaded.

After enabling new settings in the Installation Manager, it is recommended that the Application be reloaded to enforce the changes, including Auditing.

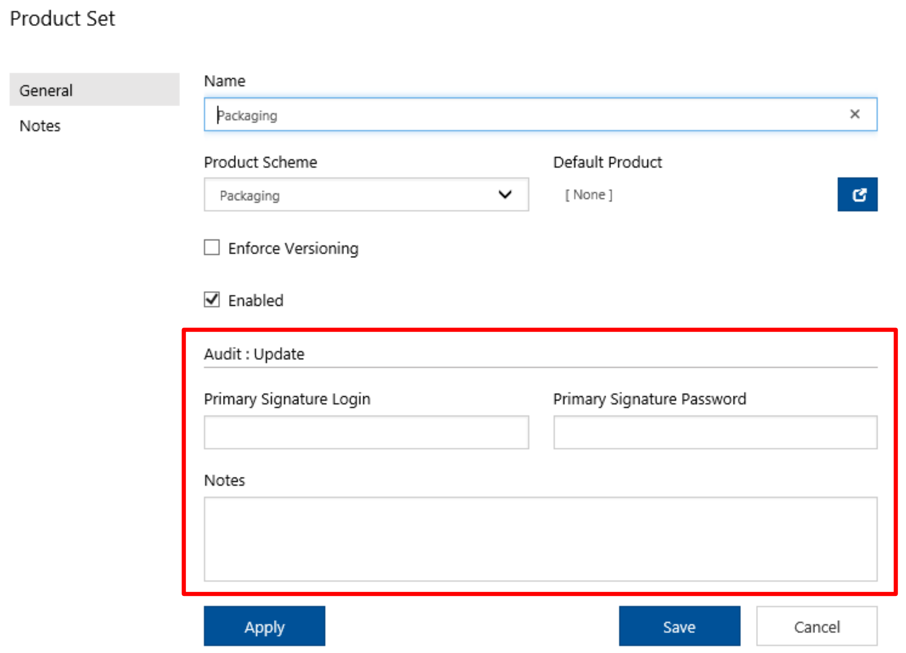
To reload the Application, go to the Administration Section and use the Settings Hub. Use the Reload Application button at the bottom of the page.



LoadingScreenDelayMs	1,500
PageMode	Debug Mode
AllowTraceOutput	True
IsTransferTarget	False

Audit Signature Interface

- Signature Fields are automatically displayed based on Audit Settings
- Reason is displayed in Audit Header
- Login and Password is verified against a TrakSYS User or Windows Credentials
- Audit Triggers automatically create a record including the Signature and the record Modifications


Product Set

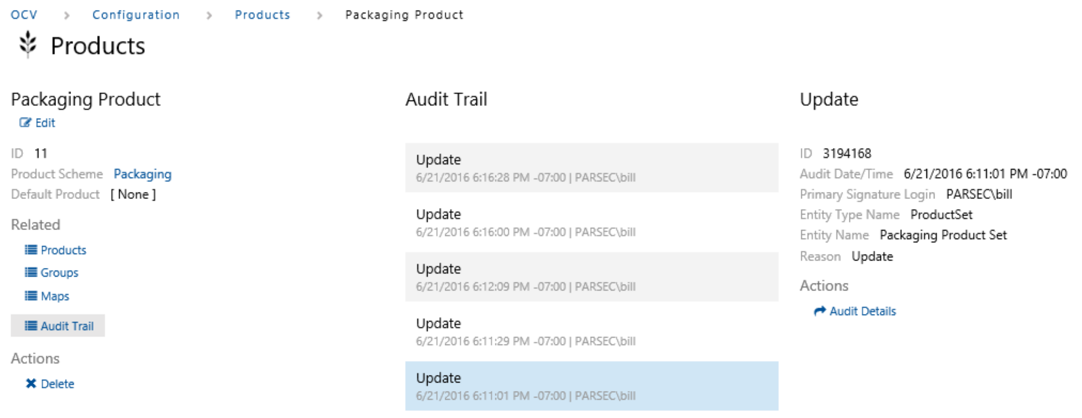
General
Notes

Name	Packaging
Product Scheme	Packaging
Default Product	[None]
Enforce Versioning	☐
Enabled	☑

Audit : Update

Primary Signature Login	
Primary Signature Password	
Notes	

Audit Results



OCV › Configuration › Products › Packaging Product

Products

Packaging Product

Edit

ID	11
Product Scheme	Packaging
Default Product	[None]

Related

- Products
- Groups
- Maps
- Audit Trail

Actions

- Delete

Audit Trail

Update
6/21/2016 6:16:28 PM -07:00 | PARSEC\bill

Update
6/21/2016 6:16:00 PM -07:00 | PARSEC\bill

Update
6/21/2016 6:12:09 PM -07:00 | PARSEC\bill

Update
6/21/2016 6:11:29 PM -07:00 | PARSEC\bill

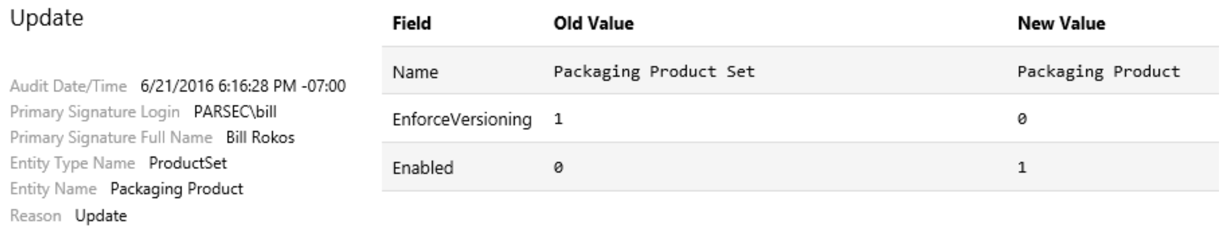
Update
6/21/2016 6:11:01 PM -07:00 | PARSEC\bill

Update

ID	3194168
Audit Date/Time	6/21/2016 6:11:01 PM -07:00
Primary Signature Login	PARSEC\bill
Entity Type Name	ProductSet
Entity Name	Packaging Product Set
Reason	Update

Actions

- Audit Details



Update

Audit Date/Time	6/21/2016 6:16:28 PM -07:00
Primary Signature Login	PARSEC\bill
Primary Signature Full Name	Bill Rokos
Entity Type Name	ProductSet
Entity Name	Packaging Product
Reason	Update

Field	Old Value	New Value
Name	Packaging Product Set	Packaging Product
EnforceVersioning	1	0
Enabled	0	1

Demonstration

- Show the Root Site
- Highlight the Entities that are Shared vs Site Specific
- Configure a Translation Locale
- Configure a Resource Group
- Configure a Resource Item
- Configure a Translation
- Reference a Translation
- Configure Audit Triggers
- Enabled Audit Signatures
- Show the Audit Signature User Interface
- Show the Audit History
- Show an example of manipulating a Chart from the Script Editor

Lab 16