2002 OSIsoft Users Conference

Outpost

Who is Brock Solutions?

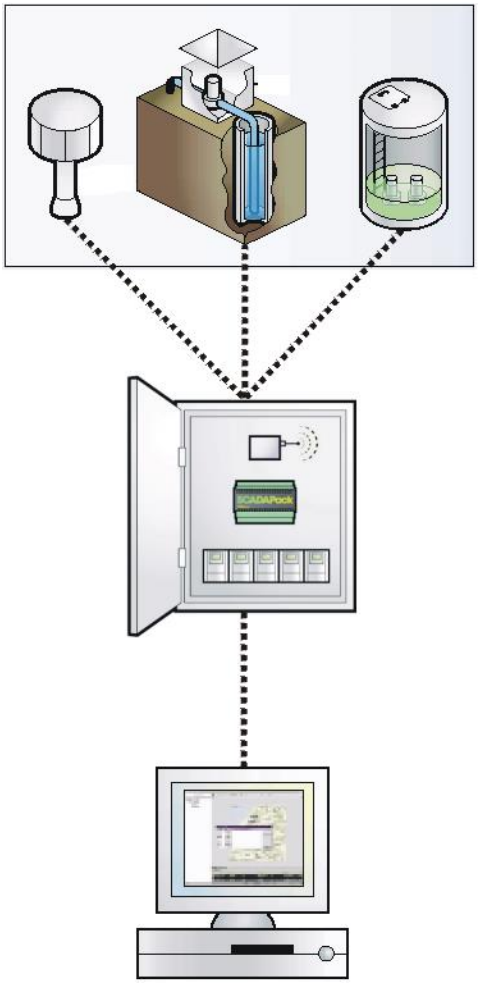
- North American Automation Engineering firm with offices in:
  - Canada: Ontario, Quebec
  - USA: Michigan, Missouri, New Jersey, Kentucky
- Experience in numerous industries including:
  - Potable/Waste water
  - Steel, Paper, Power Utilities
  - Logistics
  - Pharmaceuticals
- Privately owned and operated
- Over 220 employees

What is Outpost?

- Real-Time monitoring of Potable and Waste Water Systems
  - Archiving, Retrieval and Analysis of process data
  - High sample rates from remote equipment
  - Exception reporting directly to operation staff
  - Remote control of process equipment
  - Designed for operations staff to use and maintain
  - Regulatory Reporting
- Can be easily tailored to other industries that require Real-Time monitoring
- Developed in partnership with OCWA (Ontario Clean Water Agency)

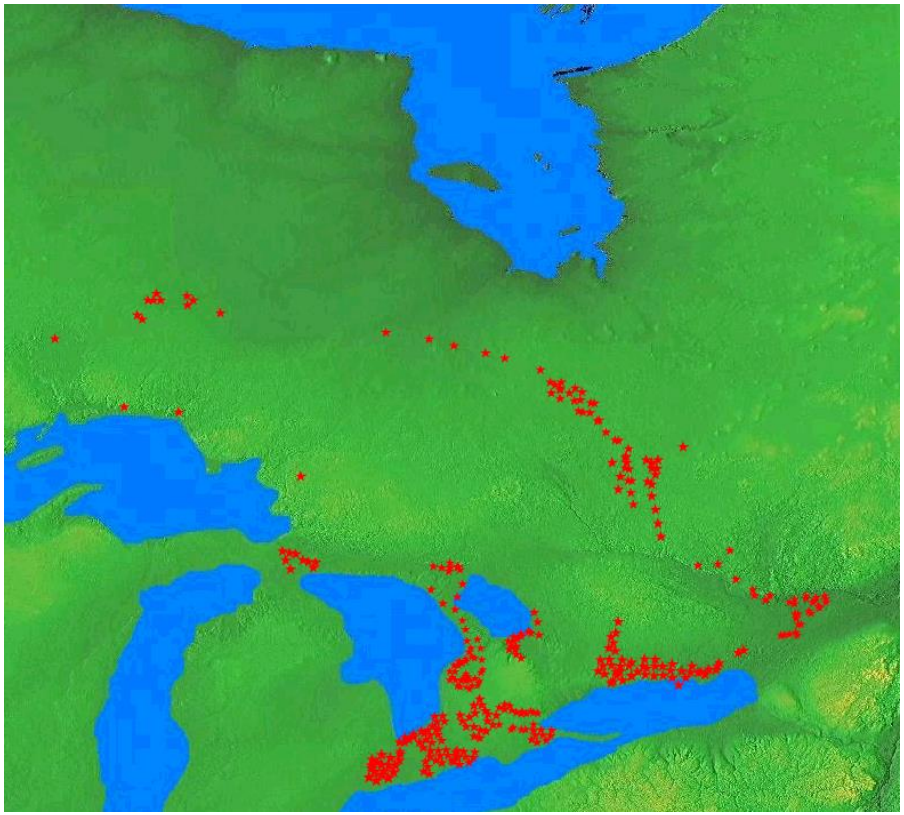
Outpost

Outpost Installations



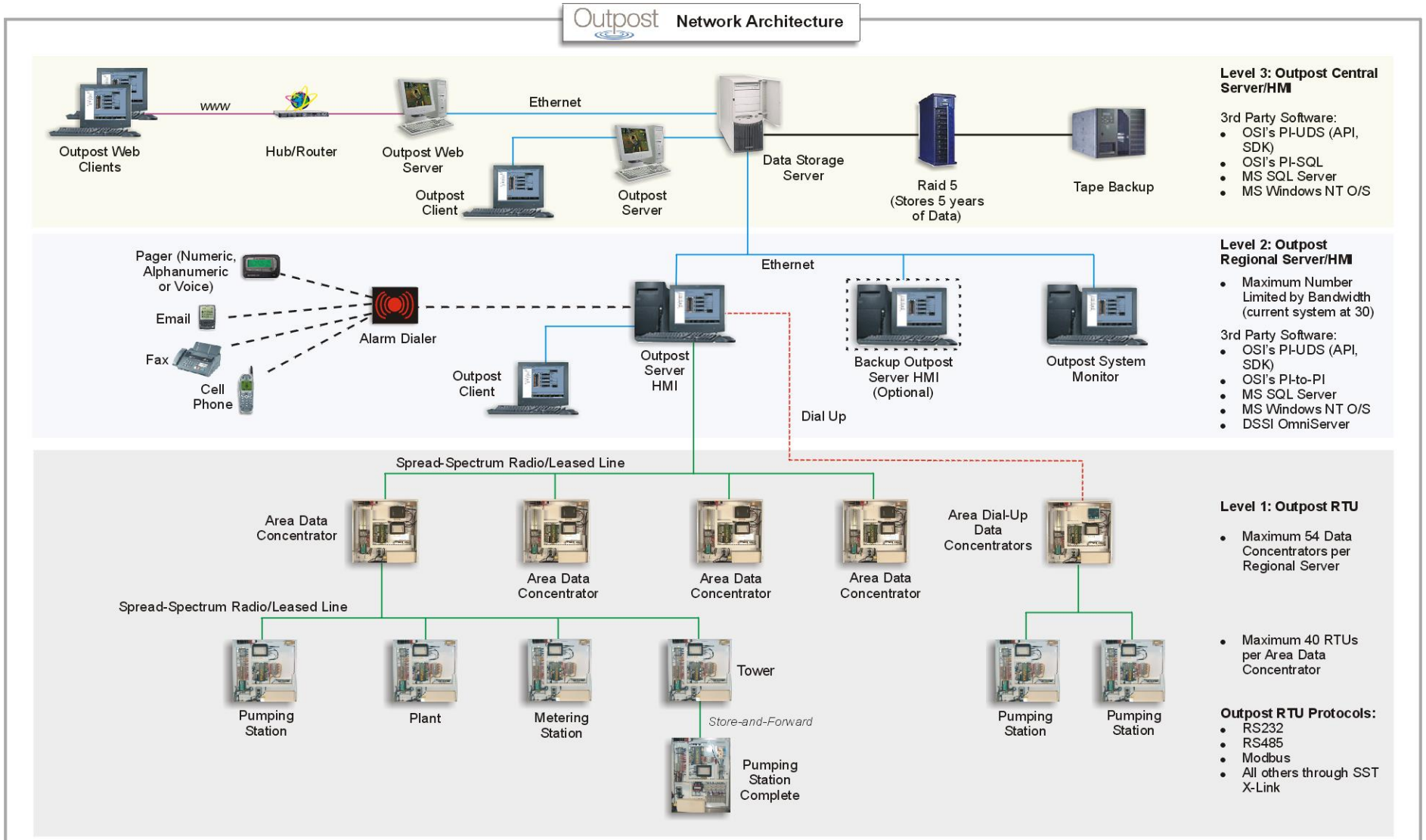
- 16 Outpost Servers across Ontario
- Central data collection point in Toronto, Ontario, Canada, collects process data from all locations
- Monitoring approximately 700 individual facilities and sites
- PI-UDS manages approximately 75,000-100,000 data points at the central data collection point

Outpost Installations - Ontario

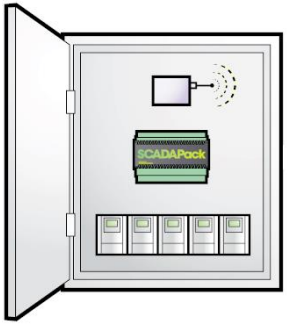


- Monitoring operations in 12 municipalities
  - 2 municipalities in Southern Ontario
  - 3 municipalities in Central Ontario including Walkerton
  - 4 municipalities in Eastern Ontario
  - 3 municipalities in Northern Ontario

Outpost Network Architecture - Overview



Outpost Network Architecture - Details

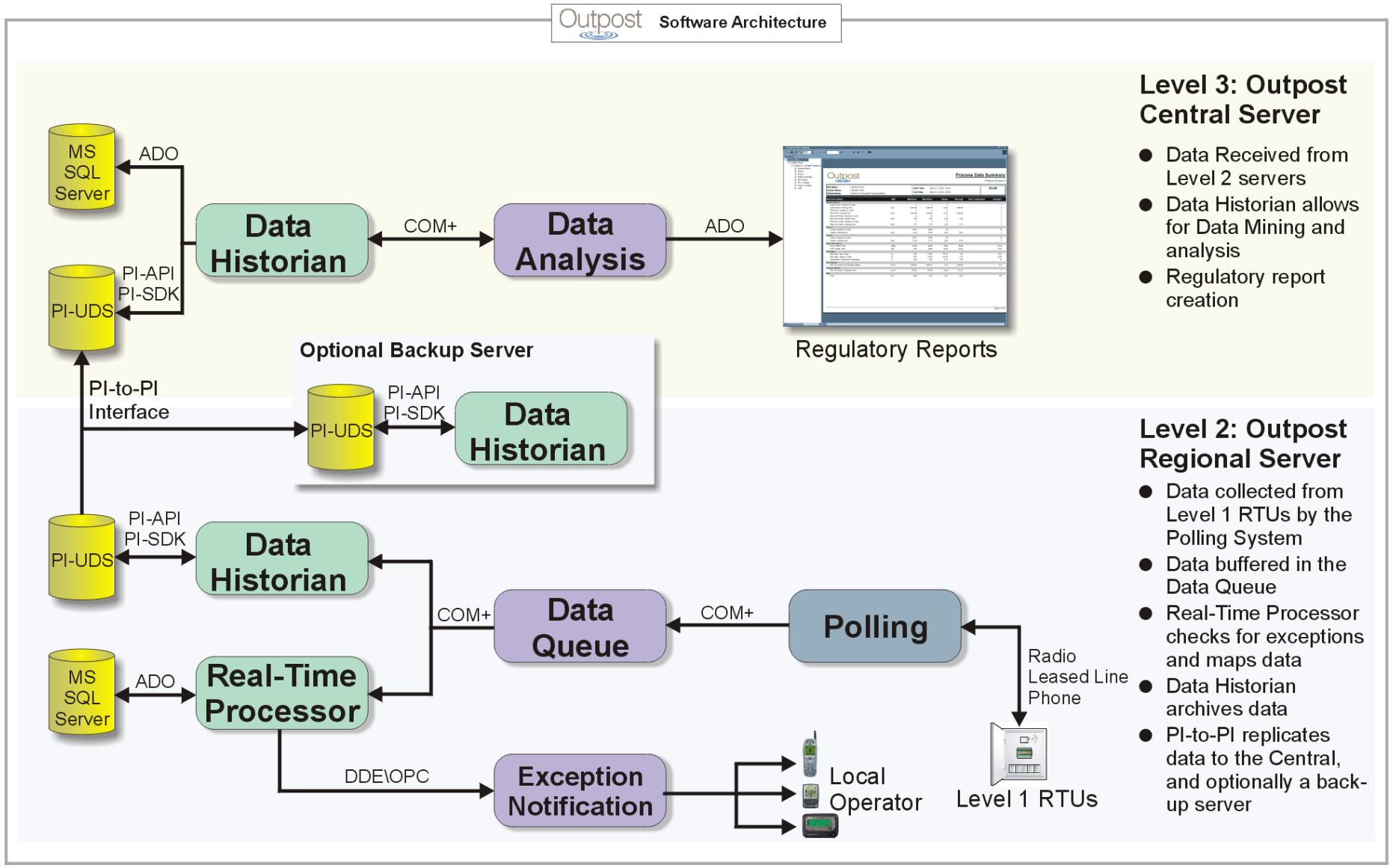


- Level 1: Outpost RTU
  - RTU level (Control Microsystems SCADAPack) with Communications via Radio, Phone Line, or Leased Line
  - Wired connections to I/O at each site
  - X-Link used as a protocol bridge to existing PLCs

- Level 2: Regional Server
  - Data is collected from Outpost RTUs and displayed on a local HMI based on calibrations in MS SQL Server
  - Data is checked for exceptions and archived in the PI-UDS
  - Data is replicated to a Central Server and optionally a backup Regional Server and using PI-to-PI

- Level 3: Central Server
  - Centralized Data Archive
  - Data from all Level 2 systems is available for data analysis and regulatory reporting

Outpost Software Architecture - Overview



Outpost Software Architecture – Details

- Level 2: Regional Server
  - Controls the collection of data from RTUs
  - Installed at each Regional Server Location
  - Client software can connect to any server on the Outpost network provided a user has security clearance
  - Single install package for client or server instances

- Level 3: Central Server
  - Data Analysis components allow for data mining and data extraction
  - Regulatory reports are produced with the Data Analysis component and Crystal Reports

Why we selected Microsoft Visual Basic for Outpost

- Flexibility
  - Allows for easy access to most COM objects (Those with IDispatch)
  - Easy access to the Windows API
  - Can rapidly prototype new system components in hours as opposed to days or weeks
  - Easy to support the COM programming model
- Easily Understood
  - Simple context
  - Does not require in-depth knowledge of the application or programming environment to understand how modules work

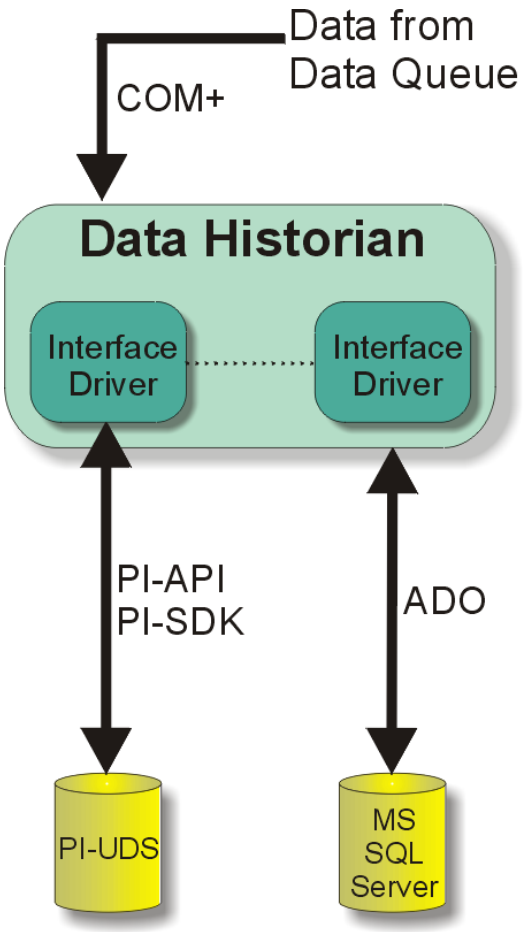
Why we selected the PI-UDS for Outpost

- Flexibility
  - Network Access, Remote Management
- Programming Interfaces
  - Easy low level access through the PI-API and PI-SDK
- Scalability
  - Can expand from a small tag count to a large tag count easily
- Availability of Replication
  - PI-to-PI from the Regional Servers to the Central Server
- Backup Procedures
  - Scripting already provided to do archive backups to an external source or tape backup
- "*Tried and Tested*" Package

Why we Use PI-to-PI

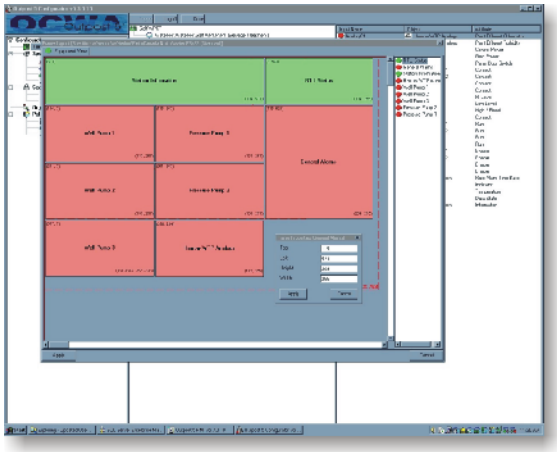
- The Central, and optional Regional backup server need to have all the data from each Regional Server
- Some latency between the data on the servers is acceptable
- Each Regional Server runs an instance of PI-to-PI for the central and an optional backup server
- Outpost's PI-to-PI Configuration
  - Historical Only mode ensures identical databases and lowers the processing requirements
  - Up to 12 hours of automatic recovery
  - Location1 is set to a numeric identifier for each Regional Server
  - Location2 is set to cause time stamps to be transferred from the source server

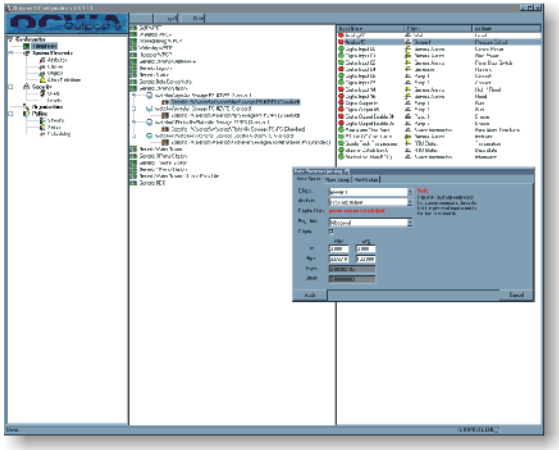
How Outpost Uses the PI-UDS



- The Data Historian component controls access to the PI-UDS
- Interface Drivers
  - Specially written for any type of Data Historian
  - COM (Component Object Model) Based
  - PI Interface Driver for the PI-API and PI-SDK
    - Stores, retrieves values
    - Manages PI tags
- PI tags are constructed from settings stored in MS SQL Server

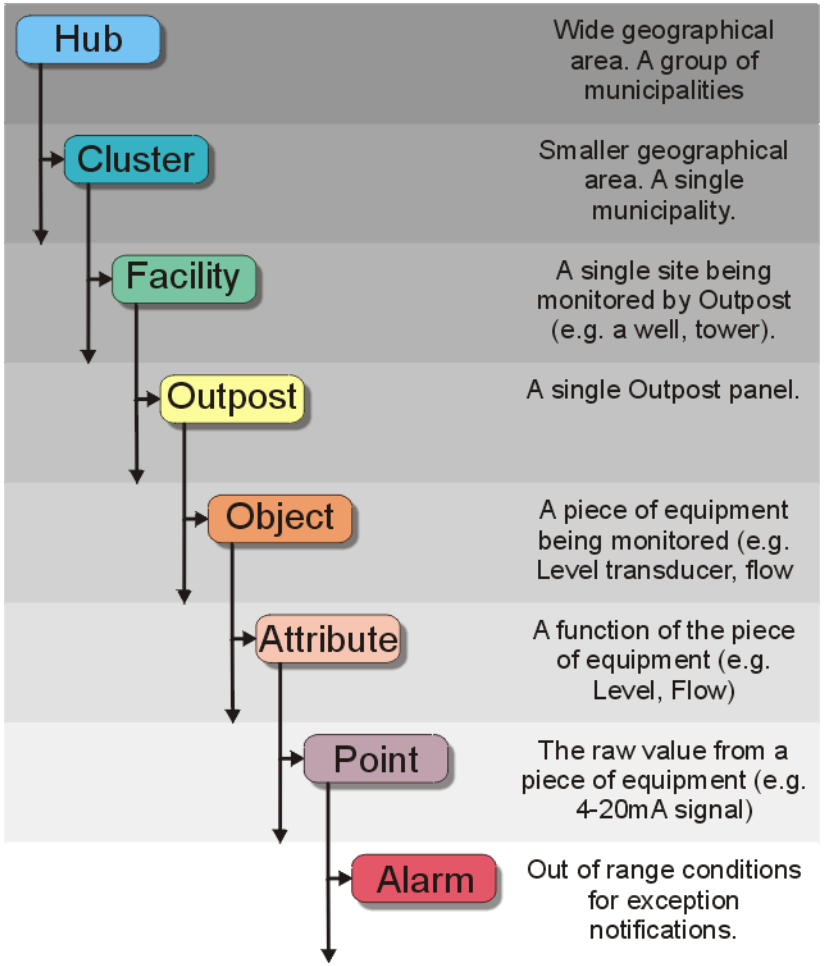
Configuring Outpost





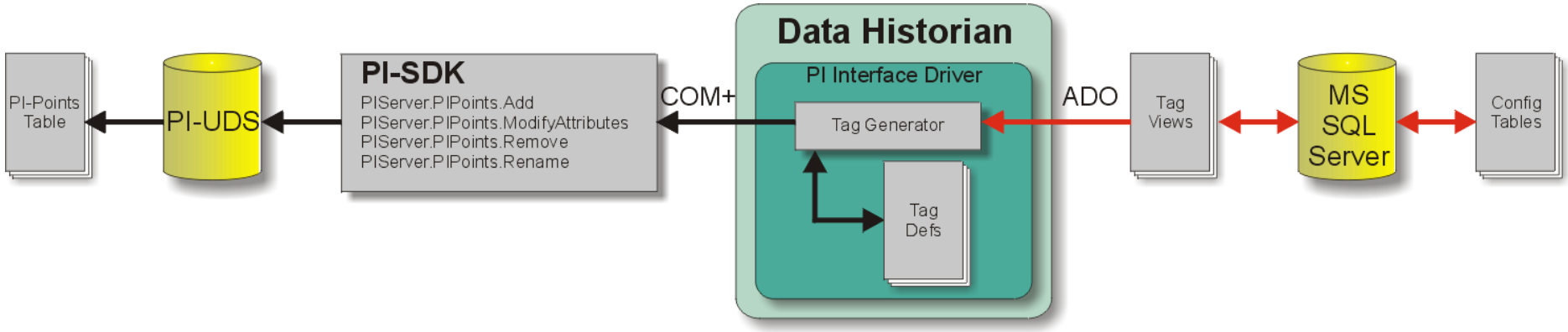
- All configuration information stored in an MS SQL Server database
- Configurations from MS SQL Server automatically converted into PI tags by the Data Historian component
- Site configurations separated into templates to reduce the number of duplicated entries
- Designed for operations staff to use

Outpost Organizational Structure



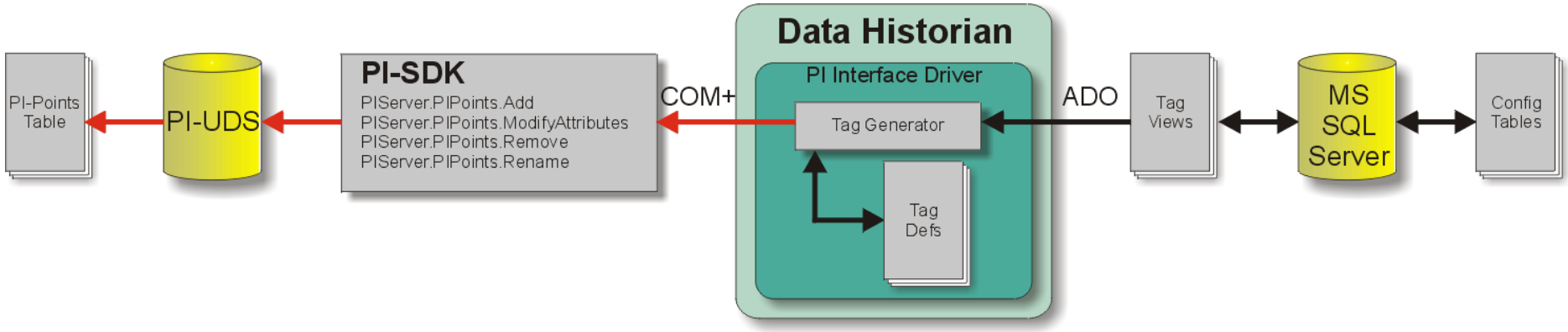
- Each level is given a short 8 character name
- PI tags are created by combining the short names from the Hub to Point levels
  - SMF.C_SMF.CP1.P1.WELL.LVL.RAI
- The Outpost Tree component is used to display the hierarchy in all client applications

PI-UDS Tag Generation – Step 1



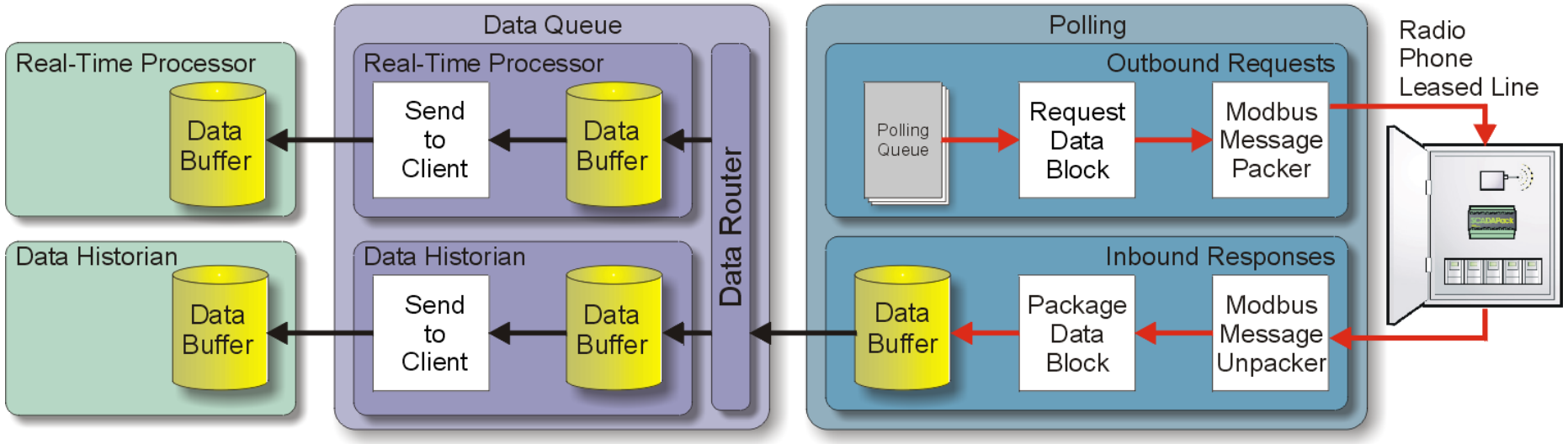
- The Data Historian component loads “Tag Generation Views” from MS SQL Server
  - Data entered using the Outpost Configurator
  - Calculate tag attributes like CompDev, Span, Zero
- Tag definitions are checked against an internal list of tags to see if they are new, changed, or removed

PI-UDS Tag Generation – Step 2



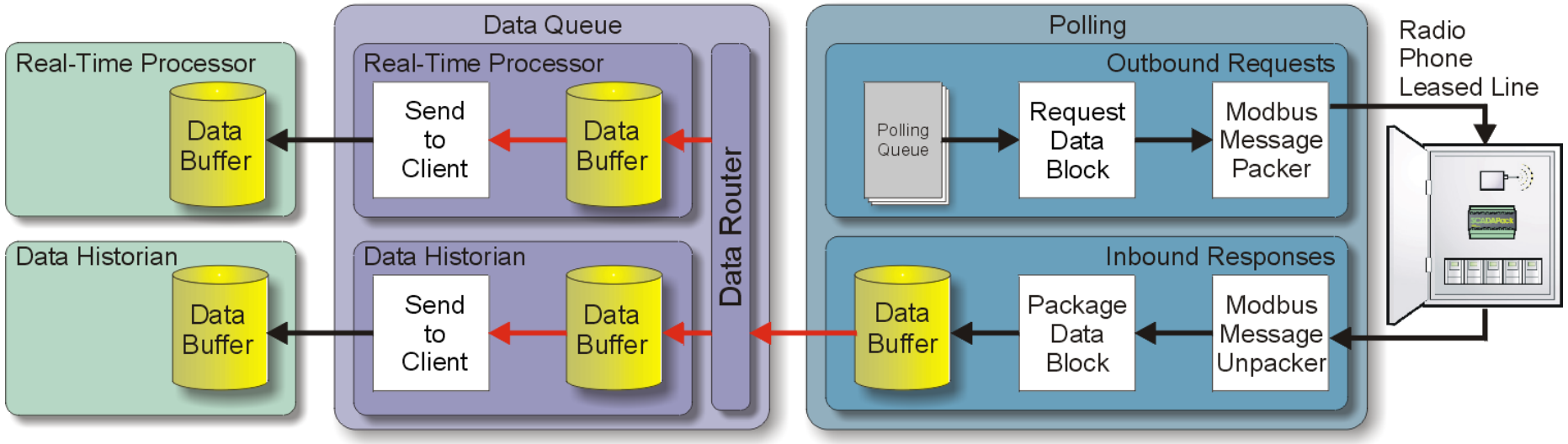
- The PI Interface driver of the Data Historian component makes calls to the PI-SDK for each tag
  - Depending on if the tag is new, changed or deleted, different PI-SDK calls are made
- The Data Historian component retrieves and records critical tag attributes like PointId, and RecNo
  - Possibility of data recovery on catastrophic failures

Data Collection – Step 1



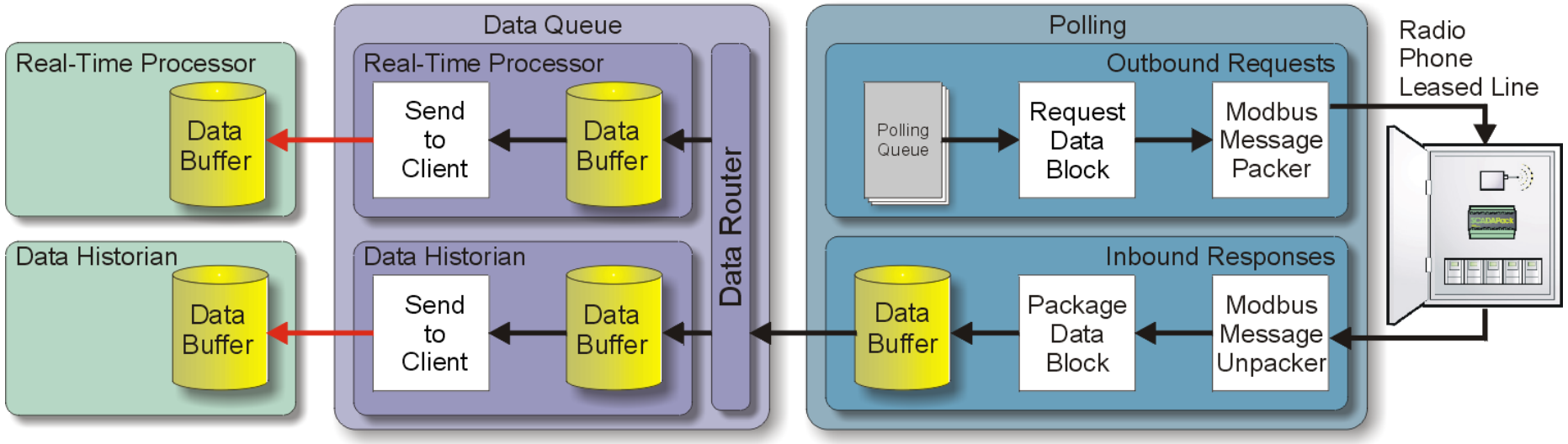
- The polling cycle is determined by the configuration in MS SQL Server
- Requests are sent and received via Modbus to the Level 1 RTUs
- All data that arrives is assigned a time stamp that is adjusted for latency on the RTUs

Data Collection – Step 2



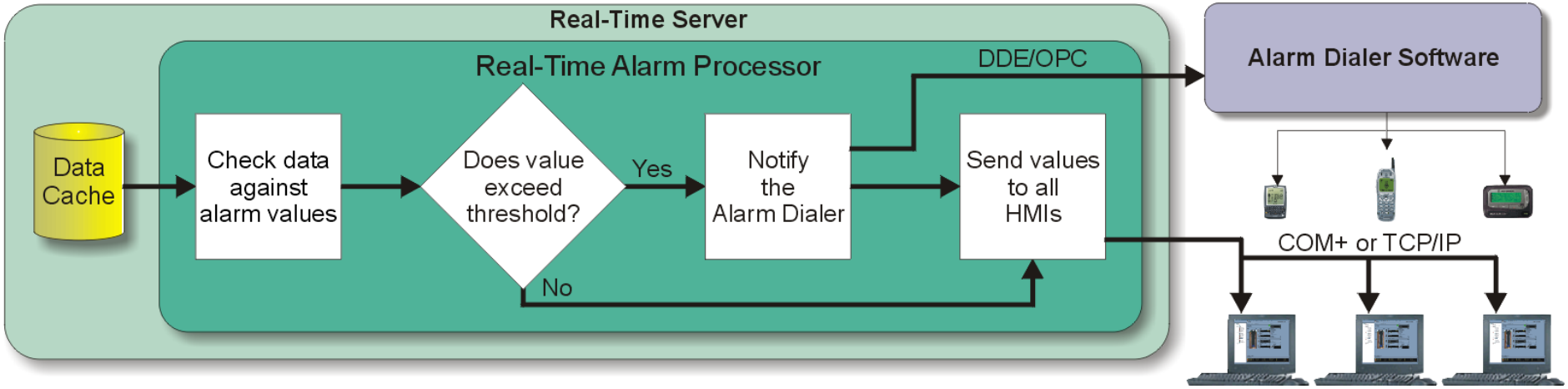
- The Data Queue buffers data for any Outpost server applications
- Queuing the data prevents data loss if a server process is busy or not running

Data Collection – Step 3



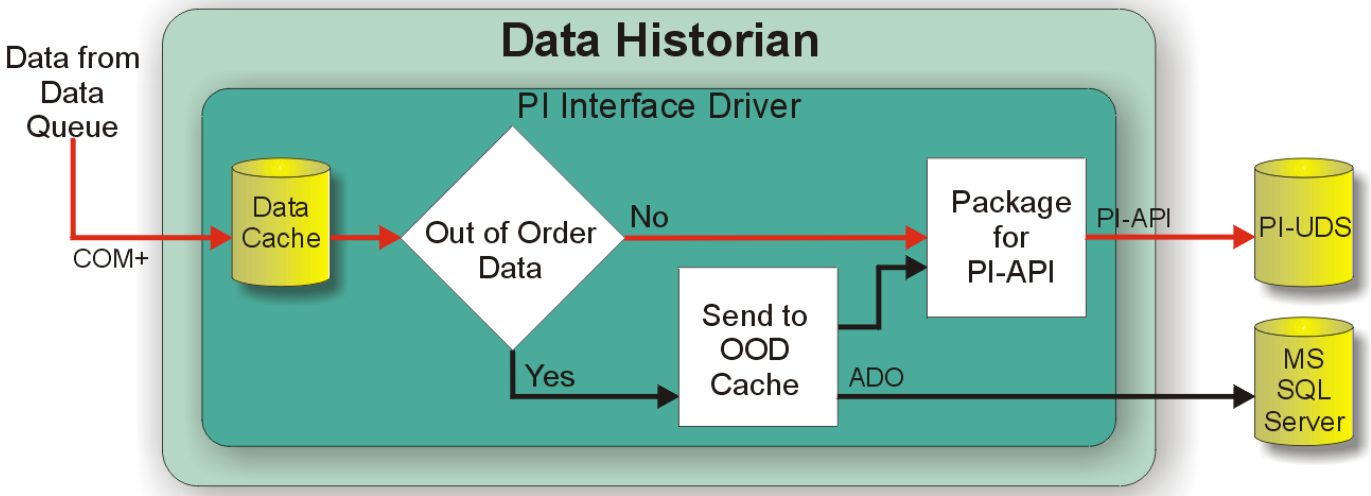
- Data is sent from a queue to an Outpost server application when it is ready to accept data
- Outpost servers are sent arrays of Identifier names, time stamps and values
- Each Outpost server maps the array of values into the organizational structure based on configuration in the MS SQL Server database

Exception Reporting and Alarming



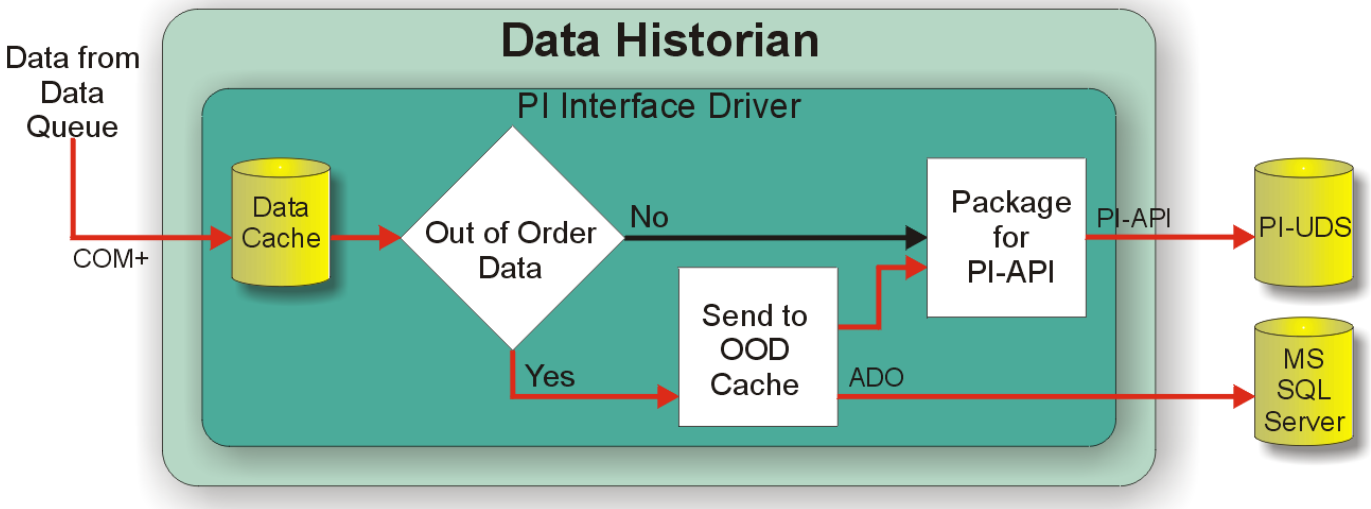
- Alarm conditions are trapped by the Real-Time Processor component as data arrives
- If data values exceed their threshold, they are sent to the alarm dialer software so that local operators can be notified
  - Thresholds are user definable by using the Outpost Configurator
- Once the thresholds are verified, the values are sent to any HMI
  - HMIs subscribe to certain segments of data and are notified when that data changes

Putting Data Into the PI-UDS – Real-Time Data



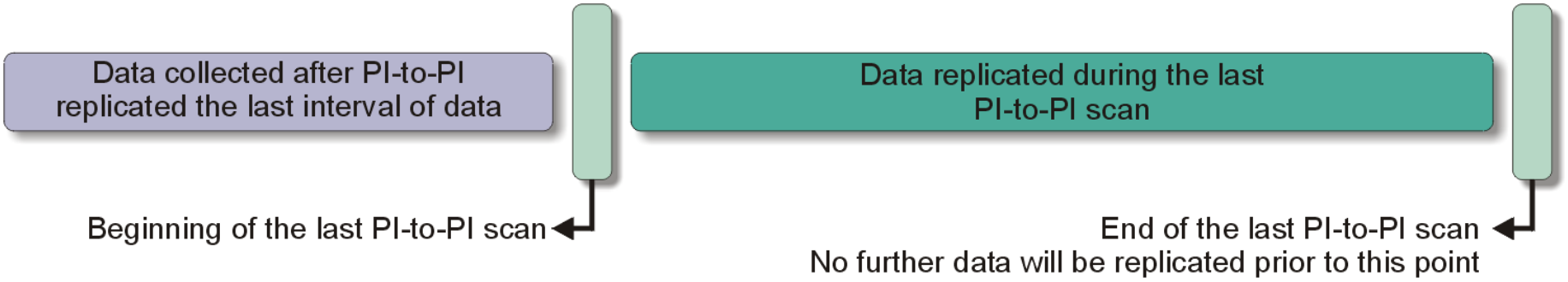
- Data time stamps are checked to see if the data is Out of Order or Real-Time
- Real-Time data is passed directly through to the packager
  - Data is converted from raw RTU values to scaled engineering values as configured in MS SQL Server
  - Data is packaged into arrays corresponding to the data type
  - Different data type arrays are sent to pisn_putsnapshotvaluesx one at a time (e.g. Float32s, Int32s)

Putting Data Into the PI-UDS – Out of Order Data



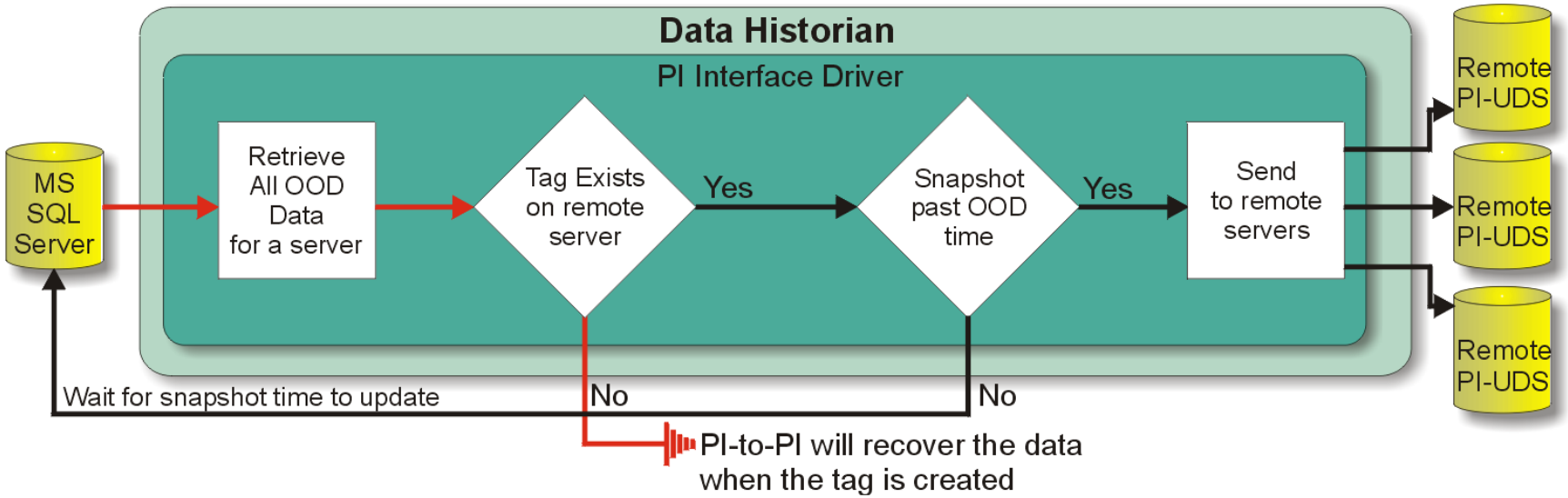
- When data arrives that is Out of Order, it must be handled specially
  - Step 1: The data is passed to the Out of Order data cache
  - Step 2: The data is passed to the packager to be sent to the local PI-UDS the same way Real-Time data would be

Out of Order Data Management



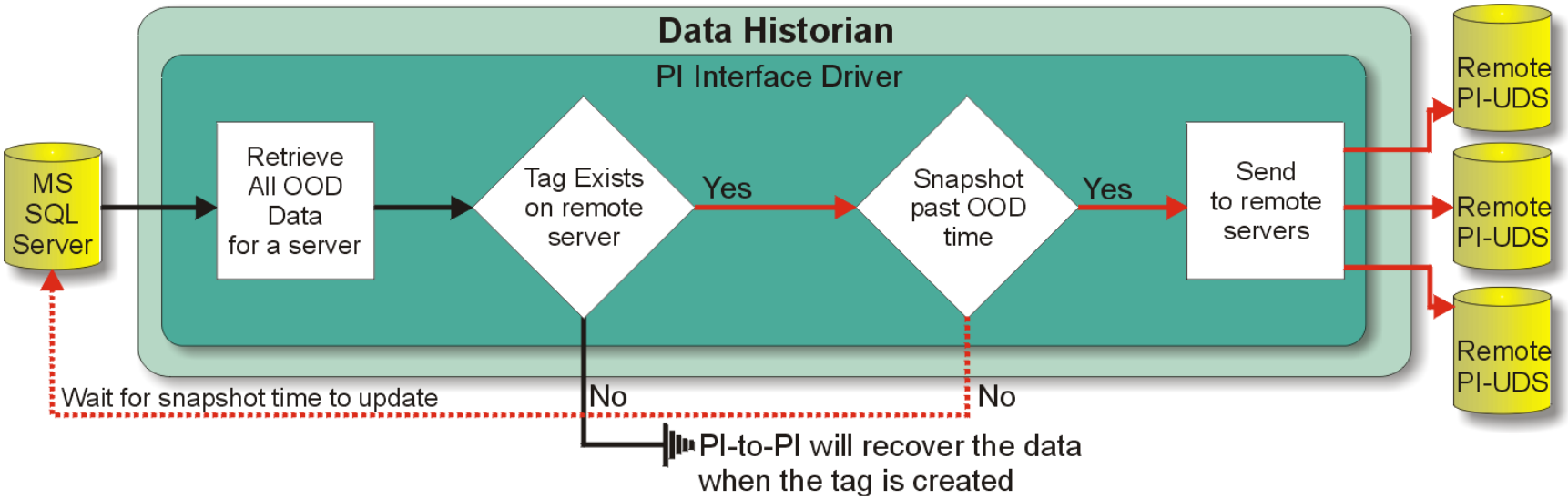
- On occasion, Outpost must collect data that is time stamped before the time stamp of the last PI-to-PI scan
- PI-to-PI will not pick up this data for replication when in history only mode
  - Can cause the appearance of missing data on remote servers
- Out of Order data that falls into this category must be identified and replicated manually
- *The Out of Order data handling routine that Outpost uses is temporary pending the release of the new version of PI-to-PI*

Replicating Out of Order Data – Step 1



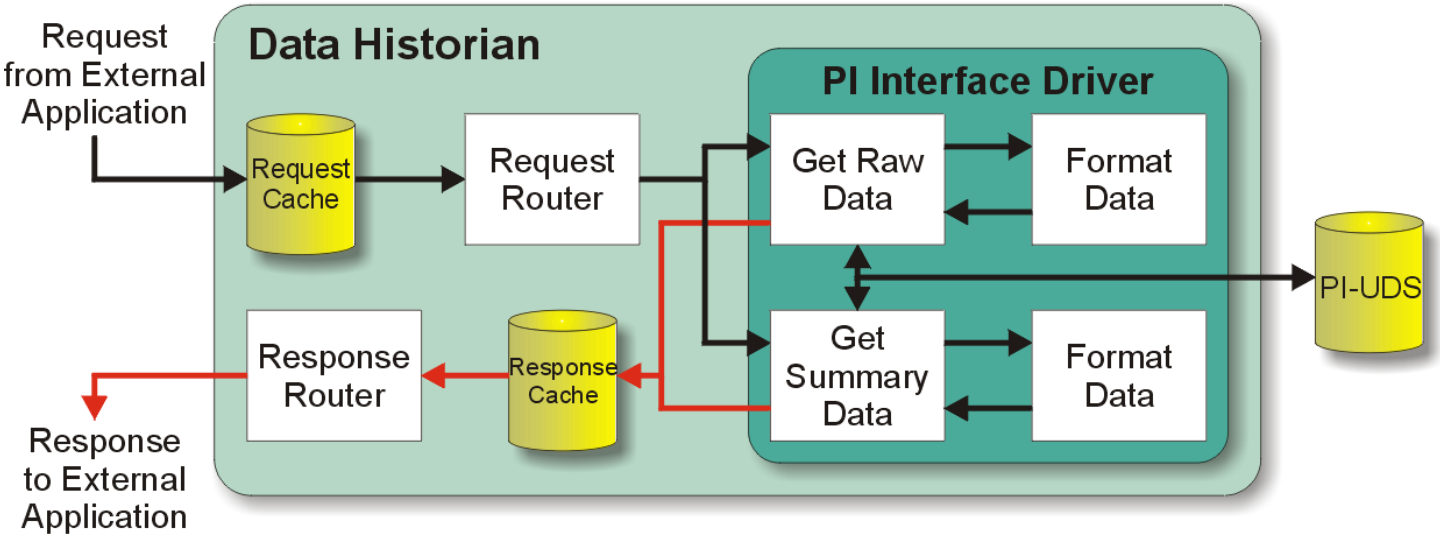
- On a given interval (default 10 minutes), the Out of Order data processor will load all data to be transferred
- Each tag is checked to see if it exists on the remote PI-UDS
  - If the tag does not exist, the stored values are discarded as PI-to-PI will fill in the data

Replicating Out of Order Data – Step 2



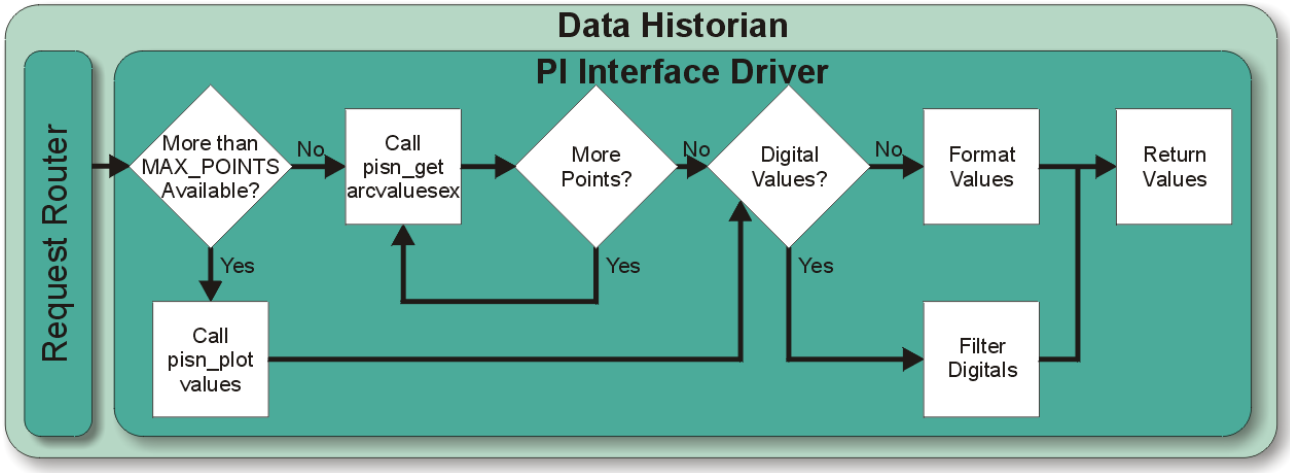
- Each time stamp is checked to see if it is past the snapshot time on the remote PI-UDS
  - This is required so that when the data is inserted it is treated as Out of Order data on the remote PI-UDS and not Real-Time data
- If the time stamp is past the snapshot time, the data is sent back to the cache to wait
- Otherwise, the data is sent to the remote PI-UDS

Getting Data out of the PI-UDS – Requests and Responses



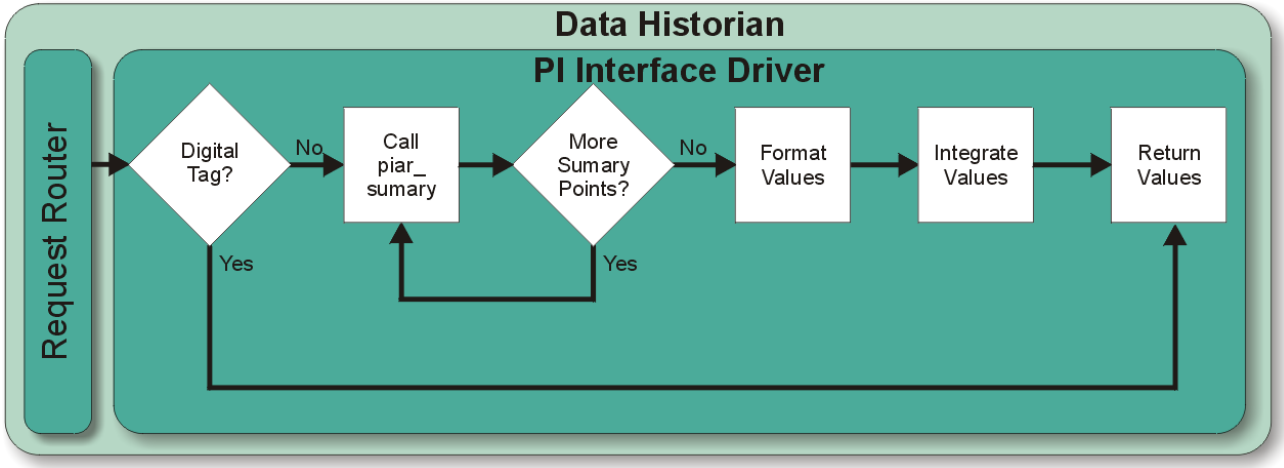
- Requests for data arrive from external applications
  - Either by COM or TCP/IP
  - Requests are for raw or summarized values
- Each request is followed by a response, even if it is as simple as an error code
- All requests and responses are processed asynchronously

Getting Data out of the PI-UDS – Raw Data



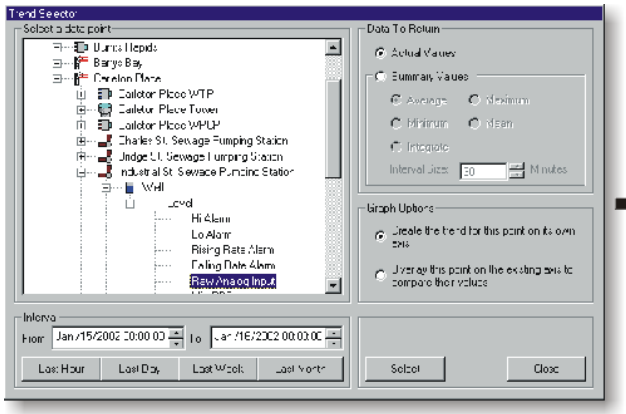
- Two ways to get raw data
  - If there are more then MAX_POINTS then use pisn_plotvalues
  - Otherwise, use pisn_getarcvaluesx
- Non-Digital values are formatted into the requested units
- Formatted arrays of values are passed back to the Response Cache

Getting Data out of the PI-UDS – Summarized Data

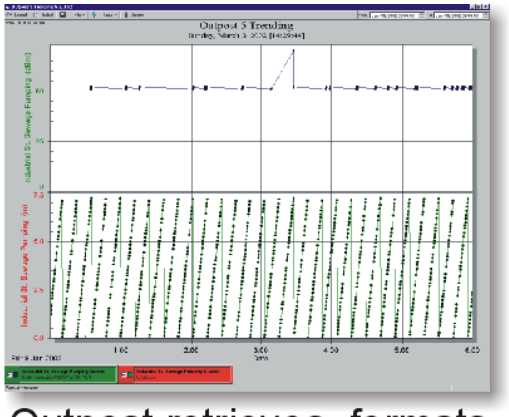


- Multiple calls to piar_summary are made for each summary code
- Values are formatted to the requested units
- Instantaneous Flow values are integrated over the time period
  - Integration = (ARCAVERAGE * SizeOfTimePeriod) cubic meters
  - Allows for approximate mass balancing of facilities
  - Reduces the need for expensive hardware flow totalizers

Viewing PI-UDS data with Outpost Trending



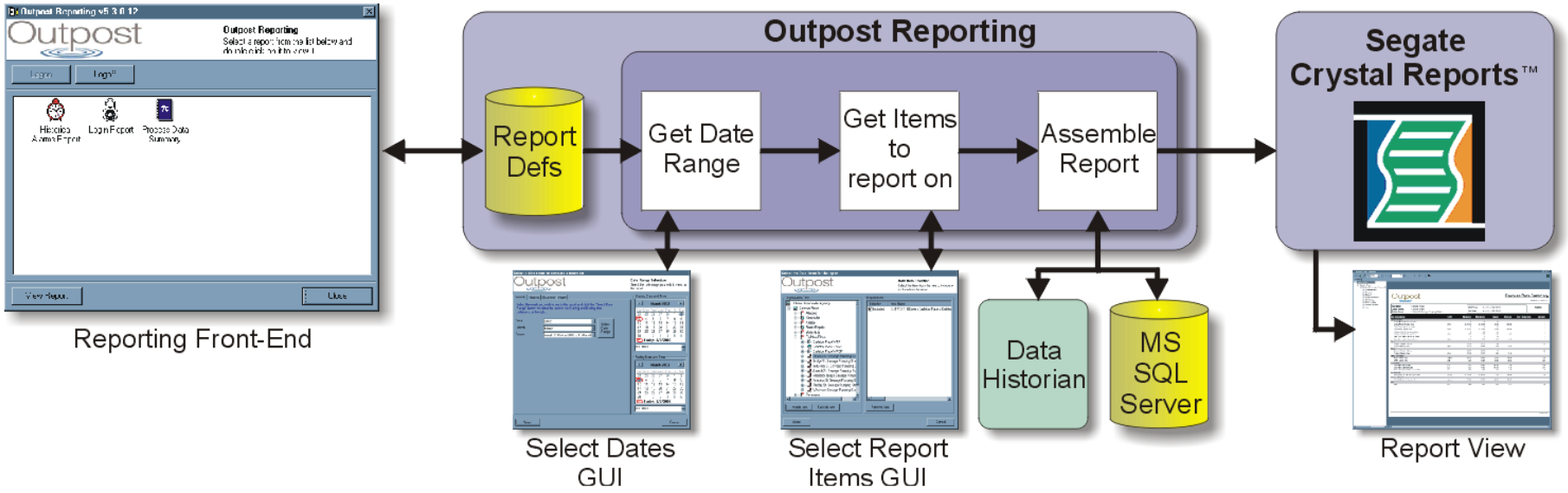
User selects the data and time range to trend



Outpost retrieves, formats and displays the data

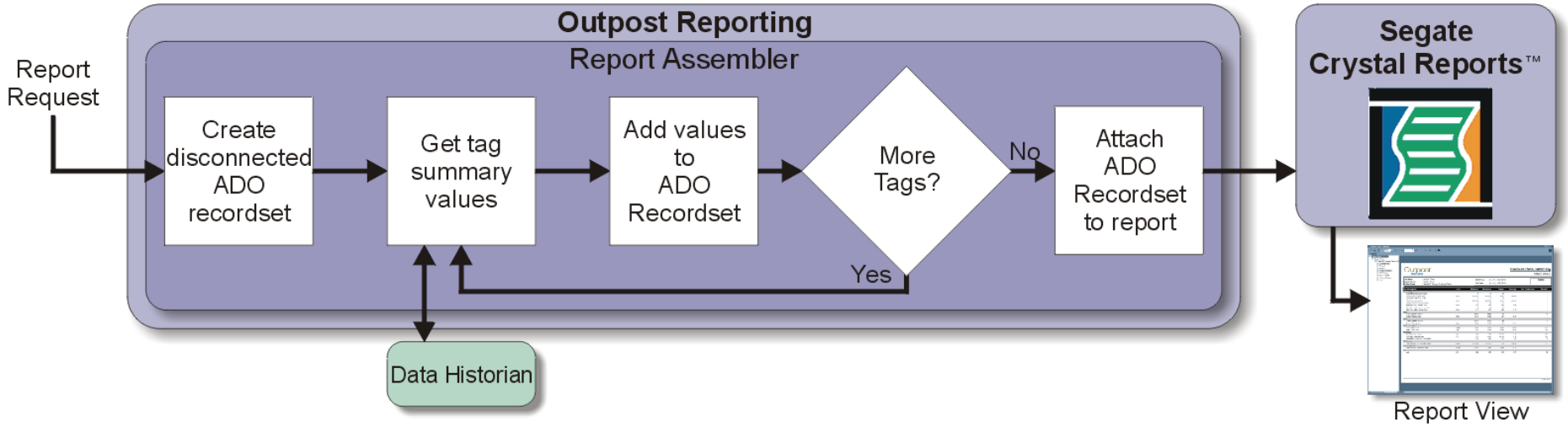
- Data requests sent to the Data Historian Module for requested tags
  - Sent through COM on local systems or TCP/IP on remote systems
- Utilizes the Outpost Tree component to select data
  - Outpost Tree component used in all client applications for navigation
  - Hides the complexities of knowing tag names
- Data can be exported to CSV files for further processing

Outpost Reporting



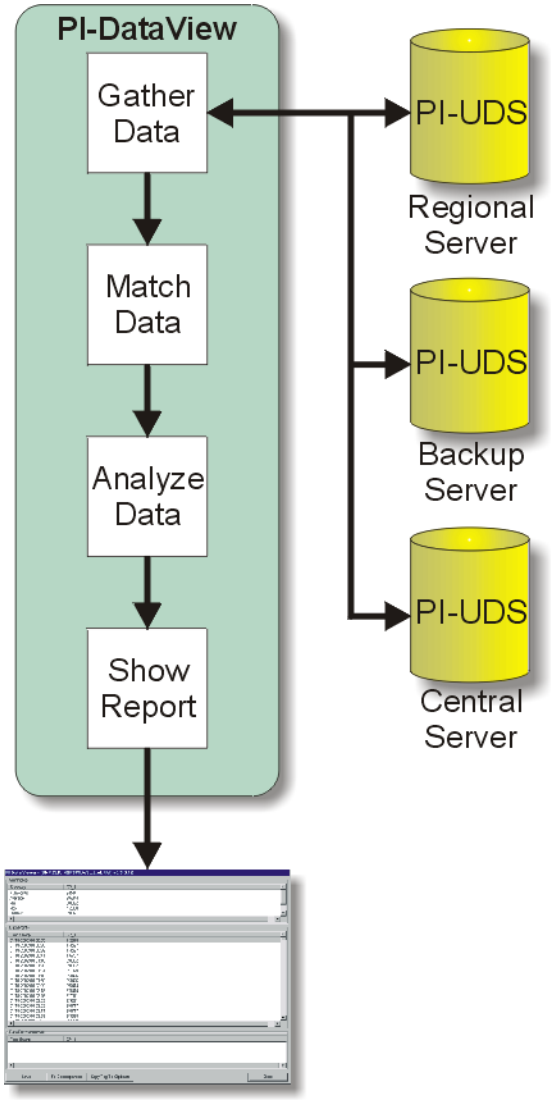
- Reports are created in Segate Crystal Reports™
- Definition files are created to tell Outpost how to format the data for Crystal Reports™
- Uses the Outpost Tree component to select data to view

Process Data Summary Reports



- A Disconnected ADO recordset is created to hold data
  - Allows for easy Crystal Reports™ integration with an ADO Field definition file
- Summary information is retrieved for each selected item from the Data Historian component and added to the ADO recordset
- The ADO recordset is attached to the report
- The report is sent to Crystal Reports™ for viewing

Data Consistency



- Due to government regulations and the number of distributed servers, Outpost needs to be accountable for the data it collects and its validity
- The PI-DataView tool allows for consistency checks
  - Data arrays from all servers are retrieved using piar_getarchvaluesx
  - Data arrays are then combined into a master array keyed on the time stamp
  - Holes are identified by searching out items in the master array with data values missing on remote servers or different values
  - A report is generated detailing the success of the routine and whether any holes were found

2002 OSIsoft Users Conference

Outpost