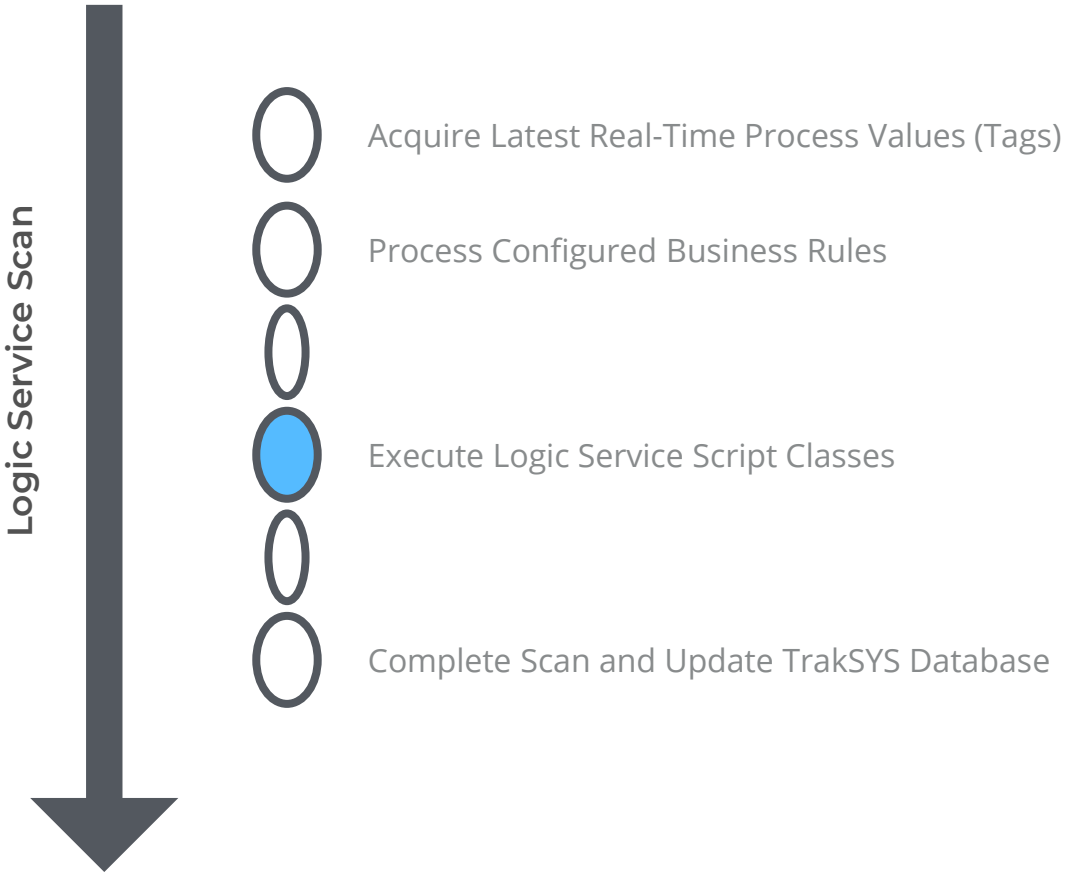
Overview

TrakSYS allows the definition of scripts that can be executed in-line with the Logic Service Real-Time data collection engine.

This allows a solution's business rules access to up to the second accurate automation values, as well as calculated information already processed by the Logic Service engine.

The diagram to the right shows a simplified depiction of a Logic Service scan and where the Script classes are hooked.



Script Class Types

When creating a Logic Service Script class, the following options are available...

Standard

Used to define common classes and methods to be referenced from other Logic Service Script classes.

Logic Service Script Class

Used to inject business rules into the startup, shutdown and on every scan of the Logic Service.

System Script Class

Used to inject business rules into various transitions for a specific System entity and its child Definitions (Event, Task, Function, etc...).

Event Definition Script Class
Function Definition Script Class
Task Definition Script Class
Transfer Definition Script Class
Sample Definition Script Class

Used to inject business rules into various transitions for a specific Definition.

KPI Calculation Script Class

Used to inject business rules into various transitions for a specific KPI/OEE Calculation.

Custom Rule Definition Script Class

Used to define custom Sample Definition Rule violations for real-time SPC data processing.

System Script Class Example

This example shows a System Script Class that sets an OPC Product Tag value when a Job is started. The OPC Product Tag is used by some labeler automation to print the correct Product Code on a label and must be set currently at the instant the Job begins.

The logic is added to the PostScanJobStart method which is called each time a Job is started…

(note the script is written with minimal error handling to promote clarity)

```
public override void PostScanJobStart(IPostScanJobStartContext context)
{
      // load some things
      var job = context.Api.Data.DbJob.Load.ByID(context.JobID);
      var product = context.Api.Data.DbProduct.Load.ByID(job.ProductID);
      var jsa = context.Api.Data.DbJobSystemActual.Load.ByID(context.JobSystemActualID);
      var sys = context.Api.Data.DbSystem.Load.ByID(jsa.SystemID);

      // calculate the tag name
      string productCodeTagName = "{0}_PRODUCT_CODE".FormatWith(sys.TemplateTagPrefix);

      // write the tag
      context.OpcWriteBack(productCodeTagName, product.ProductCode);
}
```

Attaching Logic Service Script Classes

Once a Logic Service Script Class is defined, it must be bound to the appropriate entity (and the Logic Service restarted) for its business rules to be loaded and executed.


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

namespace ETS.Core.Scripting
{
    /// ************************************************************
    /// <summary>
    /// This class contains code that can be executed by a System
    /// instance that is attached using the Script Class Name setting.
    /// </summary>
    /// ************************************************************
    public class PackagingLineSystemScript : ETS.Core.Scripting.SystemScriptClassBase
```