Complex Event Processing (CEP) with PI for StreamInsight

Presented By:

Roman Schindlauer - Microsoft
Erwin Gove – OSIsoft
Greg Douglas - Logica

Talk Outline

- Microsoft StreamInsight Overview

- PI for StreamInsight Overview

- Microsoft and PI for StreamInsight
  - User scenarios and code examples

©2010 OSIsoft, LLC. All Rights Reserved

PI Analytics - Subset & Features

calculations

event based

PI data

- Performance Equations
- Totalizers
- Alarm/Statistical Quality Control
- PI Advanced Calculation Engine
- PI for StreamInsight*
- Asset Framework supported Analytics*

Sch

User Configuration

Programme

Non-PI data

* future product

©2010 OSIsoft, LLC. All Rights Reserved

Microsoft StreamInsight

Presented By:

Roman Schindlauer
Program Manager - Microsoft

Understanding Streaming Data

- Question: “how many red cars are in the parking lot”.
- Answering with a relational database:
  - – Walk out to the parking lot.
  - – Count vehicles that are
    - Red
    - Cars

```
SELECT COUNT(*) FROM ParkingLot
WHERE type = ‘AUTO’
AND color = ‘RED’
```

©2010 OSIsoft, LLC. All Rights Reserved

Understanding Streaming Data

- What about: “How many red cars have passed the 40th street exit on the 520 in the last hour”?
- Answering with a relational database:
  - Pull over and park all vehicles in a lot, keeping them there for an hour.
  - Count vehicles that are in the lot.

Doesn’t seem like a great solution…

©2010 OSIsoft, LLC. All Rights Reserved

Understanding Streaming Data

- Different kinds of questions require different ways of answering them.
- Answering the question with a streaming data processing engine:
  - Stand by the freeway, count red cars as they pass by.
  - Write down the answer, deliver the answer.

This is the streaming data paradigm in a nutshell – ask questions about data in flight.

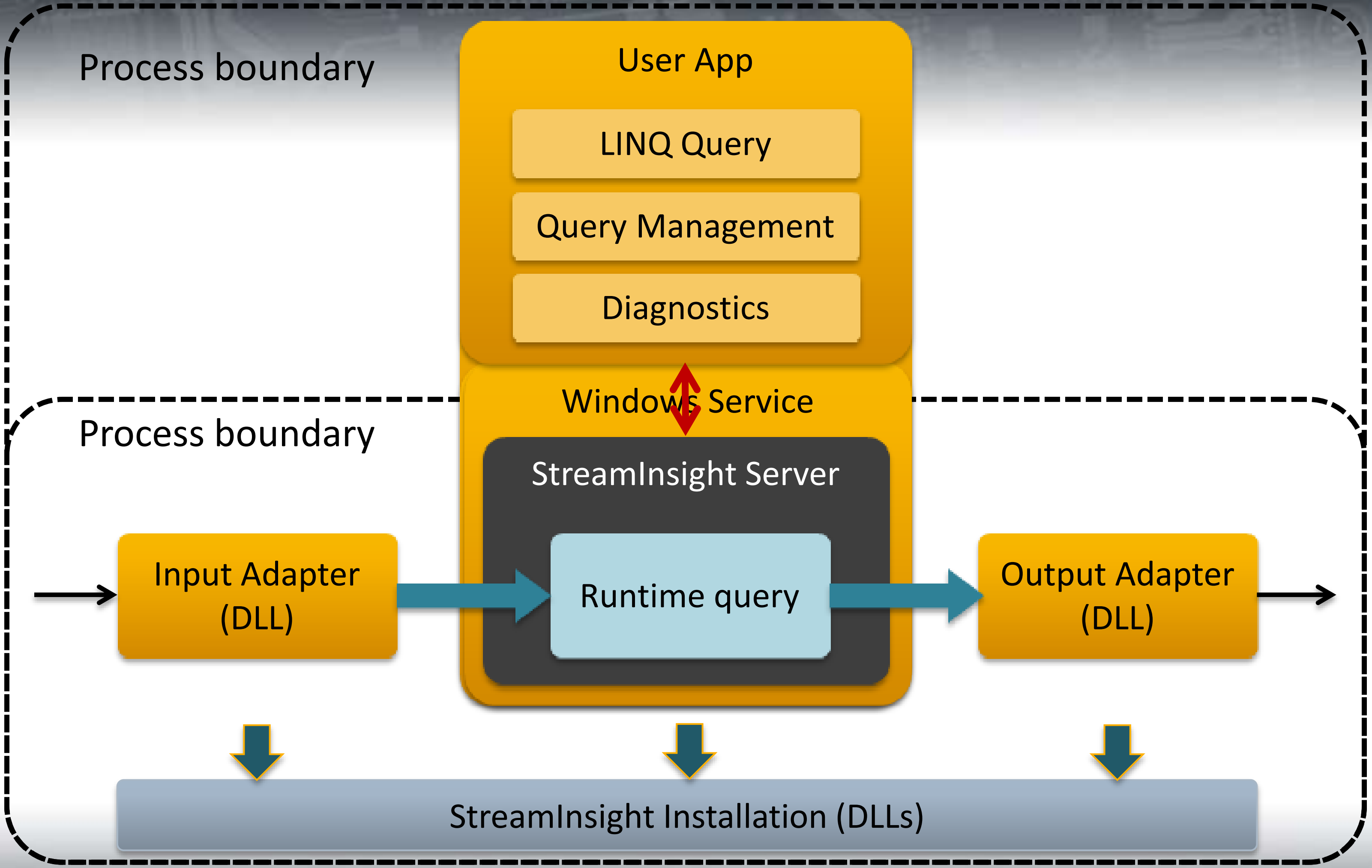
©2010 OSIsoft, LLC. All Rights Reserved

What is StreamInsight

- API to build CEP applications
- Continuous and incremental processing
  - High throughput, low latency
  - Event-driven computation
- Declarative query language (LINQ)
- Adapter model
- Diagnostic interface
- Extensibility model
- Needs a SQL Server 2008 R2 License
  - Datacenter
  - Standard, Enterprise

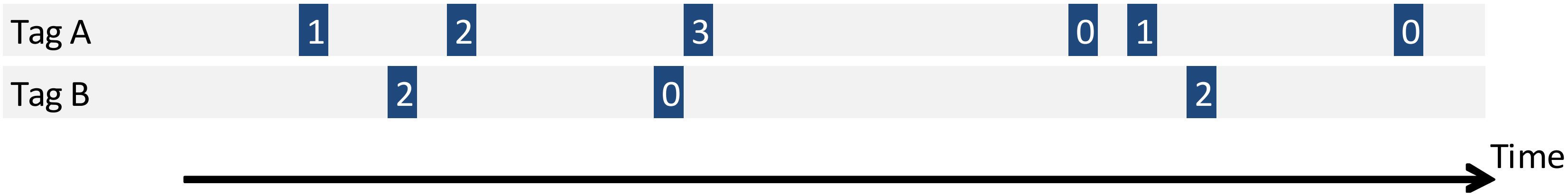
©2010 OSIsoft, LLC. All Rights Reserved

StreamInsight App Architecture



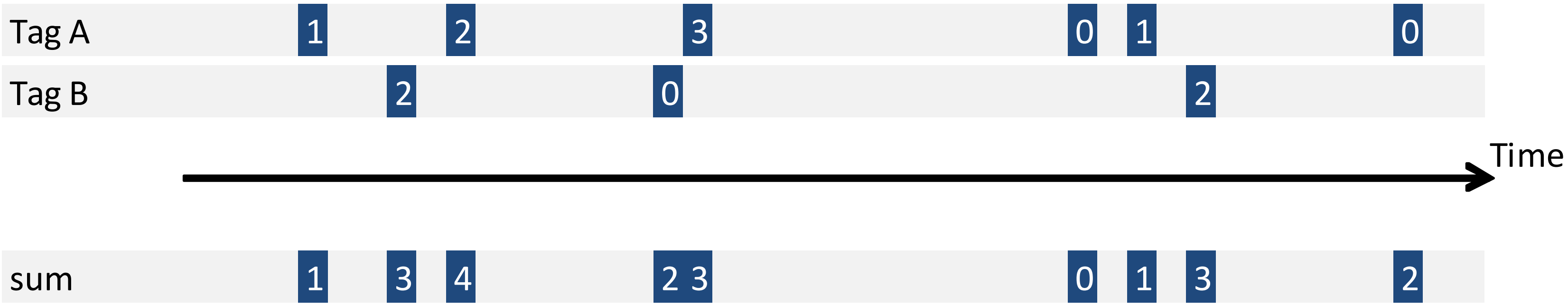
©2010 OSIsoft, LLC. All Rights Reserved

Example: Add tags

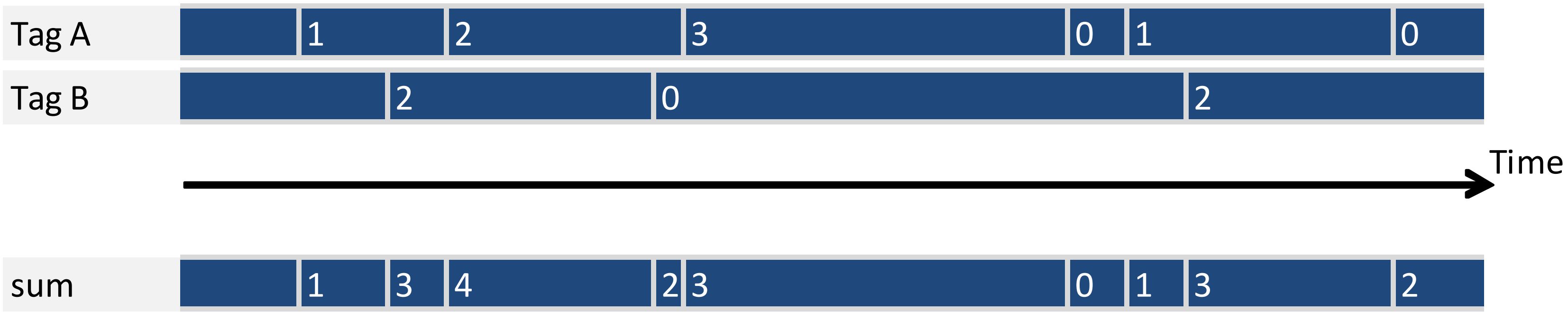


©2010 OSIsoft, LLC. All Rights Reserved

Example: Add tags



Example: Add tags



Tag A | 1 | 2 | 3 | 0 | 1 | 0

Tag B | 2 | 0 | 2

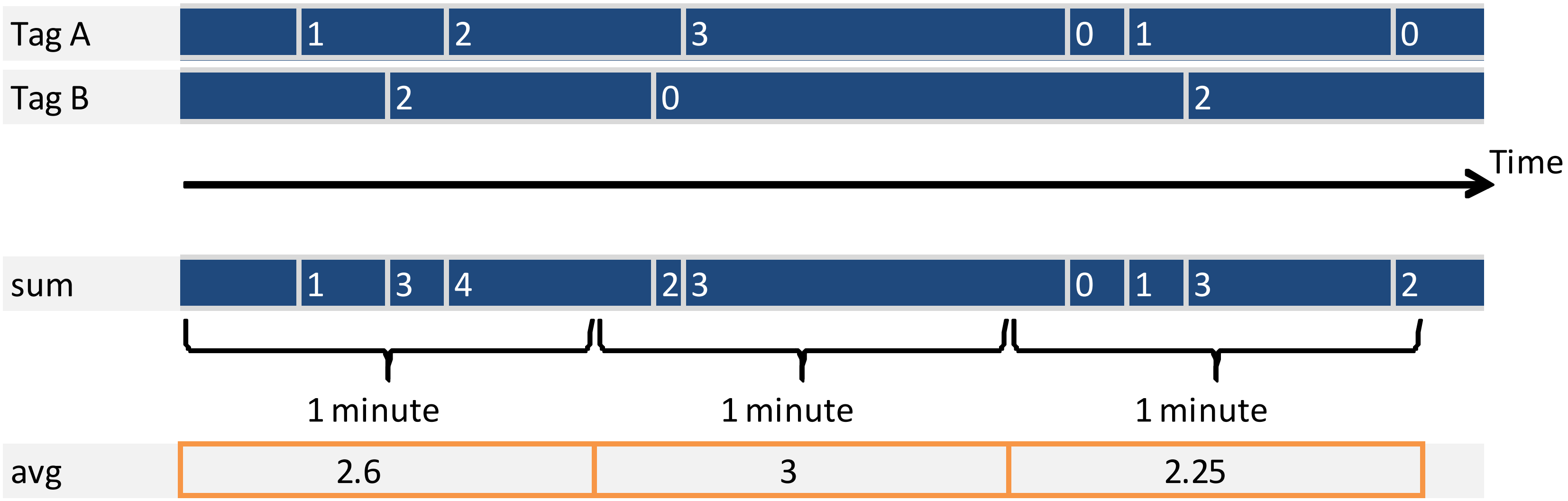
Time

sum | 1 | 3 | 4 | 2 | 3 | 0 | 1 | 3 | 2

```
from a in TagA
from b in TagB
select { sum = a.Value + b.Value };
```

©2010 OSIsoft, LLC. All Rights Reserved

Window & Aggregate



```
from window in sumstream.HoppingWindow(TimeSpan.FromMinutes(1))
select new { avg = window.Avg(e => e.sum) };
```

©2010 OSIsoft, LLC. All Rights Reserved

PI for StreamInsight

Presented By:

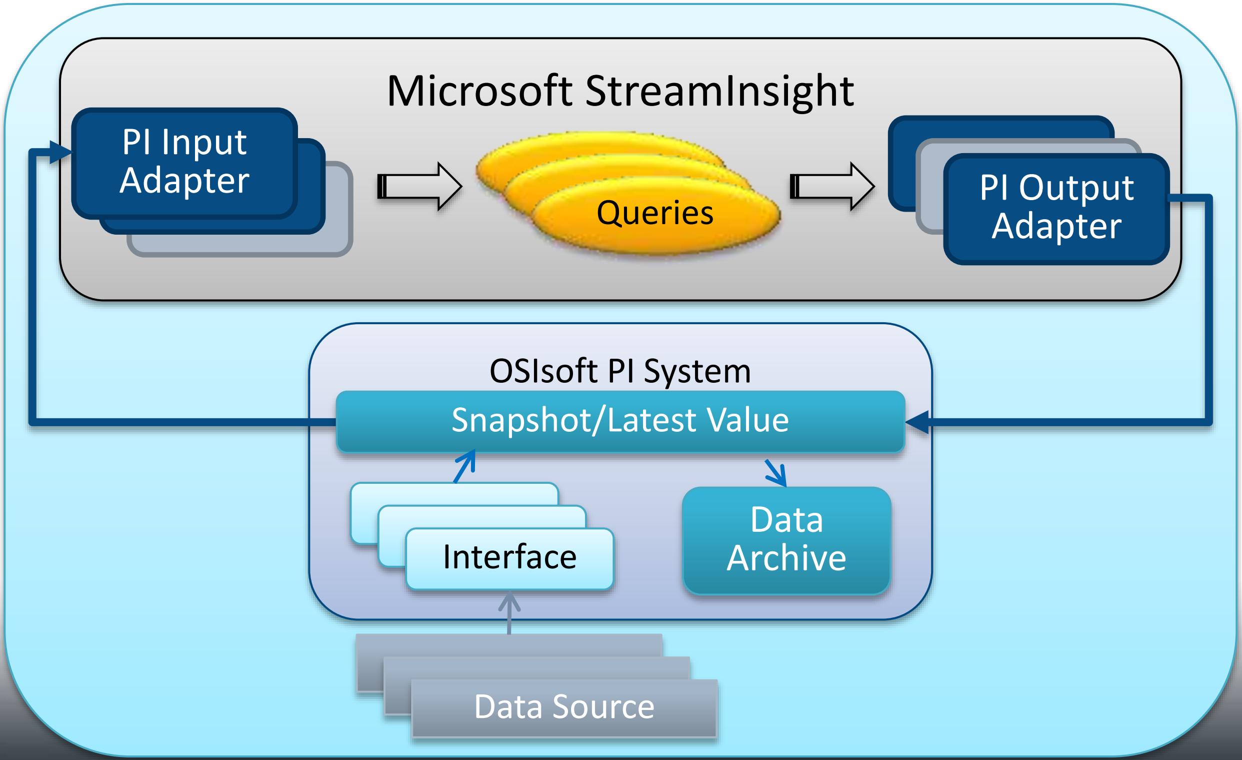
Erwin Gove
Development Lead - OSIsoft

OSIsoft®

©2010 OSIsoft, LLC. All Rights Reserved

PI for StreamInsight

- Enable data access to the PI System from Microsoft StreamInsight



©2010 OSIsoft, LLC. All Rights Reserved

Configure a PI Input Adapter

```
// 1. Create an Input adapter configuration

SnapshotInputConfig inputConfig = new
SnapshotInputConfig();


// 2. Specify configuration basics

    inputConfig.Server = “PIServer”;

// 3. Specify points, by query
    inputConfig.PointsQuery =
      PISearch.TAG.eq("AlarmTest.Input.*") +
      PISearch.TAG.eq("Test.Input.Float32.*");
```

©2010 OSIsoft, LLC. All Rights Reserved

// Create a PI Input Adapter Stream

Create a StreamInsight CEP Stream by specifying

1. A uniform event type (PIEvent<Double> below)
2. The Adapter Factory that will be used to control the input adapter
3. The event shape (EventShape.Point below)

```
var rawStream = CepStream<PIEventBasic<double>>.Create(
      “Alarm Stream",
      typeof(SnapshotInputFactory),
      inputConfig,
      EventShape.Point);
```

Event Payload

Event payload can be defined by implementer

A number of events are provided for convenience

```
public class PIEvent<T>
{
    public int Id { get; set; } // PI point identifier
    public string Path { get; set; } // PI path (tag)
    public T Value { get; set; }
    public int Status { get; set; } // event status
    public bool IsEdited { get; set; } // was the point edited
    public bool IsQuestionable { get; set; }
}
```

Digitals include ValueText

note: not all payloads are shown

©2010 OSIsoft, LLC. All Rights Reserved

// Configure a PI Output Adapter

```
// 1. Create an Output adapter configuration
      SnapshotOutputConfig outputConfig =
         new SnapshotLOutputConfig();

// 2. Specify configuration basics
      outputConfig.Server = “OutputPIServer”;
```

Note: the output tags/paths are generated within the query

©2010 OSIsoft, LLC. All Rights Reserved

// Specify the PI Output Adapter to be used as the CEP Stream is turned into a query

Use the ToQuery() method to convert the CEP stream into a Linq Query by specifying

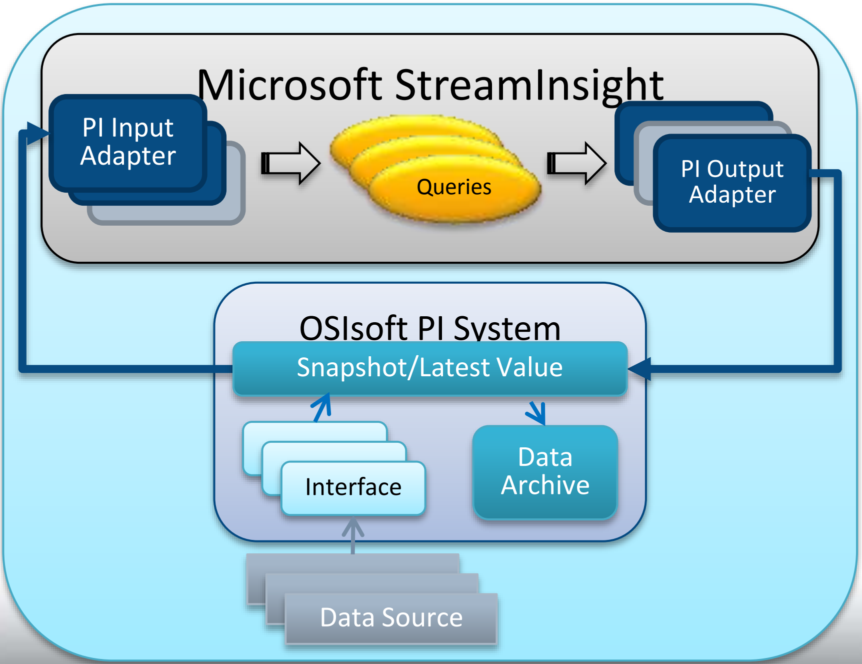
1. The Adapter Factory that will be used to control the output adapter
2. The output adapter configuration
3. The event shape (EventShape.Point below)

```
var query = alarmStream.ToQuery(
   app, “Alarm Query", “Alarm query sample",
   typeof(SnapshotOutputFactory),  outputConfig,
   EventShape.Point, StreamEventOrder.FullyOrdered);
```

©2010 OSIsoft, LLC. All Rights Reserved

PI for StreamInsight – Version 1.0

- Support for
  - Read and write to PI points
  - Tag search
  - Snapshot input data



©2010 OSIsoft, LLC. All Rights Reserved

Requirements

- OSIsoft PI Server 2010
  - which will include PI for StreamInsight when released
- Microsoft StreamInsight
  - Included with SQL Server 2008 R2
  - StreamInsight can be run as a standalone component without SQL Server
- Programming experience in .NET and LINQ

©2010 OSIsoft, LLC. All Rights Reserved

Where can I get it?

- A Community Technology Preview (CTP) of PI for StreamInsight is available on OSIsoft vCampus
- PI for StreamInsight is due for release later this year

©2010 OSIsoft, LLC. All Rights Reserved

PI for StreamInsight

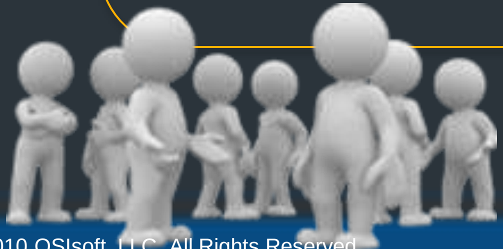
- More information
  - OSIsoft vCampus
    - Blog
    - Webinar
    - StreamInsight examples

©2010 OSIsoft, LLC. All Rights Reserved

PI for StreamInsight in action

Presented By:

Greg Douglas
Technical Architect – Logica
greg.douglas@logica.com



©2010 OSIsoft, LLC. All Rights Reserved

Logica

- **Logica** is a **business and technology service** company, employing 39,000 people across 36 countries. We deliver **business consulting, systems integration** and **outsourcing** across all industries and business functions
- 30 years of Manufacturing IT Excellence
- Logica provides services and support to global PI System customers that include:
  - RT Architecture and PI System Implementation
  - Project Management and Training
  - Maintenence and Support
  - Proof of Concept Management
- The PI System is running in multiple Logica Innovation Centers worldwide

©2010 OSIsoft, LLC. All Rights Reserved

Logica and StreamInsight

- Logica is a leading SI for Microsoft StreamInsight and OSIsoft PI for StreamInsight
- We are using the powerful pair to develop Innovative CEP solutions in our Houston Innovation Center
- In the final steps of completing a Proof of Concept for Shell

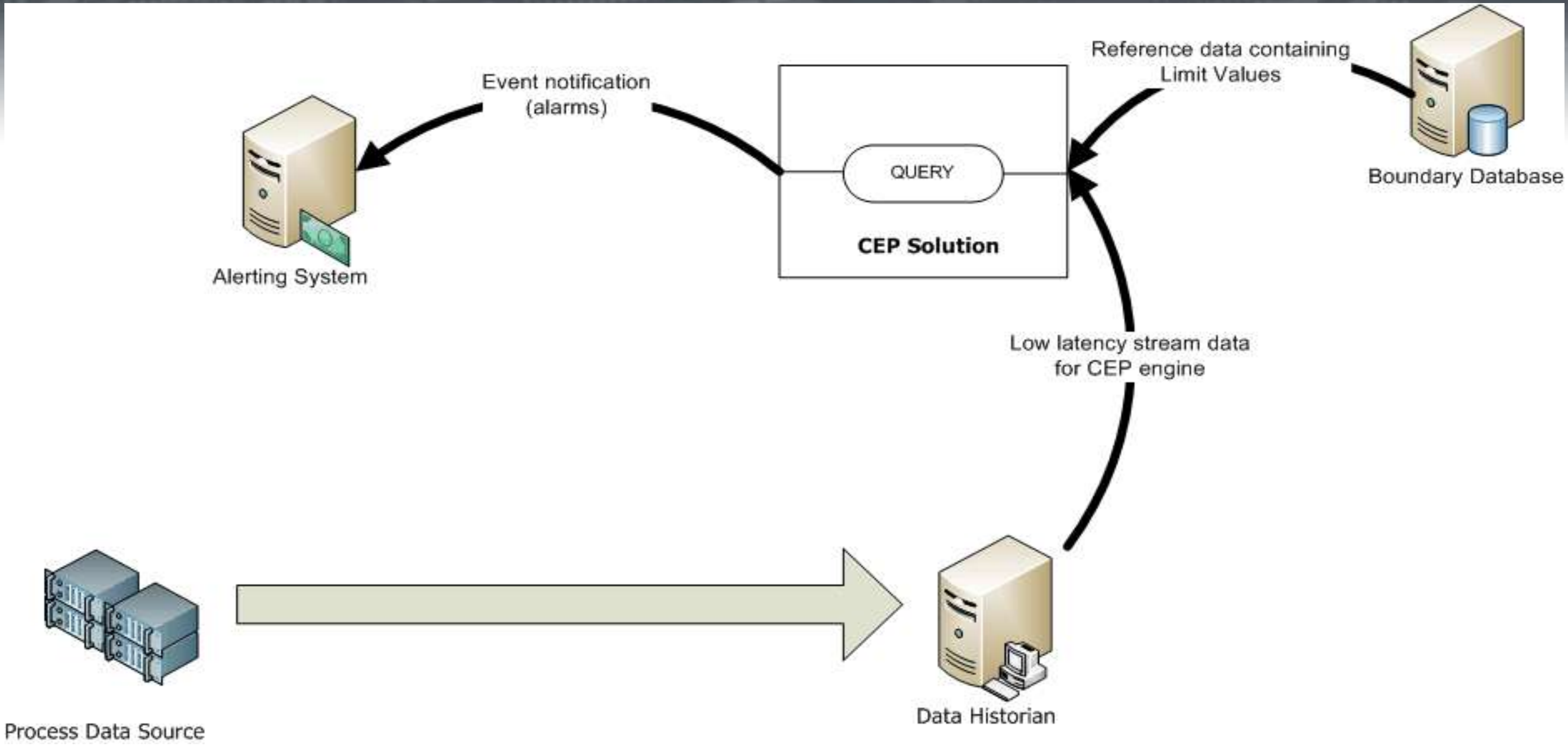
©2010 OSIsoft, LLC. All Rights Reserved

Test Scenarios

- **Data Handling –** High Frequency Data Collection
  - Buffer high-speed process data; detect an event and trigger the integration of high-speed data values into the historian.
- **Time Window -** Continuous Time Window
  - Detect pattern-based process data events (e.g. limit exceedance) in moving time windows and report the non-conformance immediately
- **Data Quality –** Data Cleansing
  - Analyze high speed process data and associated diagnostic information to identify suspect data; call routines to “cleanse” suspect data and output the processed data and diagnostic information; recognize critical events that need to be passed on immediately.

©2010 OSIsoft, LLC. All Rights Reserved

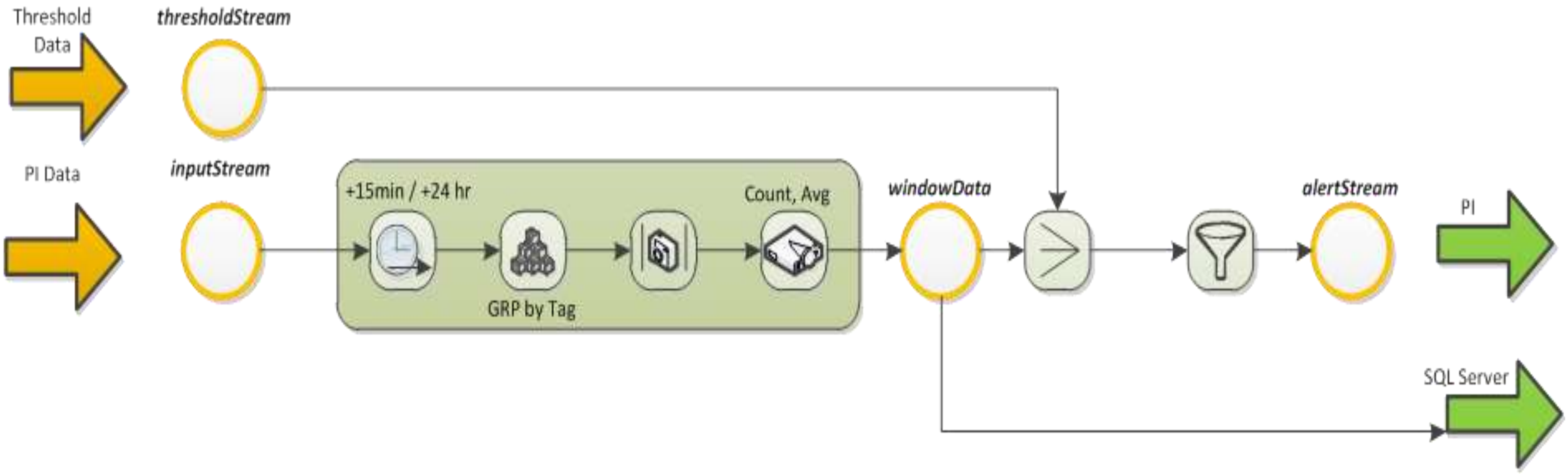
Time Window


Reference data containing Limit Values
Event notification (alarms)
Boundary Database
QUERY
CEP Solution
Alerting System
Low latency stream data for CEP engine
Process Data Source
Data Historian

Detect pattern-based process data events (e.g. limit exceedance) in moving time windows and report the non-conformance immediately

©2010 OSIsoft, LLC. All Rights Reserved

Time Window Query Design



©2010 OSIsoft, LLC. All Rights Reserved

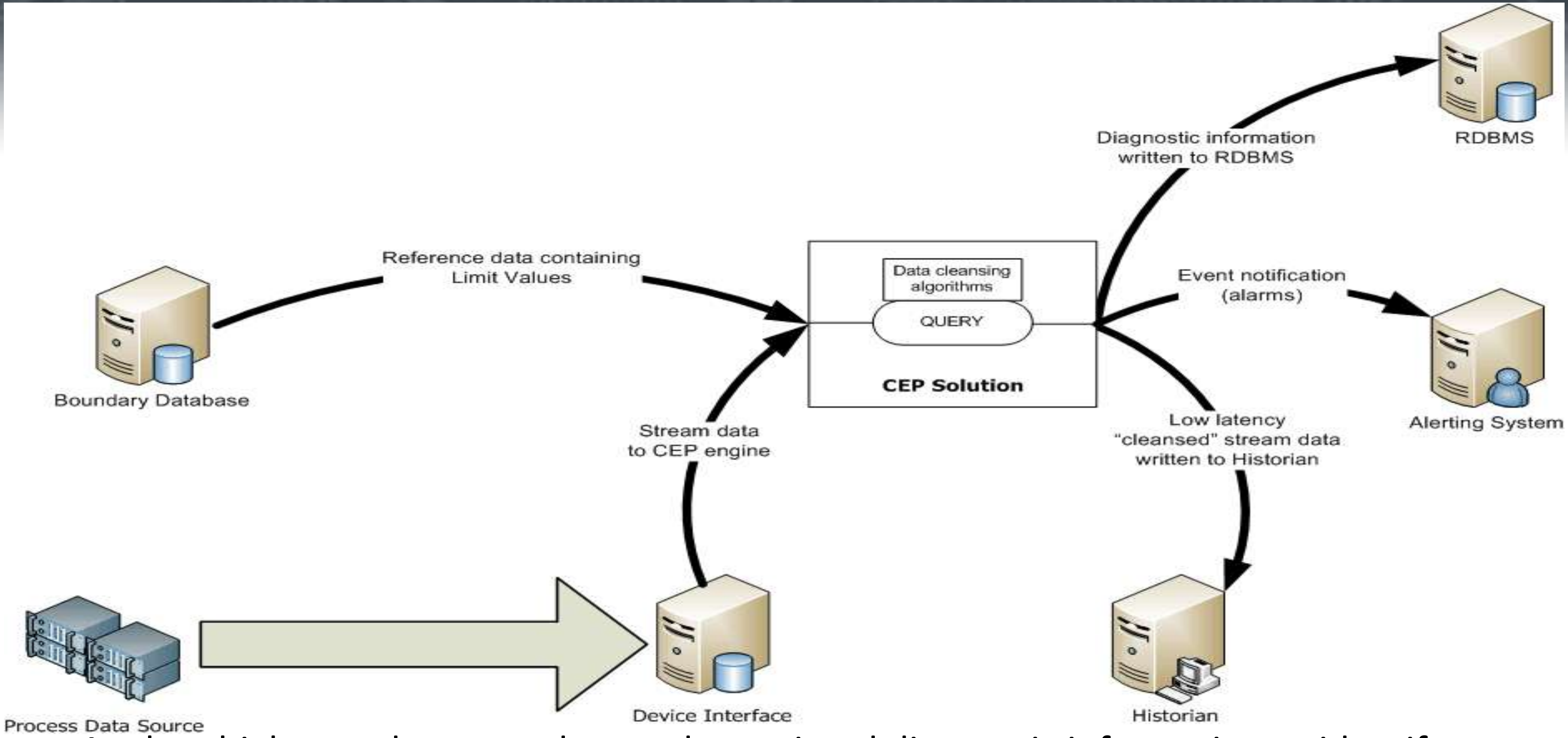
Time Windows Performance

- 52,000 Windows
  - 15 min window w/ 5 sec updates
  - 24 hour window with 1 min updates
  - Ranging between 20k and 32k tags

- Sustained average CPU Utilization was 70%
  - Real-time utilization bursty and efficiently balanced

©2010 OSIsoft, LLC. All Rights Reserved

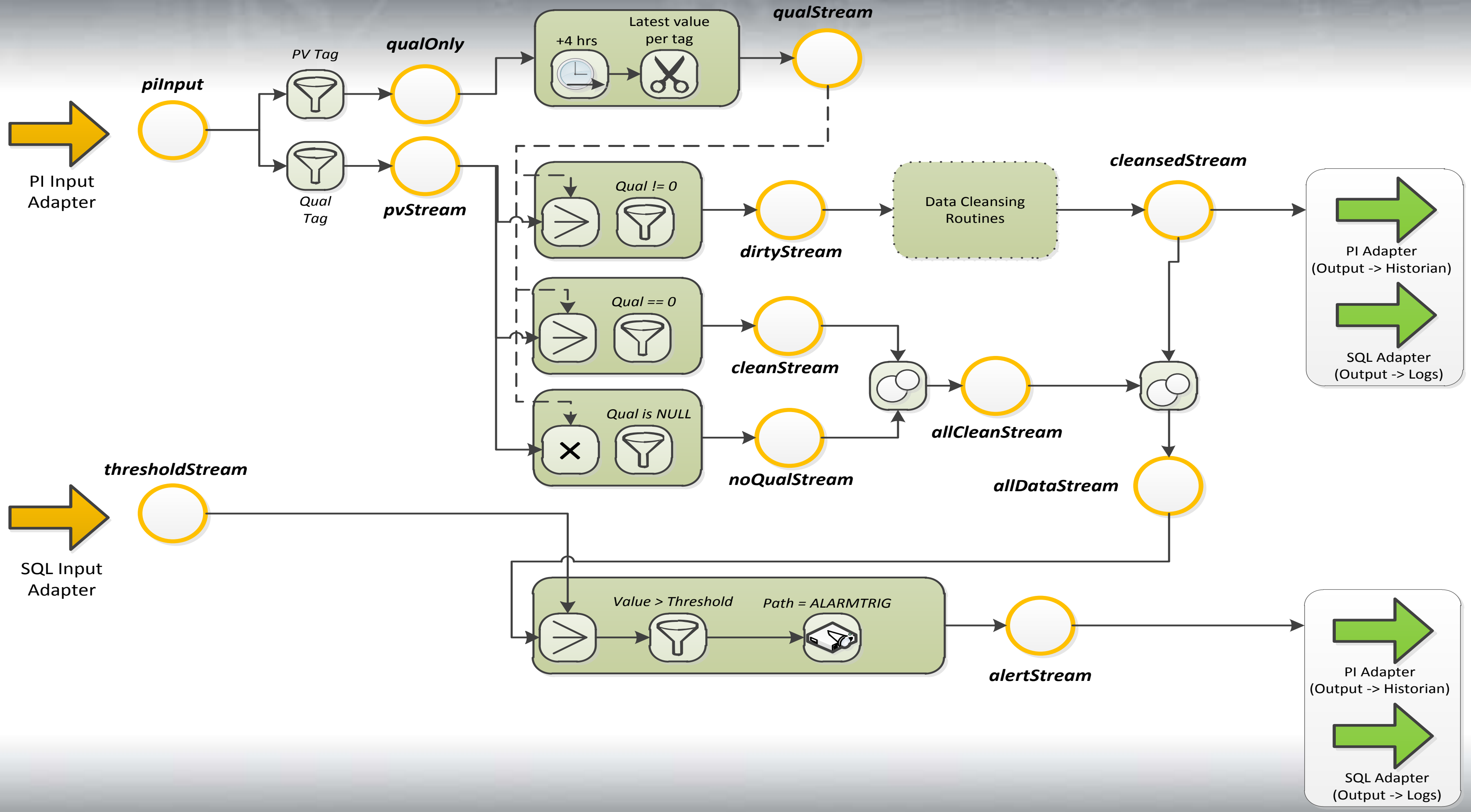
Data Quality



Analyze high speed process data and associated diagnostic information to identify suspect data; call routines to "cleanse" suspect data and output the processed data and diagnostic information; recognize critical events that need to be passed on immediately.

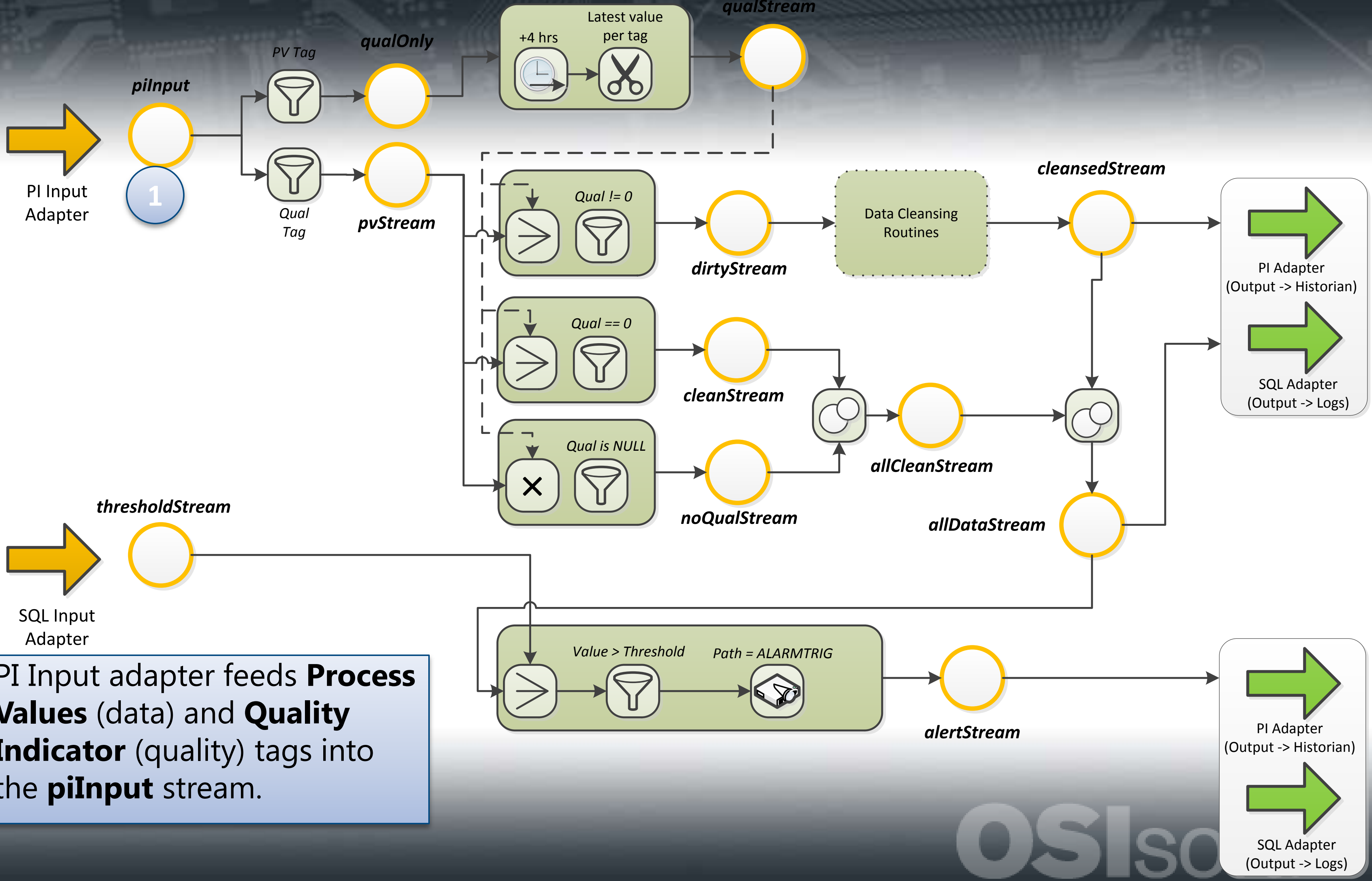
©2010 OSIsoft, LLC. All Rights Reserved

Data Quality Query



©2010 OSIsoft, LLC. All Rights Reserved

Data Quality



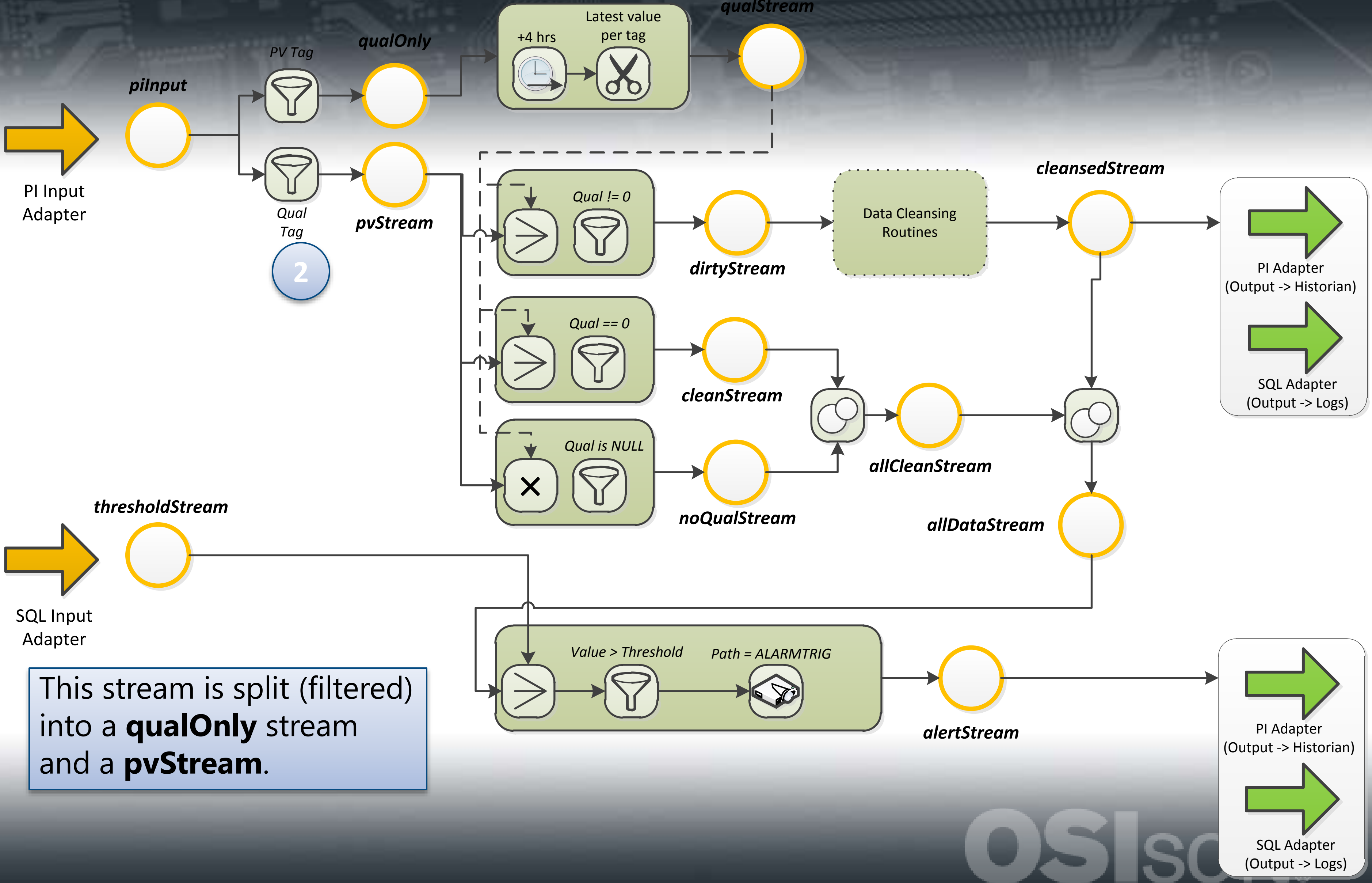
PI Input adapter feeds **Process Values** (data) and **Quality Indicator** (quality) tags into the **piInput** stream.

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality - Step 1

```
       // Create the data quality input stream PV and QUAL
tags from
         // the PI server.
         var dataQualityInputStream =
          CepStream<PIEvent<Double>>.Create("piInput",
             typeof(PIInputAdapterFactory),
          dataQualityConfig, EventShape.Point);
```

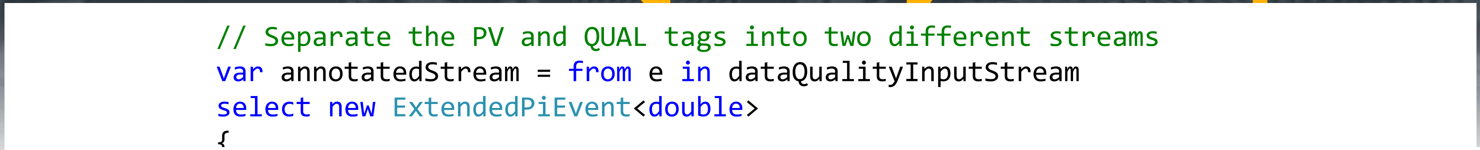
Data Quality



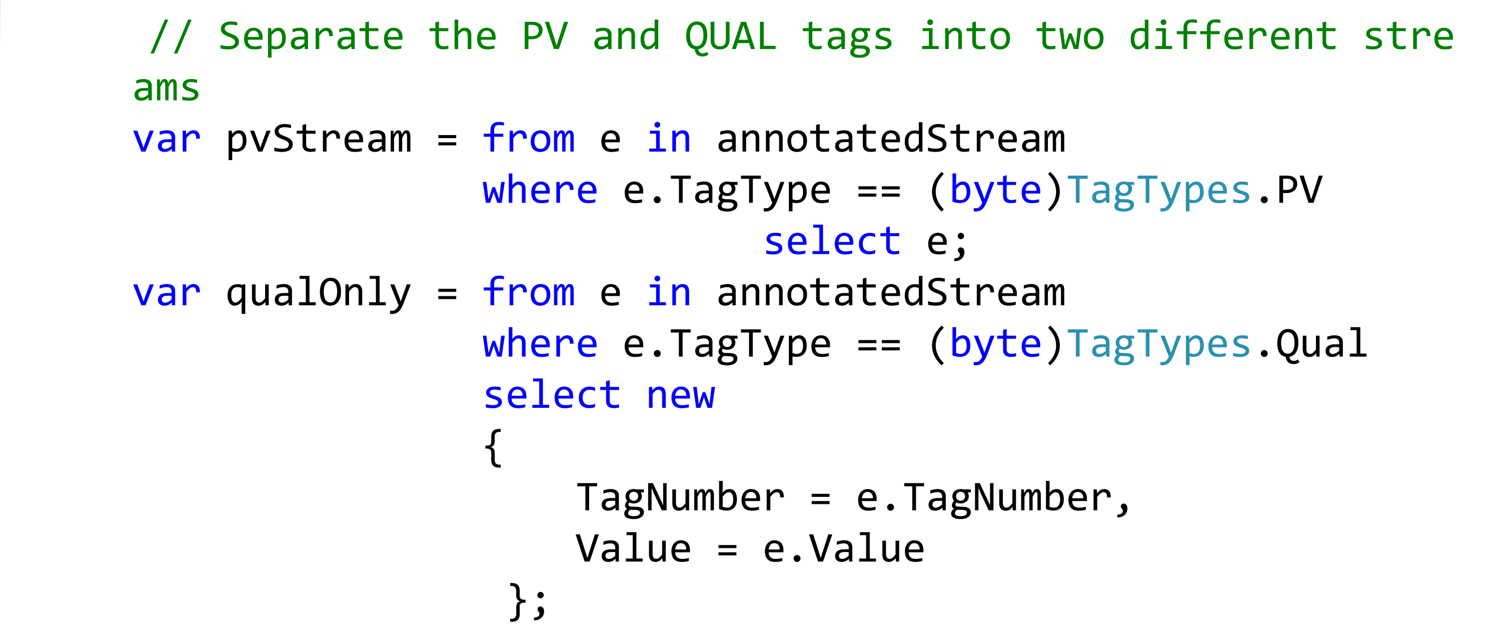
This stream is split (filtered) into a **qualOnly** stream and a **pvStream**.

©2010 OSIsoft, LLC. All Rights Reserved

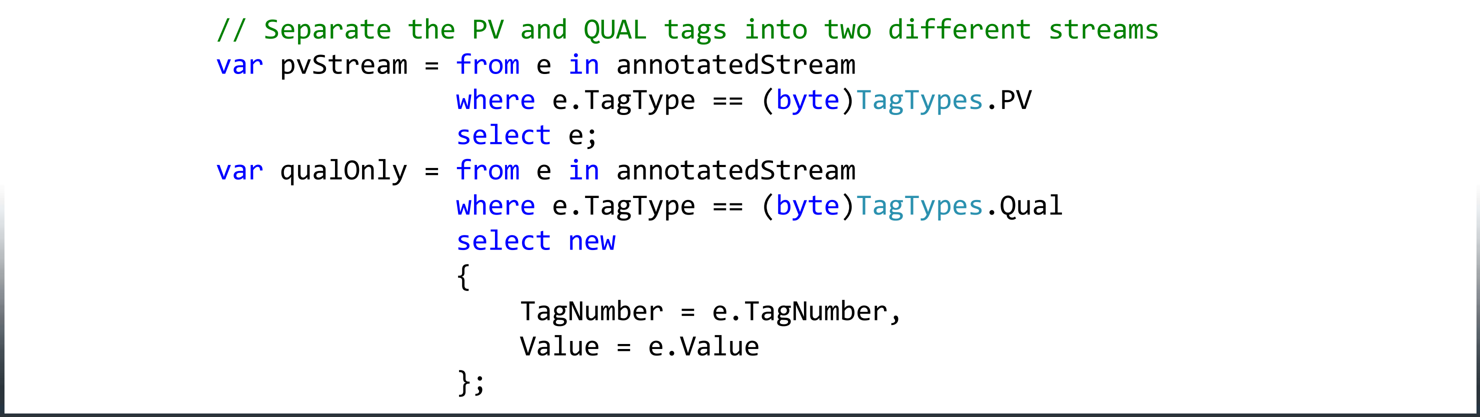
Data Quality – Step 2



```
// Separate the PV and QUAL tags into two different streams
var annotatedStream = from e in dataQualityInputStream
select new ExtendedPiEvent<double>
{
```



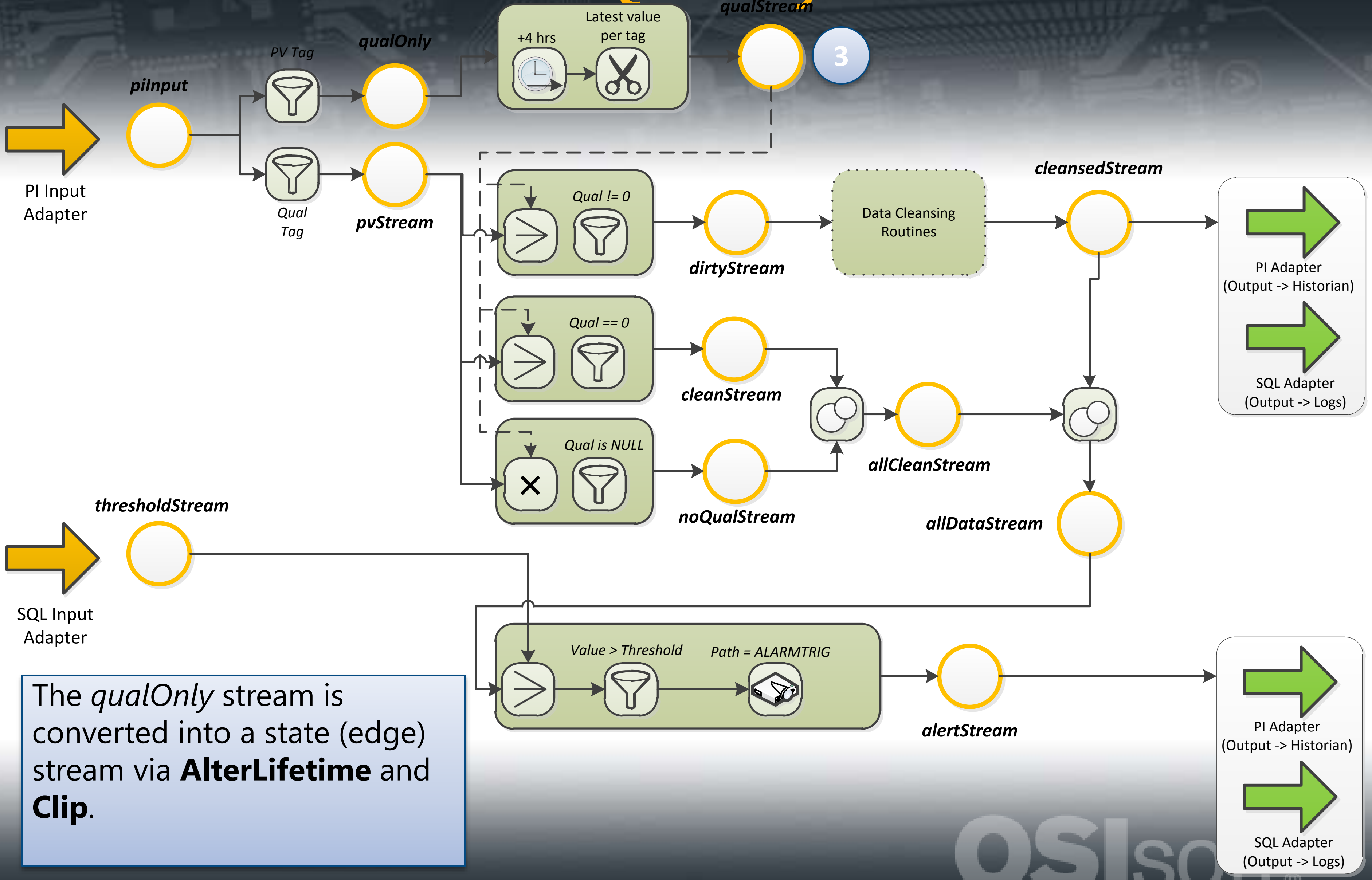
```
 // Separate the PV and QUAL tags into two different stre
ams
var pvStream = from e in annotatedStream
               where e.TagType == (byte)TagTypes.PV
                           select e;
var qualOnly = from e in annotatedStream
               where e.TagType == (byte)TagTypes.Qual
               select new
               {
                   TagNumber = e.TagNumber,
                   Value = e.Value
                };
```



```
// Separate the PV and QUAL tags into two different streams
var pvStream = from e in annotatedStream
               where e.TagType == (byte)TagTypes.PV
               select e;
var qualOnly = from e in annotatedStream
               where e.TagType == (byte)TagTypes.Qual
               select new
               {
                   TagNumber = e.TagNumber,
                   Value = e.Value
               };
```

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality



The *qualOnly* stream is converted into a state (edge) stream via **AlterLifetime** and **Clip**.

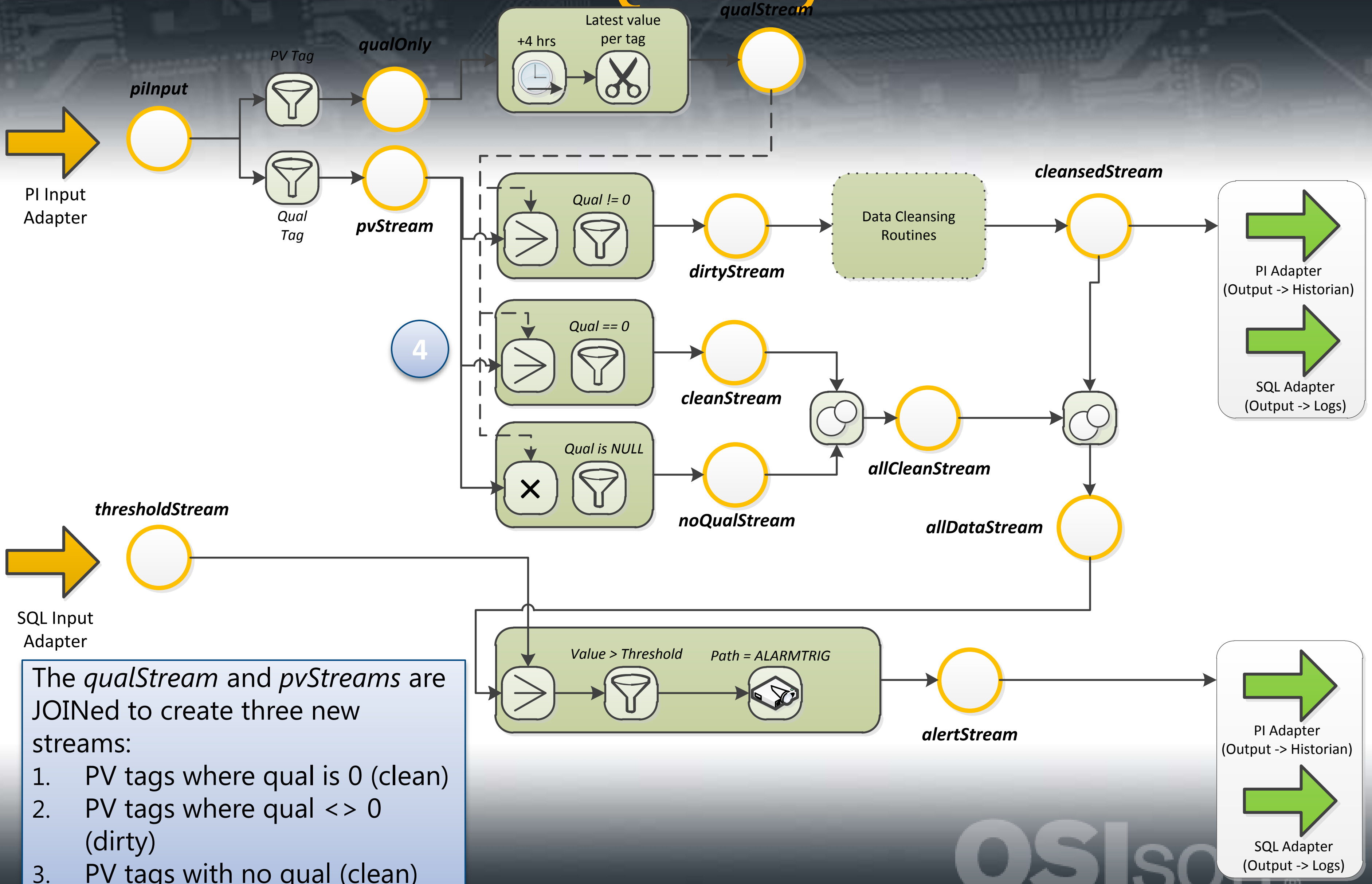
©2010 OSIsoft, LLC. All Rights Reserved

Data Quality – Step 3

```
// Convert the QUAL tags into an "edge" stream by extending the
// events to infinity, then "clipping" them off when the next
// matching path value arrives
   var qualStream = from e in qualOnly.AlterEventDuration(e => TimeSpan.FromHours(4))
                             .ClipEventDuration(qualOnly,
                             (e1, e2) => e1.TagNumber == e2.TagNumber)
                             select e;
```

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality



The *qualStream* and *pvStreams* are JOINed to create three new streams:

1. PV tags where qual is 0 (clean)
2. PV tags where qual <> 0 (dirty)
3. PV tags with no qual (clean)

Data Quality – Step 4.1

```
   // Qual bit == 0 - clean
// Qual bit <> 0 - dirty
    var dirtyStream = from pv in pvStream
                 join qual in qualStream
                on pv.TagNumber equals qual.TagNumber
                where qual.Value != 0
             select new ExtendedPiEvent<double>
             {
                 Annotation = pv.Annotation,
                 Id = pv.Id,
                 IsAnnotated = pv.IsAnnotated,
                 IsEdited = pv.IsEdited,
                 IsQuestionable = pv.IsQuestionable,
                 Path = pv.Path,
                 Status = pv.Status,
                 StatusText = pv.StatusText,
                 Timestamp = pv.Timestamp,
                 QualityFlag = qual.Value,
                 Value = pv.Value,

                 TagNumber = pv.TagNumber,
                 TagType = pv.TagType,
             };
```

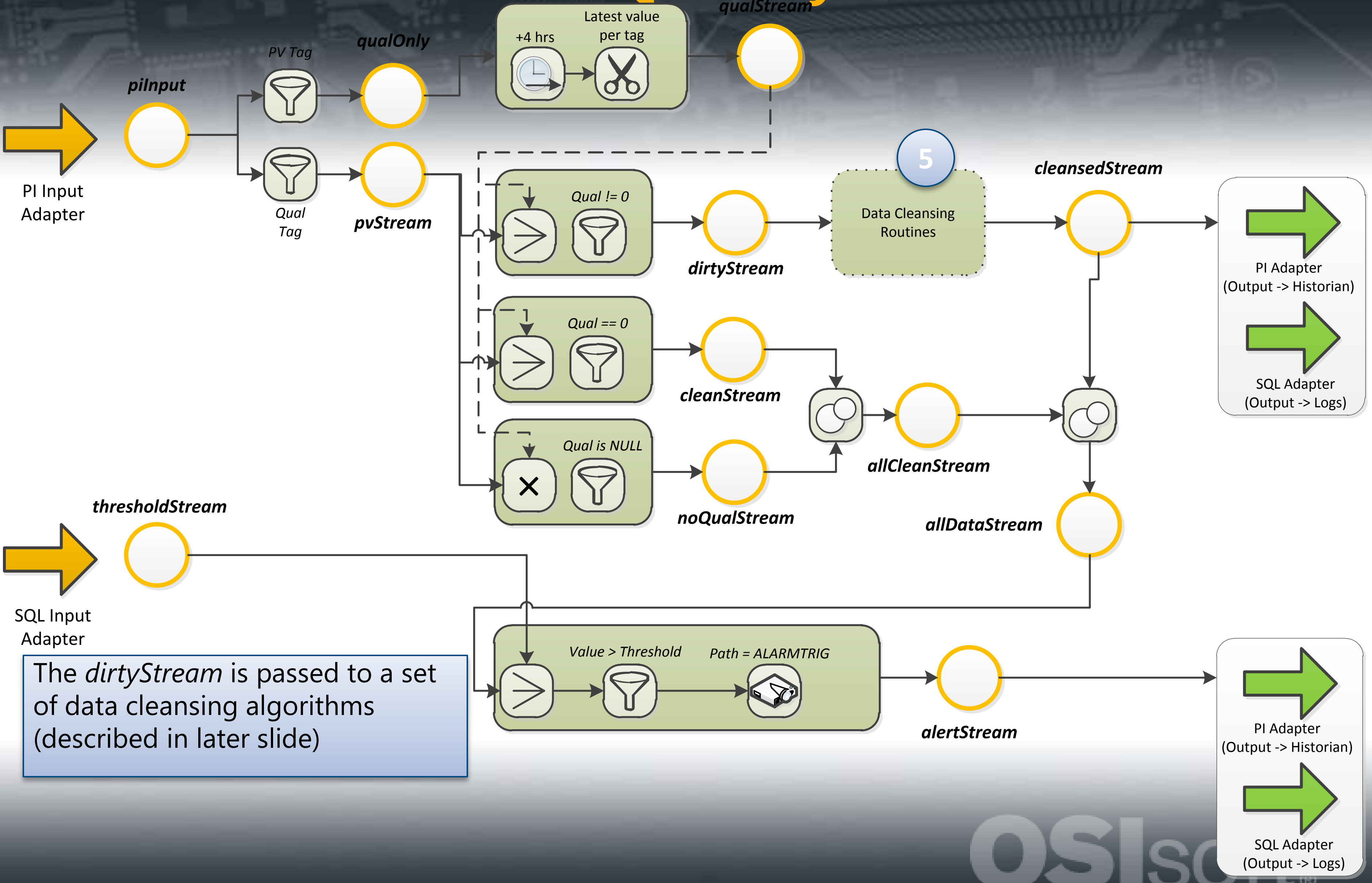
©2010 OSIsoft, LLC. All Rights Reserved

Data Quality – Step 4.2

```
// No qual bit - clean
  var noQualStream = from pv in pvStream
      where (from qual in qualStream
              where pv.TagNumber == qual.TagNumber
      select qual).IsEmpty()
      select new ExtendedPiEvent<Double>
      {
          Annotation = pv.Annotation,
          Id = pv.Id,
          IsAnnotated = pv.IsAnnotated,
          IsEdited = pv.IsEdited,
          IsQuestionable = pv.IsQuestionable,
          Path = pv.Path,
          Status = pv.Status,
          StatusText = pv.StatusText,
          Timestamp = pv.Timestamp,
          Value = pv.Value,
          TagNumber = pv.TagNumber,
          TagType = pv.TagType,
          QualityFlag = 0
      };
```

©2010 OSIsoft, LLC. All Rights Reserved

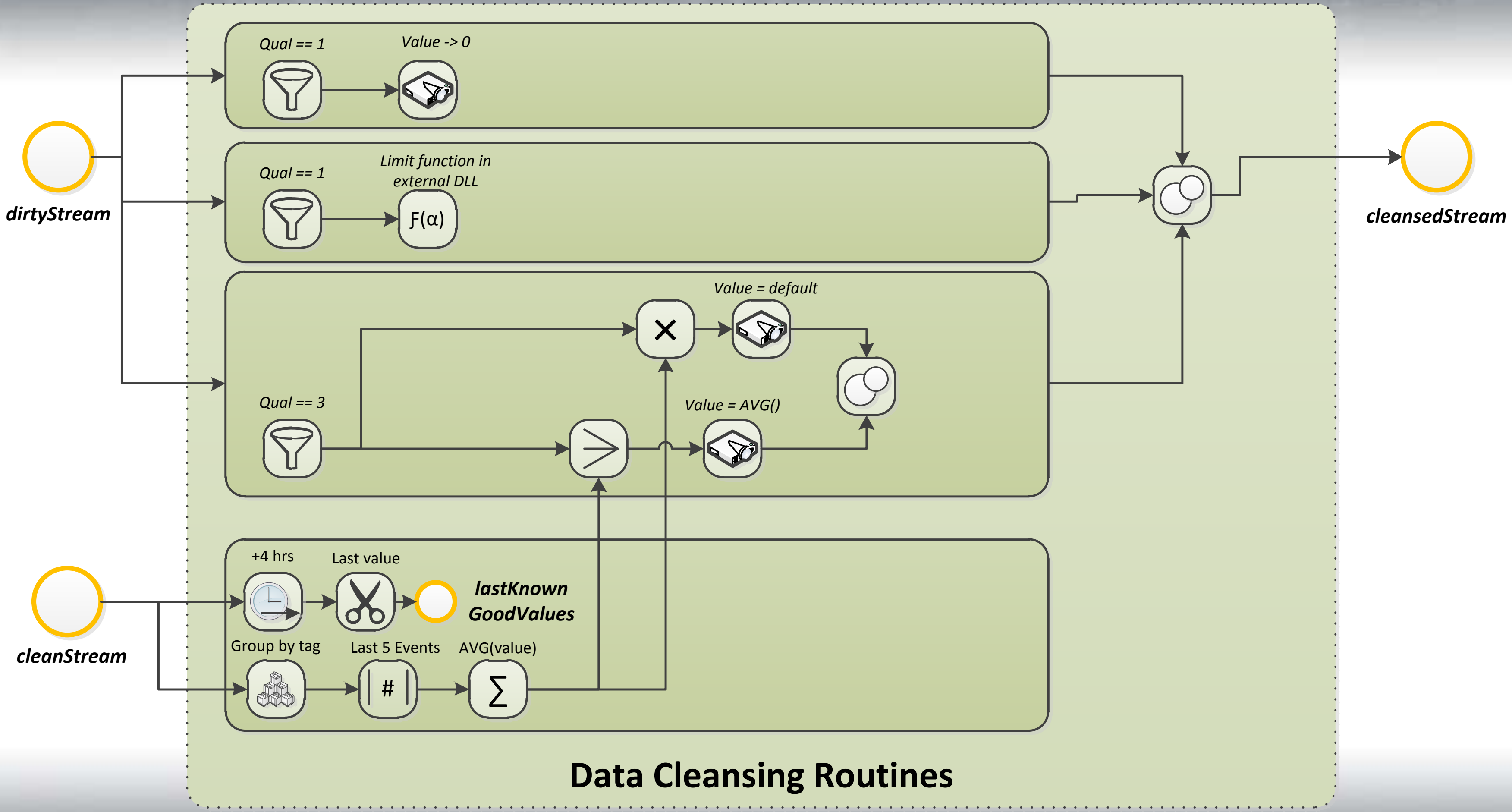
Data Quality



The *dirtyStream* is passed to a set of data cleansing algorithms (described in later slide)

©2010 OSIsoft, LLC. All Rights Reserved

Data Cleansing Routines



©2010 OSIsoft, LLC. All Rights Reserved

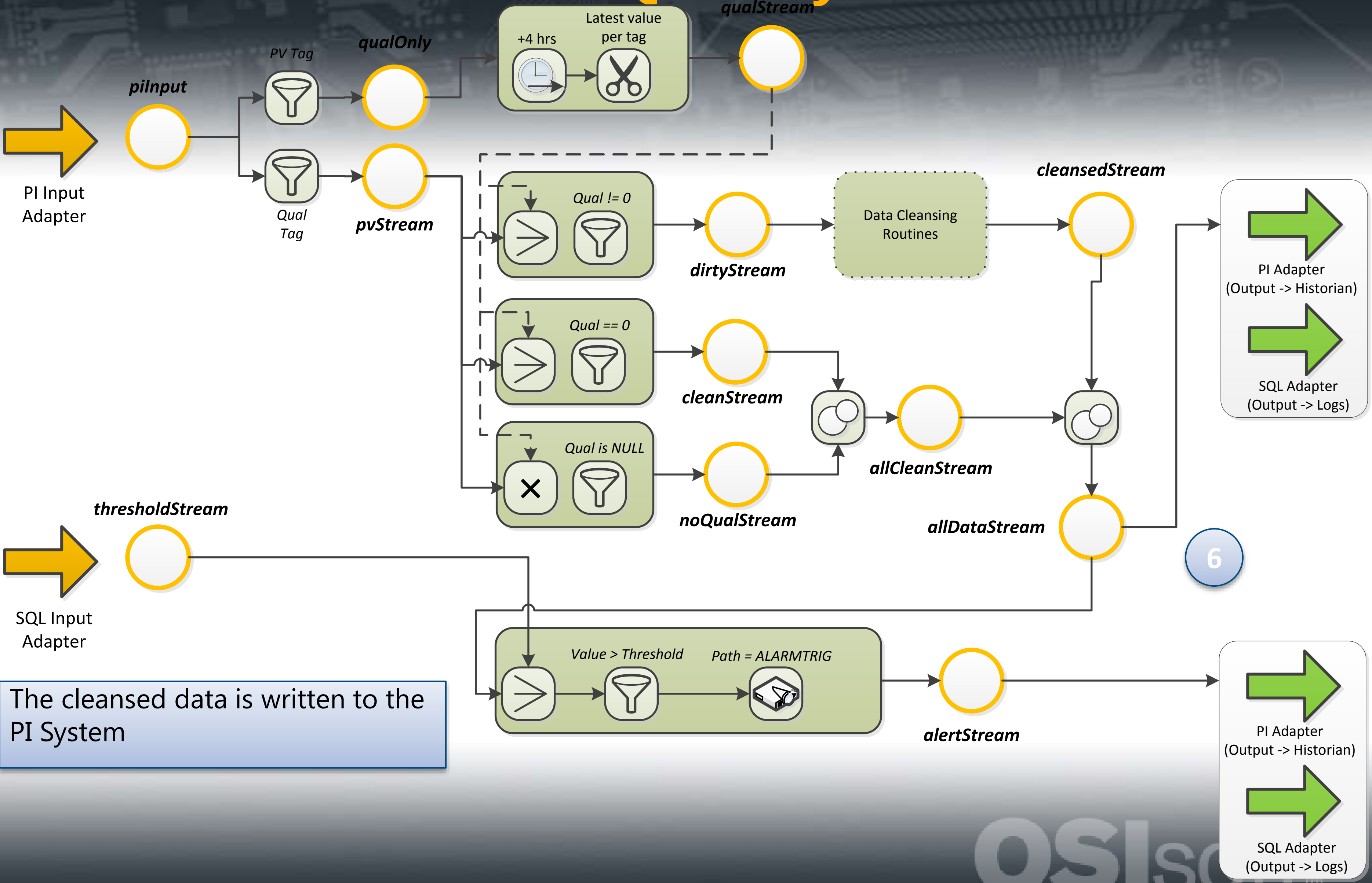
Data Quality - Step 5

```
// Get the cleansed stream (run the dirty stream through data
// cleansing routines)
// cleanStream used if required to obtain Last Good value.


var cleansedStream = GetCleansedStreams(dirtyStream, cleanStream);
```

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality



The cleansed data is written to the PI System

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality – Step 6

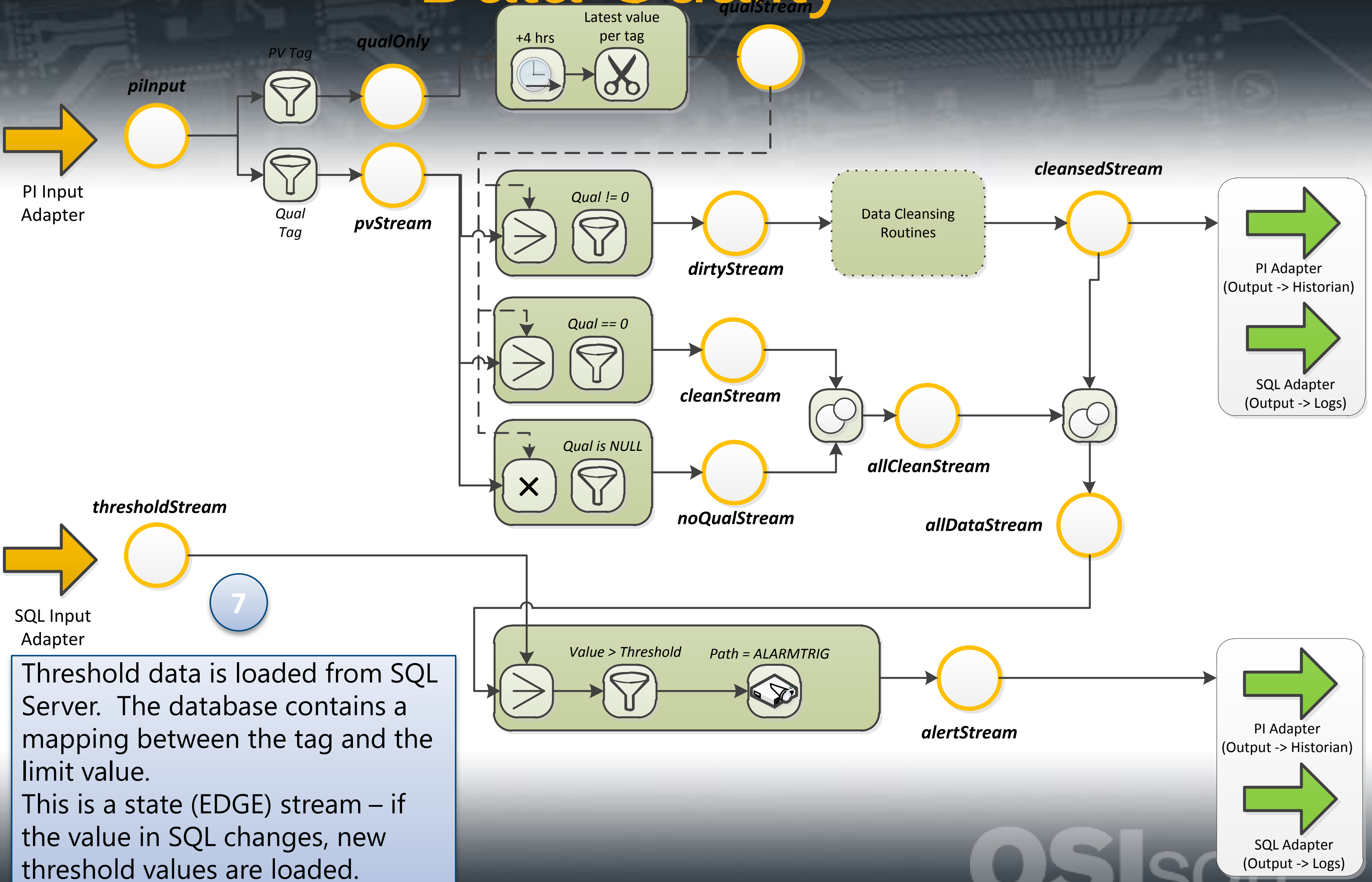
```
   // Combine the cleansed data with the clean data to create the
   // combined data stream.  We monitor the combined stream for
   // limit exceedence.
   var allDataStream = cleansedStream.Union(allCleanStream);




   // Output all data (clean and cleansed + alerts) to PI
   allDataStream.ToQuery(cepApplication, "AllDataToPi", "",
typeof(PIOutputAdapterFactory),
configStore.GetConfigurationObject<PIOutputAdapterConfig>("dataQualityOutput"),
EventShape.Point, StreamEventOrder.FullyOrdered),
```

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality



Threshold data is loaded from SQL Server. The database contains a mapping between the tag and the limit value.
This is a state (EDGE) stream – if the value in SQL changes, new threshold values are loaded.

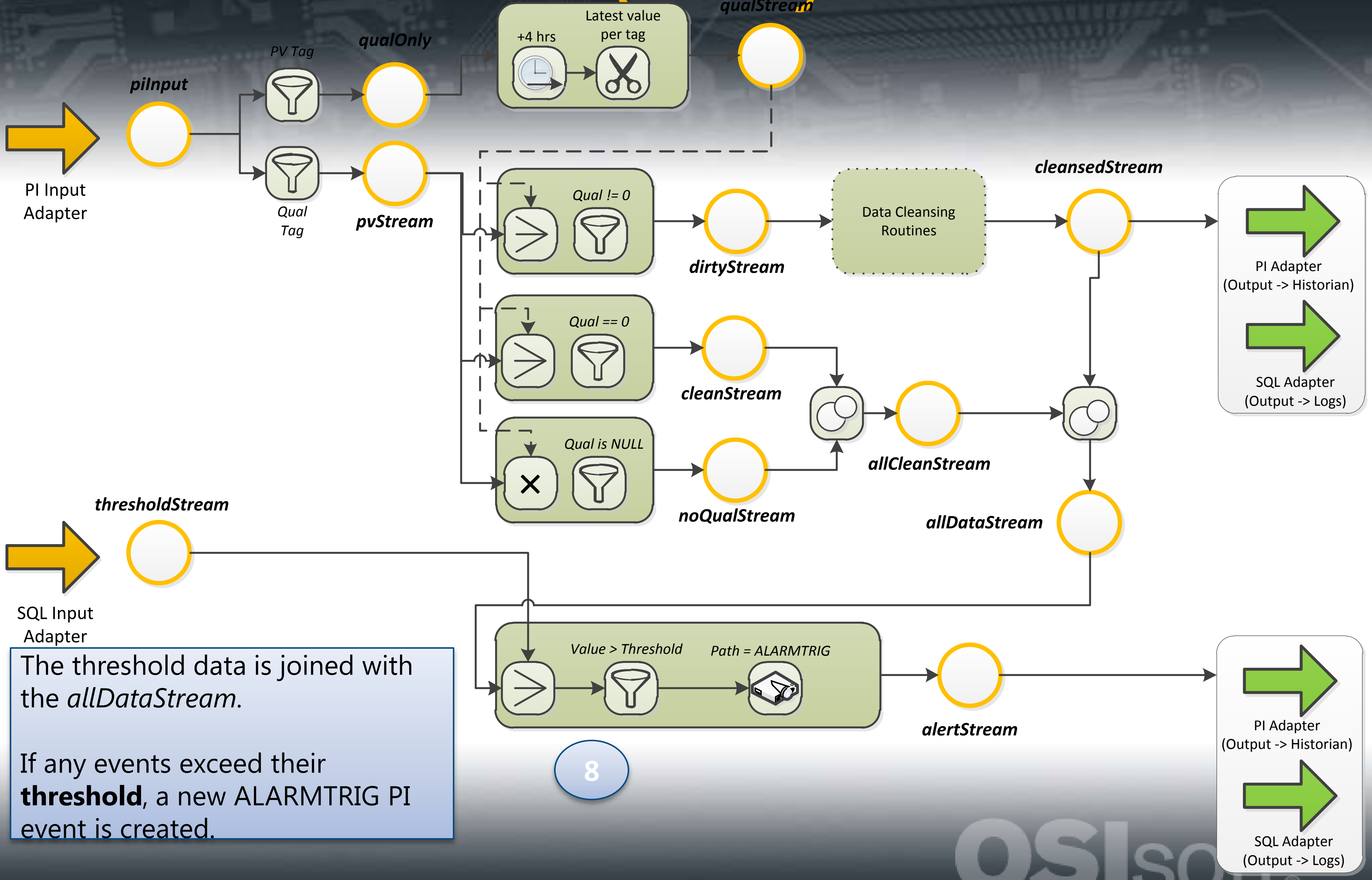
Data Quality – Step 7

```
     // Obtain Adapter configuration
var thresholdsSqlInputConfig =
configStore.GetConfigurationObject<SqlInputConfig>("ThresholdDataSqlInput");

     // Define the threshold input stream from the SQL Input Adapter
     var thresholdsStream =
CepStream<Thresholds>.Create("Thresholds Stream",
typeof(SqlInputAdapterFactory), thresholdsSqlInputConfig, EventShape.Edge, ats);
```

©2010 OSIsoft, LLC. All Rights Reserved

Data Quality



The threshold data is joined with the *allDataStream.*

If any events exceed their **threshold**, a new ALARMTRIG PI event is created.

©2010 OSIsoft, LLC. All Rights Reserved

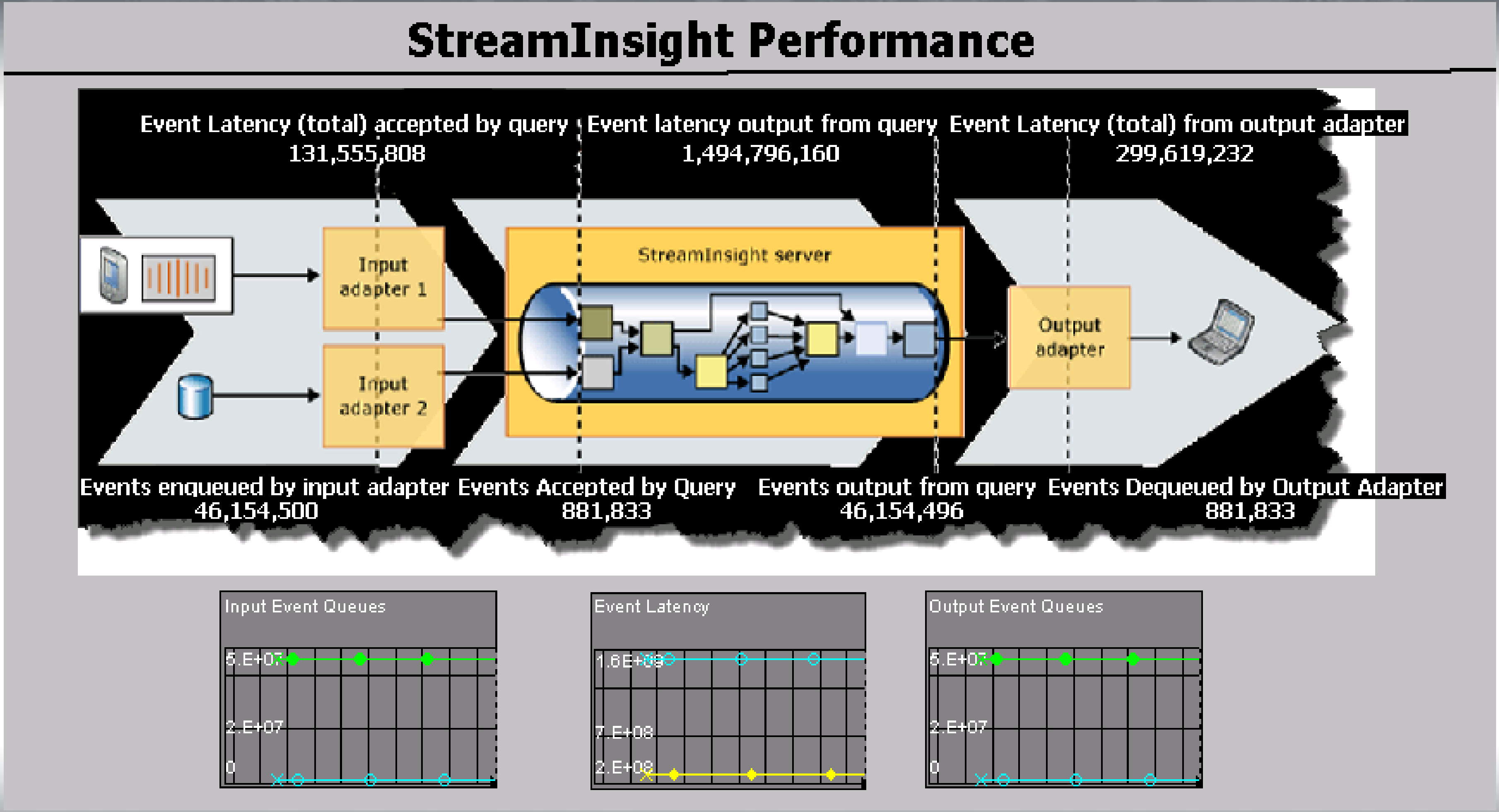
Data Quality - Step 8

```
// Join the alldataStream with the threshold reference stream
// to create the alerting stream
var alertStream = from e in allDataStream
    join th in thresholdsStream
    on e.Path equals th.thresholdName
        where e.Value > th.value
    select new PIEvent<Double>
    {
        Annotation = e.Annotation,
        Id = e.Id,
        IsAnnotated = e.IsAnnotated,
        IsEdited = e.IsEdited,
        IsQuestionable = e.IsQuestionable,
        Path = e.Path.Replace(".PV", ".ALARMTRIG"),
        Status = e.Status,
        StatusText = e.StatusText,
        Timestamp = e.Timestamp,
        Value = e.Value

    };
```

©2010 OSIsoft, LLC. All Rights Reserved

Using PI ProcesBook Displays for Performance Reports



StreamInsight Performance

Event Latency (total) accepted by query
131,555,808

Event latency output from query
1,494,796,160

Event Latency (total) from output adapter
299,619,232

Events enqueued by input adapter
46,154,500

Events Accepted by Query
881,833

Events output from query
46,154,496

Events Dequeued by Output Adapter
881,833

Observations

- Microsoft StreamInsight exceeded all performance and scalability expectations
- PI for StreamInsight CTP (pre-beta) handled significantly high data rates

©2010 OSIsoft, LLC. All Rights Reserved

More Information

- PI for StreamInsight
  - vCampus
    - Forums
    - Community Technology Preview (CTP)
- Microsoft StreamInsight
  - “Hitchhikers guide to StreamInsight”
  - Channel9 learning labs
    - http://channel9.msdn.com/learn/courses/SQL2008R2TrainingKit/SQL10R2UPD05/

©2010 OSIsoft, LLC. All Rights Reserved

Contact Information

- Email: Greg.Douglas@Logica.com
- Blog: http://gregorydouglas.wordpress.com/
- Twitter **@gwdouglas**

©2010 OSIsoft, LLC. All Rights Reserved

Thank You!

OSIsoft®

©2010 OSIsoft, LLC. All Rights Reserved

StreamInsight Query Concepts

○	Stream	A **stream** of events. Streams are **produced** by input adapters, or the result of an operation on top of another stream. Streams can be converted into **queries** by attaching an output adapter.	Ƒ(α)	External code	Callout to an external code block.
➡	Input Adapter	A source of streaming events.	▦	Group by	Groups a stream into a set of substreams .
➡	Output Adapter	A sink (or destination) for streaming events.	🕒	Alter Lifetime	Changes the lifetime and/or duration (start & end times) of an event. Can convert point events into interval events.
▭	Composite statement	A set of StreamInsight operators combined to perform some operation on a stream.	✂	Clip	Clips off the end time of events in a stream based on the start times of events in another stream. Typically used to convert a series of point events into a signal (i.e. keep the last known value in a stream)
⏷	Filter	Removes a set of events from a stream based on a filter condition (i.e. a WHERE clause)	📽	Project	Select fields from an event type in a stream into a new event (analogous to SELECT col1, col2)
>	Join	Joins two streams of events based on a relational key and a temporal overlap	|↓↑|	Hopping Window	Calculate results based on a set of events in a window.
×	Left Anti Semi Join (OUTER JOIN)	Joins two streams based on a on a relational key and a temporal overlap	|↶|	Tumbling Window	Calculate results based on a set of events in a window.
⚭	Union	Combines two streams (the two input streams flow into a single output stream)	📷	Snapshot Window	Calculate results based on a set of events in a window

© Logica 2010. All rights reserved

©2010 OSIsoft, LLC. All Rights Reserved