Promotion of AF Systems for Weekly Database Loads

Aaron Rosenthal - Operations Engineer, ERCOT

September 14, 2016

About ERCOT

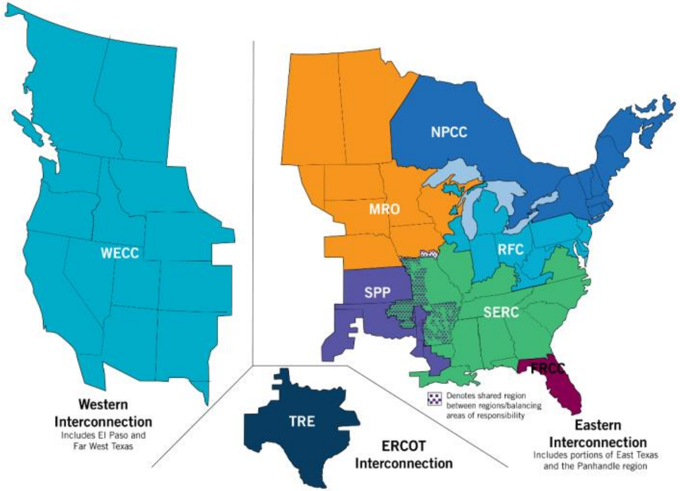
COLUMBUS
10
Gen: 36,112 mw
Load: 36,018 mw
Wind: 4,817 mw (13%)
PRC: 5,353 mw
Views
Search:
ERCOT
RLC Unit Input Data
2014-10-28 09:03:01 CDT
2014-10-28 09:10:09 CDT
Approved
COMMUNICATION STATUS
Load
Gen
24-hr Ld Delta
36042 MW
36137 MW
148 MW
PUNs
Total 2419
Steel 207
Interface Limits
ERCOT Wind
West
Panhandle
Frequency
Taylor: 59.965 HZ
Bastrop: 59.984 HZ
Regulation
Non-Spin
Quick Start

Introduction – ERCOT

- Electric Reliability Council of Texas, Inc.
  - ISO/RTO for the state of Texas
- Four primary responsibilities
  - System reliability – planning and operations
  - Wholesale market settlement for electricity production and delivery
  - Retail switching process for customer choice
  - Open access to transmission

ERCOT Quick Facts

- 90% of Texas load
  - 24 million customers
  - 75% of load is retail-choice
- 77,000+ MW expected generation
- Record peak load of 71,197 MW
  - August 11, 2016
- 16,000+ MW of installed wind capacity
  - Most of any state in the nation
- 288 MW of installed solar capacity


NPCC
MRO
RFC
WECC
SPP
SERC
FRCC
TRE
Western Interconnection
Includes El Paso and Far West Texas
ERCOT Interconnection
▩ Denotes shared region between regions/balancing areas of responsibility
Eastern Interconnection
Includes portions of East Texas and the Panhandle region

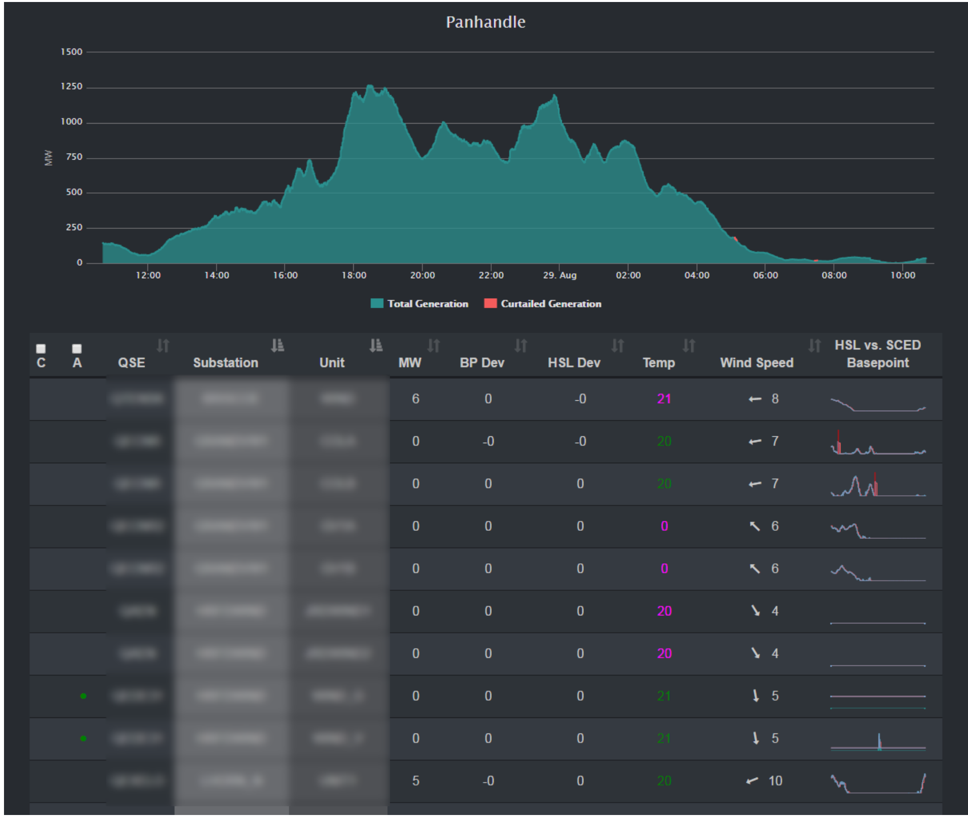
PI at ERCOT

- Started using PI in 2010
- 530,000+ PI tags
  - ~303,000 from SCADA
  - ~123,000 for Performance Equations
  - ~61,000 for AF analyses
- 2TB total used archive space
  - ~2GB daily archive size

Official Control Room PI Displays

Next-gen Displays

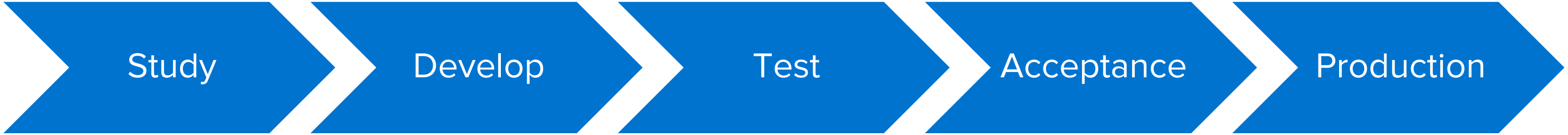
- Prototype for new renewables desk
- Tracking wind curtailment
- Uses web technologies
  - PI Web API
  - AngularJS
  - Highcharts



□ C	□ A	QSE	Substation	Unit	MW	BP Dev	HSL Dev	Temp	Wind Speed	HSL vs. SCED Basepoint
		[redacted]	[redacted]	[redacted]	6	0	-0	21	← 8	
		[redacted]	[redacted]	[redacted]	0	-0	-0	20	← 7	
		[redacted]	[redacted]	[redacted]	0	0	0	20	← 7	
		[redacted]	[redacted]	[redacted]	0	0	0	0	↖ 6	
		[redacted]	[redacted]	[redacted]	0	0	0	0	↖ 6	
		[redacted]	[redacted]	[redacted]	0	0	0	20	↘ 4	
		[redacted]	[redacted]	[redacted]	0	0	0	20	↘ 4	
	●	[redacted]	[redacted]	[redacted]	0	0	0	21	↓ 5	
	●	[redacted]	[redacted]	[redacted]	0	0	0	21	↓ 5	
		[redacted]	[redacted]	[redacted]	5	-0	0	20	↙ 10	

Building Asset Framework

Building Asset Framework



Requirements Gathering

Determine existing PI tag naming conventions for various asset types

Decide on key asset metadata for both analysis and display purposes

Plan element hierarchy structure for both analysis and navigation purposes

Develop AF Library

Build element templates based on requirements

Ensure template configuration is decoupled from the server environment

Include table connection configuration to make data outside PI accessible

Build Asset Model

Transform CIM model data into AF XML file format

Define elements based on both template and category information for maximum rollup analysis capability

DTAP Promotion

Develop, Test, Acceptance, Production

Compare previous model file and model configuration to current

Create delta files to import into target environments for reduced import times

Building Asset Model

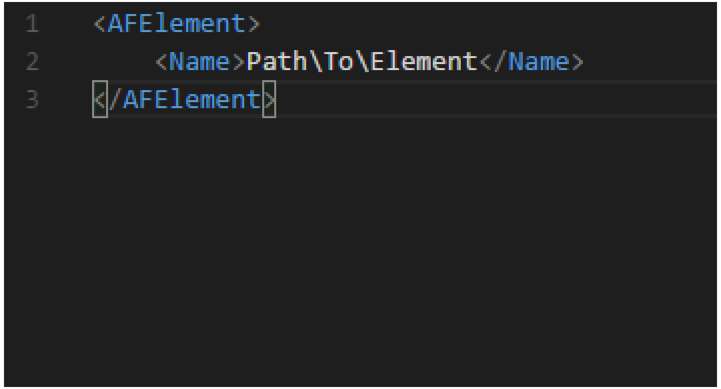
- Asset Framework model file built weekly using custom queries of CIM model (stored in Oracle database)
  - Data transform of CIM model snapshot
  - Formatted in AF XML schema format
- XML file defines element structure
  - Parent-child relationships and element references
  - Template, categories, and attribute values

Comparing Model Files

- Comparison of model files (previous and current) to produce difference file
  - Results in significantly faster import times (minutes versus hours)
  - Necessary to capture deleted or renamed assets
- Fundamental building block of XML file is the <AFElement> XML block
  - Traces a unique element path in the element hierarchy

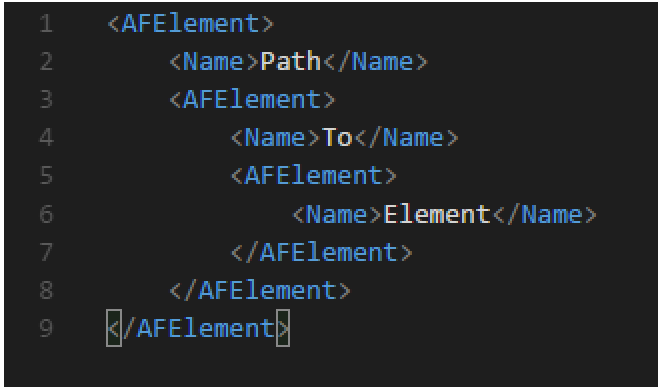
Model File Structure

Flat Structure



```xml
<AFElement>
    <Name>Path\To\Element</Name>
</AFElement>
```

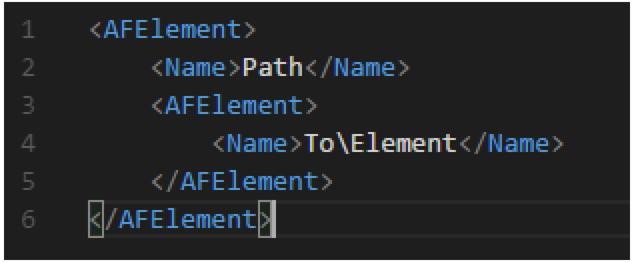
Nested Structure



```xml
<AFElement>
    <Name>Path</Name>
    <AFElement>
        <Name>To</Name>
        <AFElement>
            <Name>Element</Name>
        </AFElement>
    </AFElement>
</AFElement>
```

Model File Structure

- Can be combination of flat and nested structure
- Allows maximum flexibility for developers of CIM model data transform queries



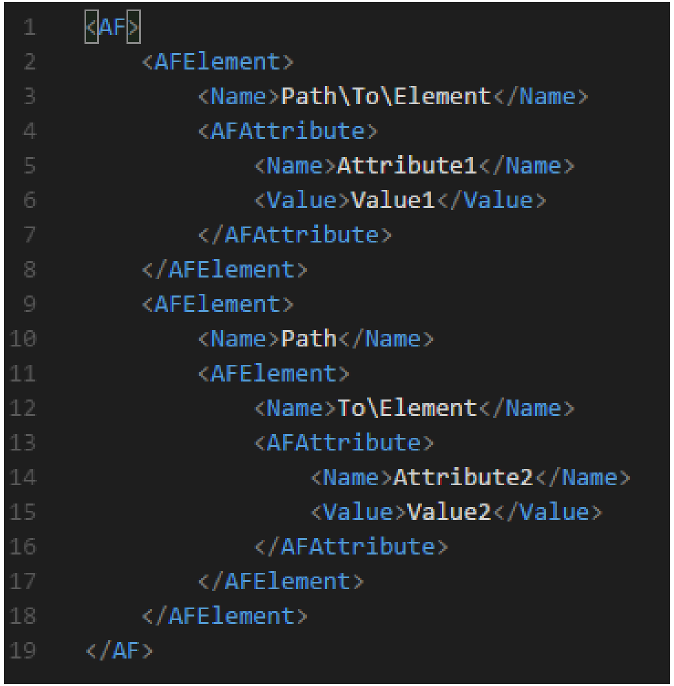
```
<AFElement>
    <Name>Path</Name>
    <AFElement>
        <Name>To\Element</Name>
    </AFElement>
</AFElement>
```

Model File Merging

- Merging process combines XML nodes that have identical paths
- Facilitates difference file creation algorithm
  - No need to account for multiple path locations
- Preserves original order of XML nodes
  - Necessary to avoid broken element reference dependencies
- Prescribes to AF XML schema validation rules

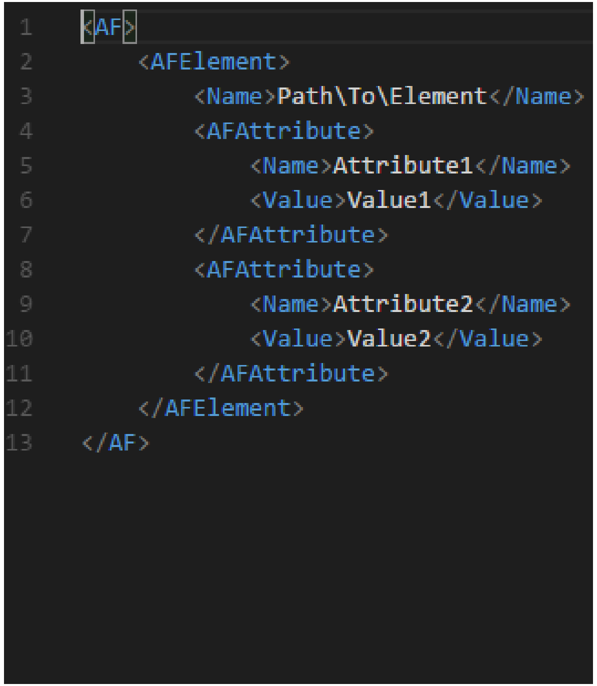
Example: Model File Merging

Before Merge



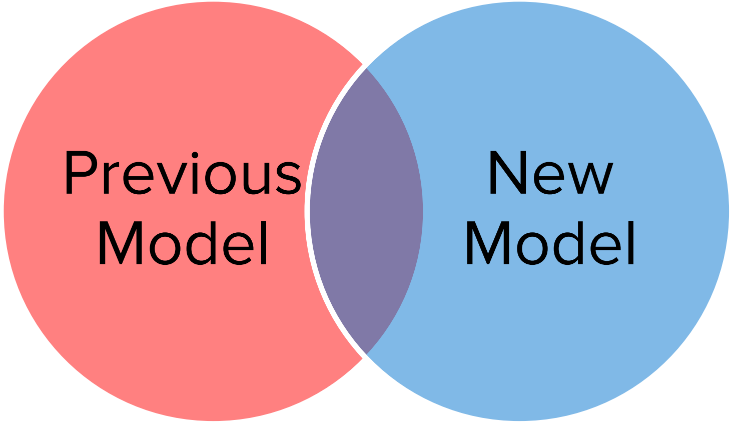
```xml
<AF>
    <AFElement>
        <Name>Path\To\Element</Name>
        <AFAttribute>
            <Name>Attribute1</Name>
            <Value>Value1</Value>
        </AFAttribute>
    </AFElement>
    <AFElement>
        <Name>Path</Name>
        <AFElement>
            <Name>To\Element</Name>
            <AFAttribute>
                <Name>Attribute2</Name>
                <Value>Value2</Value>
            </AFAttribute>
        </AFElement>
    </AFElement>
</AF>
```

After Merge



```xml
<AF>
    <AFElement>
        <Name>Path\To\Element</Name>
        <AFAttribute>
            <Name>Attribute1</Name>
            <Value>Value1</Value>
        </AFAttribute>
        <AFAttribute>
            <Name>Attribute2</Name>
            <Value>Value2</Value>
        </AFAttribute>
    </AFElement>
</AF>
```

Difference Detection



- Deleted elements
- Unchanged elements (not included)
- New elements
- Renamed elements

Difference File Creation

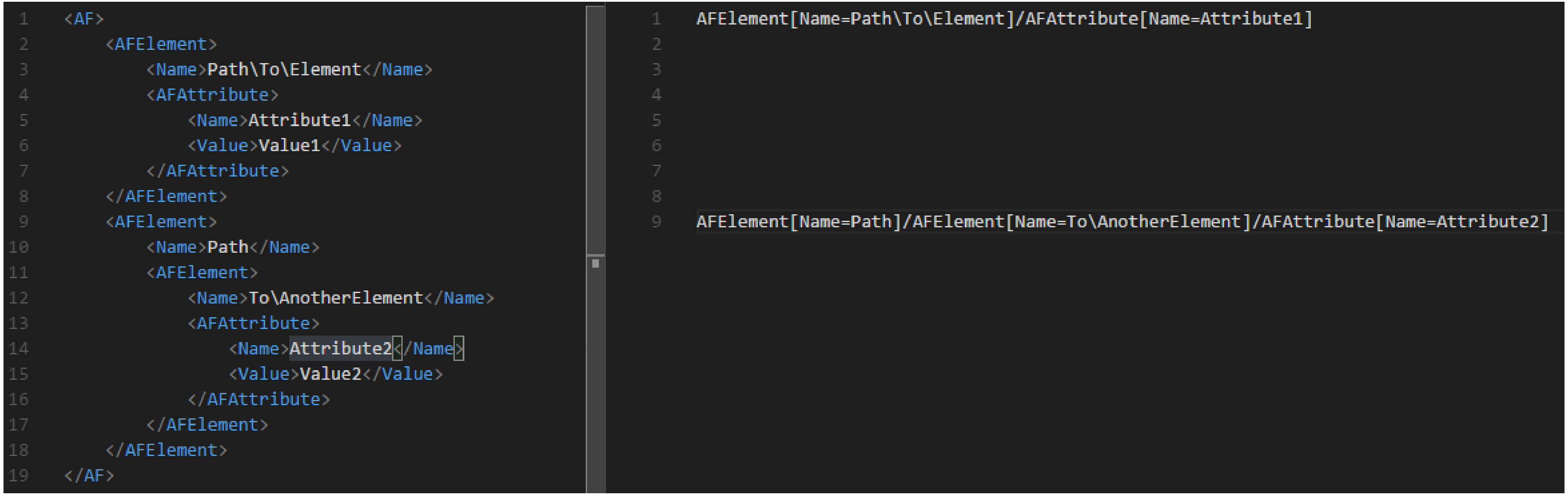
- File structure can include a combination of nested and flat element and/or attribute value structures
- Path to same element can be defined in multiple locations (developer flexibility)
- Alternative keys can be used for element rename detection, for example:
  - attribute value or extended property storing unique ID from CIM model

Difference Detection Algorithm

- Build compare key for all XML nodes using <Name> child node as primary key
- Operates from “inside-out”: XML nodes processed in order of decreasing depth

Example: Compare Key

- Path strings are guaranteed unique thanks to AF structure
- **Note** – Flat paths don't need to be expanded due to merging process

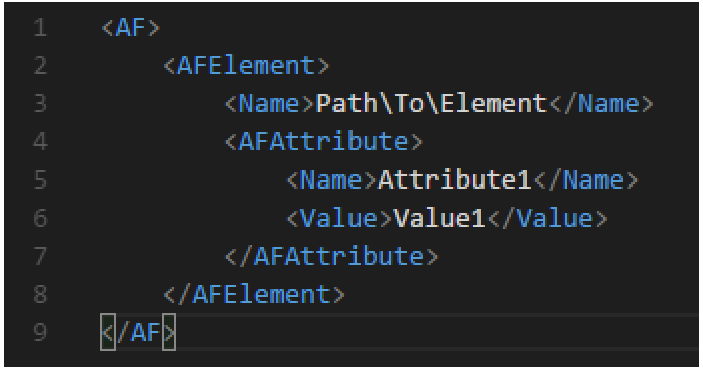


```
<AF>
    <AFElement>
        <Name>Path\To\Element</Name>
        <AFAttribute>
            <Name>Attribute1</Name>
            <Value>Value1</Value>
        </AFAttribute>
    </AFElement>
    <AFElement>
        <Name>Path</Name>
        <AFElement>
            <Name>To\AnotherElement</Name>
            <AFAttribute>
                <Name>Attribute2</Name>
                <Value>Value2</Value>
            </AFAttribute>
        </AFElement>
    </AFElement>
</AF>
```

```
AFElement[Name=Path\To\Element]/AFAttribute[Name=Attribute1]







AFElement[Name=Path]/AFElement[Name=To\AnotherElement]/AFAttribute[Name=Attribute2]
```

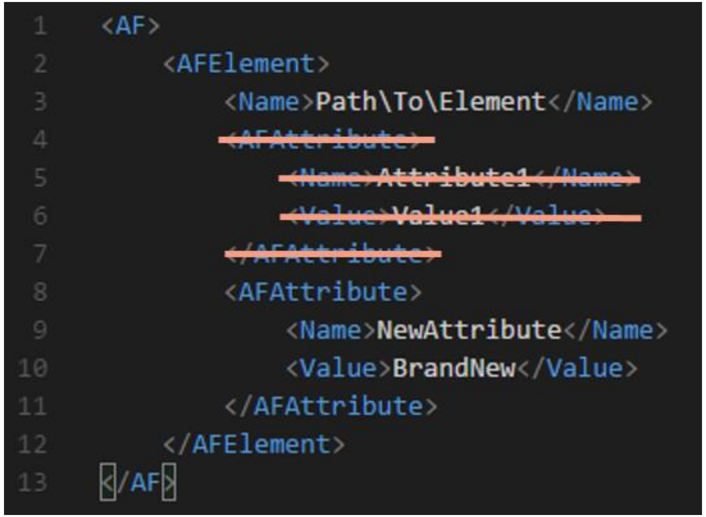
Example: Difference Detection

Previous Model



```
<AF>
    <AFElement>
        <Name>Path\To\Element</Name>
        <AFAttribute>
            <Name>Attribute1</Name>
            <Value>Value1</Value>
        </AFAttribute>
    </AFElement>
</AF>
```

New Model



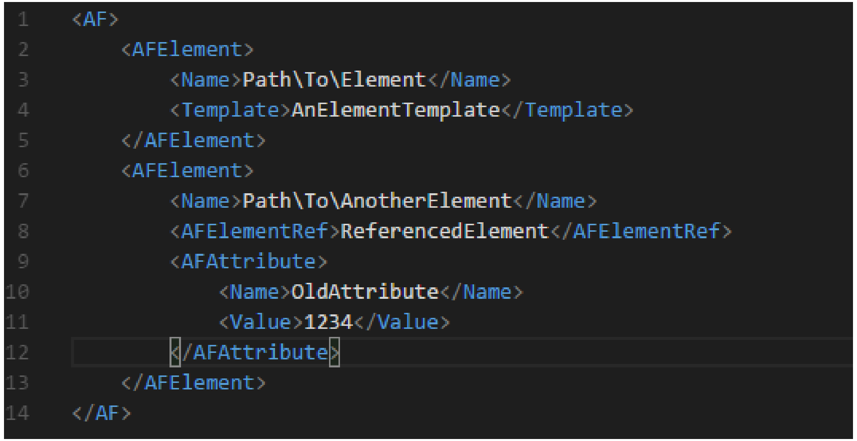
```
<AF>
    <AFElement>
        <Name>Path\To\Element</Name>
        ~~<AFAttribute>~~
            ~~<Name>Attribute1</Name>~~
            ~~<Value>Value1</Value>~~
        ~~</AFAttribute>~~
        <AFAttribute>
            <Name>NewAttribute</Name>
            <Value>BrandNew</Value>
        </AFAttribute>
    </AFElement>
</AF>
```

Deleted Elements/Attributes

- Element/attribute nodes missing from new model file indicate a deleted attribute or element
- Indicated by operation=“delete” attribute in XML
- Program must keep track of parent XML nodes in new model file with same path as parent node in old file
  - Needs knowledge of where to insert delete operation
- Note – Element references can also be deleted

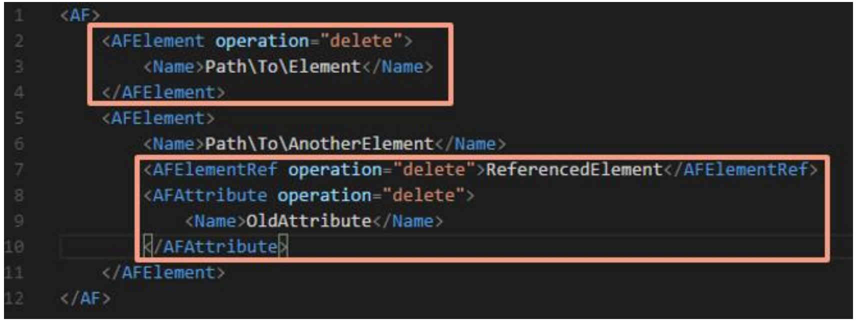
Example: Deleted Elements

Previous Model



```
<AF>
    <AFElement>
        <Name>Path\To\Element</Name>
        <Template>AnElementTemplate</Template>
    </AFElement>
    <AFElement>
        <Name>Path\To\AnotherElement</Name>
        <AFElementRef>ReferencedElement</AFElementRef>
        <AFAttribute>
            <Name>OldAttribute</Name>
            <Value>1234</Value>
        </AFAttribute>
    </AFElement>
</AF>
```

New Model



```
<AF>
    <AFElement operation="delete">
        <Name>Path\To\Element</Name>
    </AFElement>
    <AFElement>
        <Name>Path\To\AnotherElement</Name>
        <AFElementRef operation="delete">ReferencedElement</AFElementRef>
        <AFAttribute operation="delete">
            <Name>OldAttribute</Name>
        </AFAttribute>
    </AFElement>
</AF>
```

Special Case: Attribute Reset to Template

- Attribute nodes missing from new model which are part of an attribute template
  - Indicate a “reset to template” operation
  - Requires knowledge of AF library configuration
  - Sets attribute default value, or data reference/configuration string (if available)

Rename Detection

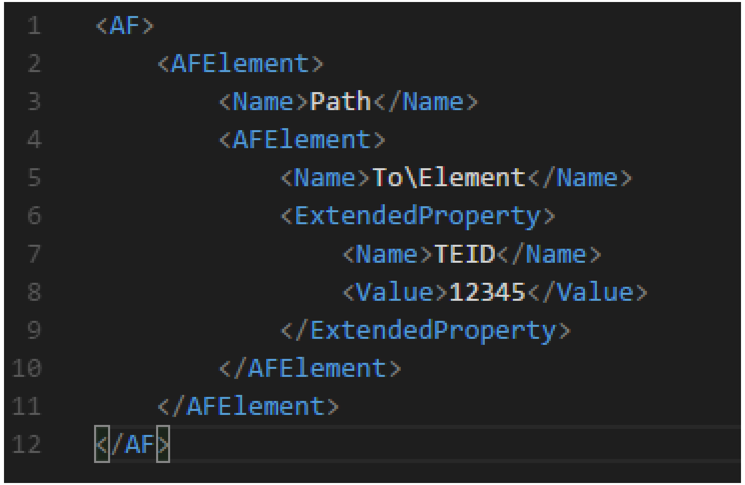
- Only necessary for <AFElement> nodes
- Build rename keys to all <AFElement> nodes using a combination of alternative keys (e.g., attribute or extended property value) and <Name> nodes
- Build element path strings to all <AFElement> nodes
- Renames occur when rename keys match but element path strings do not

Example: Rename Detection

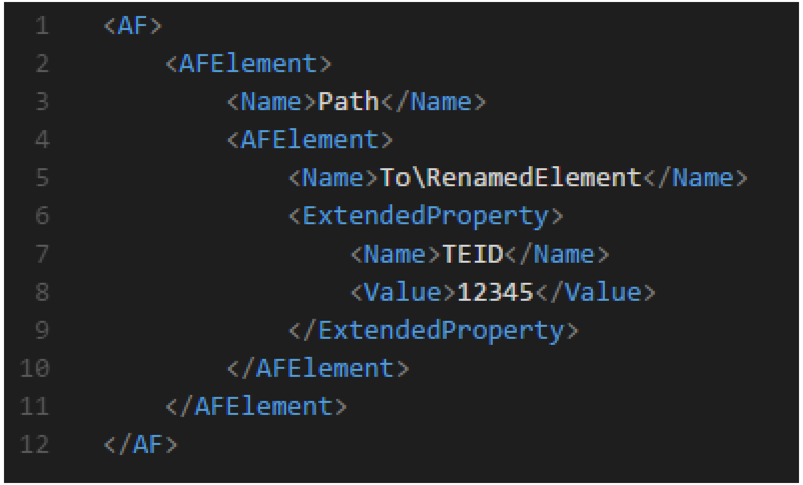
Previous: Path\To\Element

New: Path\To\RenamedElement

Compare key: AFElement[Name=Path]/AFElement[TEID=12345]



```xml
<AF>
    <AFElement>
        <Name>Path</Name>
        <AFElement>
            <Name>To\Element</Name>
            <ExtendedProperty>
                <Name>TEID</Name>
                <Value>12345</Value>
            </ExtendedProperty>
        </AFElement>
    </AFElement>
</AF>
```



```xml
<AF>
    <AFElement>
        <Name>Path</Name>
        <AFElement>
            <Name>To\RenamedElement</Name>
            <ExtendedProperty>
                <Name>TEID</Name>
                <Value>12345</Value>
            </ExtendedProperty>
        </AFElement>
    </AFElement>
</AF>
```

Renamed Elements

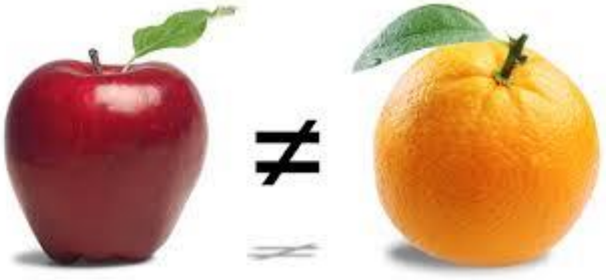
- Functionally the same as a create-plus-delete operation
- Rename detection facilitates PI tag renaming
- Note – Would be superior if operation="rename" attribute existed in the AF XML schema
  - Feature would automatically re-evaluate PI tag configuration strings and perform a tag rename if necessary

Library File Comparisons

- Development (or "gold standard") library export
  - Compared against multiple environment exports (test, production, etc.)
- Compare key generation follows similar logic
  - Traces ancestor XML nodes
  - Use <Name> child element as primary key
- Note – Rename detection not necessary for library files

Library File Normalization

- Library files require some preliminary transformations
  - Replace references to PI system and AF database in the “gold standard” library before comparing
  - Remove ‘override’ extended properties
  - Remove sensitive or environment specific information



Library Difference File Creation

- Create/update/delete detection follows same logic
- Some exceptions:
  - AFTable
  - AFTableConnection
  - AFAnalysisRule
  - AFTimeRule

Ongoing Challenges

- Analysis/notification instance statuses
  - Changes in development currently not propagated
- PI tag renames for attribute templates
  - Complicated by multi-part renames
  - Requires interaction with AF SDK or PI builder
- Better version control for library changes
  - Possible integration with Git/Bitbucket
- What to do with GUIDs?

Aaron Rosenthal

aaron.rosenthal@ercot.com

Operations Engineer II

Advanced Network Applications

ERCOT, Inc.

Thank You