AVEVAWORLD

OCTOBER 2024

Vattenfall Hydro Nordic

Using the SDK to create and monitor standardized data structures

Magnus Holmbom & Mattias Wångblad Inga, Vattenfall Hydro





ENERGY – HYDRO POWER | SWEDEN

Vattenfall standardizes AF and PI System Point structures by using SDK

Challenge

- Old AVEVA PI System structure built up over long time
- Geographically spread out with strong local focus
- Transition from focus on technicians to use by analysts

Solution

 Creation of SDK scripts to create a well templatised AF structure, PI Point naming and ensure a fleet wide comparable data quality

Results

- Standardized AF Structure with 150 turbines and 40.000 PI Points
- Increased Capacity: PI System team's increased ability to support new business cases
- Improved Efficiency: Reduced time spent on developing and maintaining PI AF templates
- Improved Efficiency: Reduced implementation time of new sensors
- Increased Capacity: Enabled fleet wide analysis and comparisons
- Increased Capacity: Enabled use of CBM in the organization



Introduction

This is Vattenfall

One of Europe's largest producers and retailers of electricity and heat

VATTENFALL



Introduction





8 million Electricity customers



Electricity network customers











Introduction

Location of our operations and major plants





Vattenfall

Our challenges with AVEVA PI System

- → Spread out over large area with isolated technicians
- → Varied fleet of powerplants built from 1910 to 1980s
- → Used PI System for over 10 years starting from small scale
- → Historically PI System has been used as monitoring tool for the technicians
- → Main focus on having a local connection between measurement value and equipment





Increasing demand for analysis

- → Production statistics/optimisation
- → Predictive maintenance
- → Weather and hydrology forecasting
- → Environmental reporting
- → Investment planning
- → Root cause analysis





- → SDK is .NET but possible to wrap in Python Inspired by the package PIConnect
- → Versatile and thorough in usage
- → Great documentation
- → More automated use of scripts instead of manual updates

<pre>def createGriftLid(base_child: AFELment, target_parent: AFELment): try: childTemplate = AFTemplate(base_child.element.Template) base_att = base_child.element.Attributes_get_Item('Tekniskt id') base_dt = base_child.element.Attributes_get_Item('Tekniskt id') base_dt = base_child.element.Elements.Add(name, childTemplate.element) att = med_l.MaxburgetLitement.element.figures_time('Tekniskt id') etc.ent Exception as te: print(neel.lement.element.time(texn) etc.ent Exception as e: print(neel.lements(base_parent: AFELment, target_parent: AFELment): for child in base_parent.element.template(childTemplate.element) etc.ent Exception as e: print(neel.lements(base_parent: AFELment, target_parent: AFELment): for child in base_parent.element.template(childTemplate.element) if not target_parent.element.template(childTemplate.element): for child in base_parent.afElment, target_parent: AFELment): for child in base_parent.element.template(childTemplate.element):</pre>		
<pre>d Created intruduade_child: Artimeter, target_parent: Artimeter); t childTemplate = AFTemplate(base_child.element.Template) base_dit = base_child.element.Attributes.get_ltem(Tisoniskt id') base_dit = base_child.element.Attributes.get_ltem(Tisoniskt id') base_dit = base_child.element.tilements.Add(name, childTemplate.element) att.conigString = att.conigString + base_id pi.Art.MameSubstitution.ResolveName(newEl) print(newEl.Name) cxccpt Exception as e: print(newEl.Name) cxccpt Exception as e: print(newEl.Name) def createChildElements(base_parent: AFEIement, target_parent: AFEIement): for childTemplate = AFTemplate(Child.element.Template) if not target_parent.basChildKithTemplate(childTemplate): newEl = target_parent.element.Element.Stad(name, childTemplate.element) for childTemplate = target_parent.element.Elements.Add(name, childTemplate.element) print(newEl.Name) print(newEl.Name) for childTemplate.sisSasedOn("printfd"); createDrift(d) f childTemplate.iSsasedOn("printfd"); createDrift(d)(id, target_parent) f childTemplate.iSsasedOn("printfd"); createDrift(d)(id, target_parent)</pre>		
<pre>v cos</pre>		<pre>+ createUrifttid(base_child: AFElement, target_parent: AFElement): +</pre>
<pre>init(emplate = Arlemplate(base_introlection=internet(implate))</pre>		
<pre>def = baie_chile.tement.thtribute; get_ttem('temist to ') baie_dt = baie_chile.tement.thtribute; get_ttem('temist to ') if not target_parent.ashumpHthd(base_dd): name - 'tog' def creatchildLements(base_parent: AFLement, target_parent: AFLement): for child in base_parent: AFLement, target_parent: AFLement): for child in base_parent: AFLement, target_parent: AFLement): for child in base_parent: AFLement.template(childLement.template): if not target_parent.elsent.template(childLement):</pre>		childremplate = Arlemplate(base_child.element.lemplate)
<pre>des_ls = hest_etrovale()(Alle()-3)) if ont target_parent.hasPamphill(hasc_ls); nest = target_parent.element.Elements.Add(name, childTemplate.element) att = newEl.Attributes.get_ltem('(Fenistit id')) att.conigEring = att.conigEring = tass_ld pl.47.AFNameSubstitution.ResolvetmemE(newEl) print(newEl.Name) except Exception as e: print(newEl.Name) except Exception as e: print(newEl.Name) def def createChildElements(base_parent: AFELment, target_parent: AFELment): f for child in base_parent.getChildTemplate(childTemplate): if not target_parent.hasPamphitChild(Name) if not target_parent.element.ElementS.Add(name, childTemplate.element) if not target_parent.element.ElementS.Add(name, childTemplate.element) if not target_parent.element.ElementS.Add(name, childTemplate.element) if childTemplate.idemEnt(ind.ResolveName(newEl)) if childTemplate.idemEnt(Frue) if childTemplate.idemEnt() if child</pre>		<pre>base_att = base_child.element.Attributes.get_item('lekniskt id') base_base_base_child.element.Attributes.get_item('lekniskt id')</pre>
<pre>if not target_parent.hashapairinl((base_Jd):</pre>		<pre>base_id = base_att.GetValue().Value[-3:]</pre>
<pre>newEl = traget_parent.element.Elements.Add(name, childTemplate.element) att = newEl.Attributes.get_Item('feknisti id') att.corrigitring = att.corrigitring = hsse_id pi.At.NewElsubstitution.ResolveName(newEl) print(newEl.Name) except Exception as e: print(newEl.Name) v 0x d of createChildElements(base_parent; AfElement, target_parent; AfElement); for childTemplate = AfTemplate(hild.element.Template) if not target_parent.hssChildHithTemplate(childTemplate); newEl = target_parent.element.Element.Elements.Add(name, childTemplate.element)</pre>		if not target_parent.hasPumpWithId(base_id):
<pre>def createChildElements/bad(name, childTemplate.element)</pre>		name = 'tmp'
<pre> att = newEl.Attributes.get_Item('Tewnist id') pi.K.ArNameSubstitution.ResolveName(newEl) pi.K.ArNameSubstitution.ResolveName(newEl) print(newEl.Name) except Exception as e: print(newEl.Name) def createChildElement(base_parent: AFELement, target_parent: AFELement): for child in base_parent.getChildEren(): for child in base_parent.atChildElement.Template) if not target_parent.natChildElement.Template) if not target_parent.atclindEliner(): for dhild in base_parent.atChildElement.Template) if not target_parent.atclindEliner(): for dhild in base_parent.atclindEliner(): for dhild in base_parent.atclindEliner(): for dhild in base_parent.atclindEliner(): for dhildEment(base_parent.atclindEliner(): for dhildEment(): for attribute(): for attribute():</pre>		<pre>newE1 = target_parent.element.Elements.Add(name, childTemplate.element)</pre>
<pre>ded</pre> <pre>d</pre>		<pre>att = newEl.Attributes.get_Item('Tekniskt id')</pre>
<pre>def createChildElements(base_parent: AFEIement, target_parent: AFEIement):</pre>		att.ConfigString = att.ConfigString + base_id
<pre>/</pre>		pi.AF.AFNameSubstitution.ResolveName(newE1)
<pre>except Exception as e: print(e) newEl.UndoCheckOut(True) v os</pre>		print(newEl.Name)
<pre>print(e) newEl.UndoCheckOut(True) v dos def createChildElements(base_parent: AFElement, target_parent: AFElement): for child in base_parent.getChildren(): childTemplate = AFTemplate(child.element.Template) if not target_parent.basChildWithTemplate(childTemplate): if not target_parent.base.basetUnton.ResolveBame(newEl) pTi.AF.NameSubstitution.ResolveBame(newEl) pTi.AF.NameSubstitution.ResolveBame(newEl) pTi.AF.NameSubstitution.ResolveBame(newEl) pTi.AF.NameSubstitution.ResolveBame(newEl) if childTemplate.isBasedDm('_Driftid'); createDriftid(child, target_parent) /</pre>		except Exception as e:
<pre>def creatchildElements(base_parent: AFElement, target_parent: AFElement): for child in base_parent.getChildren(): for child in base_parent.afElement, target_parent: AFElement): for child an base_parent.asChilddithTemplate(childTemplate): neme = 'temp' recit = target_parent.element.Element.Add(name, childTemplate.element) print(newEl.Hame) except Exception as e: print(newEl.Hame) createDriftLid(child, target_parent) if childTemplate.idsaedDn('parent): createDriftLid(child, target_parent) /</pre>		print(e)
<pre>de def creatsChildElseents(base_parent: AFElement, target_parent: AFElement): for child in base_parent.getChildren(): childrenplate = AFEmplate(child.elsemt.Template) if not target_parent.hasChildWithTemplate(childTemplate): if neeEl = target_parent.elsemt.ElsementS.Add(name, childTemplate.elsement) pl.iAF.NameSubstitution.ResolveName(newEl) pl.iAF.NameSubstitution.ResolveName(newEl) if childTemplate.isBasedOn('_Dreiftid'); createDriftid(child, target_parent) / acb </pre>		newEl.UndoCheckOut(True)
<pre>def createChildElements(base_parent: AFElement, target_parent: AFElement): for child in base_parent.getChildren(): for child in base_parent.aFElement, target_parent: AFElement): for ctarget_parent.aFElement.clement.complate) if not target_parent.element.element.Add(name, childTemplate.element) resell = target_parent.element.element.Add(name, childTemplate.element) print(name1.Mame2) eccept Exception as e: print(name1.Mame2) for childTemplate.elsment) for childTemplate.elsment) for childTemplate.elsment) print(name1.Mame2) cccept Exception as e: print(name1.Mame2) ccceptSolititid(child, target_parent) v ada </pre>		
<pre>def createChildElement(base_parent: AFElement, target_parent: AFElement): for child in base_parent.getChildren(): childTemplate = AFTemplate(child.element.Template) if not.target_parent.basChildWithTemplate(childTemplate): nome = 'tsp' try: medi = target_parent.element.Element.Add(name, childTemplate.element)</pre>	✓ 0.	Us
<pre>def createChildElements(base_parent: AFElement, target_parent: AFElement): for child in base_parent.getChildren(): for child in base_parent.etChildren(): for ctarget_parent.hasChilddithTemplate(childTemplate): namee = 'temp'; review = target_parent.element.element.Add(name, childTemplate.element) print(newEl.Name) except Exception as e: print(newEl.Name) for child inductor(True) if children(): createOriftid(child, target_parent) // cae // ac // ac</pre>		
<pre>def createChildIsemitS(bate_parent: AFElement; target_parent: AFElement): for childIsemitS(bate_parent.getX)interno(): childTemplate = AFTemplate(hild.element.Template) if not target_parent.hasChildWithTemplate(childTemplate): name = 'tmp' try: newEl = target_parent.element.floments.Add(name, childTemplate.element)</pre>		
<pre>for child in base_parent.getChildren():</pre>		f createChildElements(base_parent: AFElement, target_parent: AFElement):
<pre>childTemplate = AfTemplate(child.elment.Template) if not target_parent.hasGhildWithTemplate(childTemplate):</pre>		for child in base_parent.getChildren():
<pre>if not target_parent.hasChildHithTemplate(childTemplate): name = 'tap' row: name = 'tap' row: newEl = target_parent.element.Elements.Add(name, childTemplate.element) pI.AF.AFNameSubstitution.ResolveName(newEl) print(newEl.Name) except Exception as e: print(newEl.Name) rewEl.UndoCheckOut(True) if childTemplate.is8asedOn('_Driftid'): createDriftid(child, target_parent)</pre>		<pre>childTemplate = AFTemplate(child.element.Template)</pre>
<pre>name = 'tmp' try: newGl = AtlaneSubstitution.ResolveName(newEl) pl.AF.AtlaneSubstitution.ResolveName(newEl) resolveName(newEl) createDriftId(child, target_parent) v</pre>		<pre>if not target_parent.hasChildWithTemplate(childTemplate):</pre>
<pre>try: newEl = target_parent.element.Elements.Add(name, childTemplate.element) pI.AF.AFlameSubstitution.ResolveName(newEl) print(newEl.Hame) except Exception as e: print(ewEl.Hame) rewEl.UndoCheckOut(True) if childTemplate.isBasedOn('_Oriftid'): createDriftId(child, target_parent)</pre>		name = 'tmp'
<pre>medi = target_parent.clement.laments.Add(name, childTemplate.clement)</pre>		try:
<pre>PI.AF.AFMamSubstitution.ResolveName(newE1) print(newE1.Name) except Exception as e: print(e) newE1.UndoCheckOut(True) if childTemplate.is8asedOn('_DriftId'): createDriftId(child, target_parent) ✓ @@</pre>		<pre>newEl = target parent.element.Elements.Add(name, childTemplate.element)</pre>
<pre>print(newElMase) except Scapiton as e: print(e) restLubacheckou(Ture) if childTemplate.is8ase0n('_DriftId'): createDriftId(child, target_parent) ✓ @&</pre>		PT.AF.AFNameSubstitution.ResolveName(newF1)
<pre>vcept Coupling as e: print(e) newEl.UndoCheckOut(True) if childTemplate.is8asedOn('_Driftiid'): createDriftid(child, target_parent)</pre>		print(newFl.Name)
<pre>> Comparint(0) neeEl.DowdOnecdodt(True) if childTemplate.isBasedOn('_Driftld');</pre>		event Evention as e
<pre>protect protect (True) if childremplate.is8asedOn('_Driftld'): createDriftld(child, tanget_parent) ✓ ads</pre>		print(a)
<pre>if child(child, targe_parent)</pre>		princes
<pre>/ functionegrate.isbaseduit _pirt(in):</pre>		is shidten has a Record () bisterid).
Creates/Intlig(child, target_barent) ✓ 0.6		1 - chturempice::sbabedon(Onitetta);
		createUrifttid(child, target_parent)
× 0.0s		
j √ 0.0s		
	1 🗸 0	.05
def createFlomentr/bace paperts AFFloment, target paperts AFFloment);	do	another of the another the another the another and the another the
<pre>def createElements(base_parent: AFElement, target_parent: AFElement):</pre>		createchildelements(base_parent, target_parent)
<pre>def createElements(base_parent: AFElement, target_parent: AFElement):</pre>		
<pre>def createElements(base_parent: AFELement, target_parent) AFELement): createChildElement(base_parent, target_parent) for target_parent, target_parent, targetChildren();</pre>		<pre>for target_child in target_parent.getChildren():</pre>
<pre>def createElements(base_parent: AFELement, target_parent: AFELement): createChildElements(base_parent, target_parent): for target_child in target_parent.getChildren(): template = AFEmplate(target_child.clement.Template) </pre>		<pre>for target_child in target_parent.getChildren(): template = AFTemplate(target_child.element.Template)</pre>
<pre>def creat@lements(base_parent: AFElement, target_parent: AFElement): creat@childlements(base_parent, target_parent) for target_child in target_parent.getChildren(): template = AFTemplate(target_child.element.Template) base_child = base_parent.getChildWithemplate(template)</pre>		<pre>for target_child in target_parent.getChildren(): template = AFTemplate(target_child.element.Template) base_child = base_parent.getChildWithTemplate(template)</pre>
<pre>def createElements(base_parent: AFELement, target_parent: AFELement): createChildElements(base_parent, target_parent) for target_child in target_parent.getChildren(): template = AFTemplate(target_child.element.template) bose_child = base_parent.getChildWithTemplate(template) if base_child:</pre>		<pre>for target_child in target_parent.getChildren(): template = AFenplate(target_child.clearnt.template) bas_child + base_parent.getChildWithTemplate(template) if base_child:</pre>
<pre>def creat@lements(base_parent: AFElement, target_parent: AFElement): creat@childBlements(base_parent, target_parent) for target_child in target_parent, targetChildren(): template = AFTemplate(target_child.element.Template) bosc_child = bosc_parent.getChildRightBlentLemplate(template) if base_child: creat@lements(base_child, target_child)</pre>		<pre>for target_child in target_parent.getChildren(): tanglate = AFTanglate(target_child element_semplate) base_parent_getChilddithTemplate(tanglate) if base_child: createtElementS(base_child, target_child)</pre>
<pre>def createElements(base_parent: AFELement, target_parent: AFELement): createChildElements(base_parent, target_parent) for target_child in target_parent, setChildren(): template = AFTemplate(target_child.element.Template) base_child = base_parent.getChildWithTemplate(template) if hase_child createElements(base_child, target_child) else: } } </pre>		<pre>for target_child in target_parent.getChildren(): template = AFenplate(target_child.element.Template) bas_child = base_parent.getChildWithTemplate(template) if base_child: createElements(base_child, target_child) else:</pre>
<pre>def createElements(base_parent: AFELement, target_parent: AFELement): createChildElements(base_parent, target_parent) for target_child in target_parent, targetChildren(): template = AFTemplate(target_child.element.Template) base_child = base_parent.getChildWithTemplate(template) if base_child: createElements(base_child, target_child) else: orint("No: * + target child.element = * " in base") </pre>		<pre>for target_child in target_parent.getChildren(): template = AFrequite(target_child = hearent.lemplate) base_parent_getChild#ithTemplate(template) if base_child: createfLeents(base_child, target_child) else: print("No: " + target_child.lemmt.Name + " in base")</pre>
<pre>def createflements(base_parent: AFELement, target_parent: AFELement): createChildLements(base_parent, target_parent) for target_child in target_parent.getChildren(): template: AFTemplate(target_child.element.remplate) base_child = base_parent.getChildren(): if base_child: createflements(base_child, target_child) cls: print("No: " + target_child.element."mages") </pre>		<pre>for target_child in target_parent_getChildren(): template = AFrequist(exprest_child.element.Template) bess_child = base_parent_getChilddithTemplate(template) if base_child = child, target_childdithTemplate(template) if base_nhild: createflements(base_child, target_child) else: print("No: " + target_child.element.Name + " in base") </pre>



AF Structure generation

- → Tedious to set up 150 turbines manually
- → We have created SDK script to generate the entire template structure
- → All is templatised, easy setup of PI Vision displays
- → Uses a referenced template structure
- → Roll out update on all turbines in case of change
- → Gap analysis on missing PI Points
- → Reduces risk of errors during AF tree generation





SDK usage

PI Point renaming

- → Loops through existing PI Points and recommends new name that fits AF templates
- → Mixture of language and AI models
- → Generating descriptions to couple the point to the actual equipment
- → Some manual work





Data analysis through SDK

- → Quick combination/analysis of data
- → Transfer data to cloud using SDK
- → Healing faulty data
- → Combining after sensor switch





Next step

Data quality monitoring

- → Next step
- → Compare all standardised signals and check for deviations
- → Data rate monitoring
- → Fluctuation deviations

-	-	Karrine.	Interior State	-		-	-
-	-	-	11.04.0	1811	-	Nation 1	- 13
					100	- Date:	
1000	Inclusion in the	-		100	1.000	Raa)	-
				- 10	-	Nation 1	
-	The second second	1				No.	-
						1000	-
ACCES.	Constant Printed	Design Ball	11.000		11984	841	
	THE OWNER		11.000		Sec.	1.0	
-	Carlo Carlos	Conception in the local division of	111		100	Test	
	Concession in succession	-			- 861	1.0	
-	-					100	
and the set	Contractor of Contractor		10000		A local	199	
	Card Fred				10.0	2.00	
-	Intel Party	Theory Pro-	1000		1.00	1990	1
		Thereis Maria	11.04	-	to the second	17.00	-
			HISING STREET		1011	Figs.	11

	a teretan	17	and the lot of the lot		11.1
CONTRACTOR OF CONTRACT			There are		100
		Concession in concession of the local data	Services.	100	1.51
		-	the second s	e	
		-	the second	-	1
			and the second s	-	
		-	and the second s	144	-
		1	termine .	811	-

-	-	-	See See 1	a primary and	AL OWL DOG AL
	-	Name of Street, or other	1000		16.2
Contract I	Banarian Million	the prove	A State	and the second second	100
-	Date: MIC	Terr manners	Contract of the local division of the local		
-	Income Second		R. Barr		
		Distance.	a des		
-	1 Capel Process	THE BURGET	10100		
1708	No. All	-	ALC: NO	-	10.1
	1000		0.000	-	100
-	Base Toola	Internet	States 1		100
			R. Barr		10 A
-	THE OWNER	and the second	R(Bart)	100	14



Outcome

- → Enabled Condition Based Monitoring
- → Simplified the setup of any new analysis
- → Show increased maintenance costs in the fleet due to new production patterns
- → Gap analysis to identify stations which lacks important features





Lessons learned

- **The Importance of Naming Standards**: Ensure consistency and clarity in all data management.
- Data/AF Structure: Essential for effective analysis deployment.
- Analysis Needs Drive Structure: Tailor the structure to meet specific analysis requirements.
- **Data Quality and Standardization**: The foundation for reliable and comparable results.
- Automation and Efficiency with Python: Maximize productivity through scripting.
- Using PI SDK for Data Transfer: Secure and efficient data transfer.
- **Preliminary Data Studies (Business Cases)**: Analyse and visualize data to simplify understanding before setting up the entire workflow







ENERGY – HYDRO POWER | SWEDEN

Vattenfall standardizes AF and PI System Point structures by using SDK

Challenge

- Old AVEVA PI System structure built up over long time
- Geographically spread out with strong local focus
- Transition from focus on technicians to use by analysts

Solution

 Creation of SDK scripts to create a well templatised AF structure, PI Point naming and ensure a fleet wide comparable data quality

Results

- Standardized AF Structure with 150 turbines and 40.000 PI Points
- Increased Capacity: PI System team's increased ability to support new business cases
- Improved Efficiency: Reduced time spent on developing and maintaining PI AF templates
- Improved Efficiency: Reduced implementation time of new sensors
- Increased Capacity: Enabled fleet wide analysis and comparisons
- Increased Capacity: Enabled use of CBM in the organization



This presentation may include predictions, estimates, intentions, beliefs and other statements that are or may be construed as being forward-looking. While these forward-looking statements represent our current judgment on what the future holds, they are subject to risks and uncertainties that could result in actual outcomes differing materially from those projected in these statements. No statement contained herein constitutes a commitment by AVEVA to perform any particular action or to deliver any particular product or product features. Readers are cautioned not to place undue reliance on these forward-looking statements, which reflect our opinions only as of the date of this presentation.

The Company shall not be obliged to disclose any revision to these forward-looking statements to reflect events or circumstances occurring after the date on which they are made or to reflect the occurrence of future events.



ABOUT AVEVA

AVEVA is a world leader in industrial software, providing engineering and operational solutions across multiple industries, including oil and gas, chemical, pharmaceutical, power and utilities, marine, renewables, and food and beverage. Our agnostic and open architecture helps organizations design, build, operate, maintain and optimize the complete lifecycle of complex industrial assets, from production plants and offshore platforms to manufactured consumer goods.

Over 20,000 enterprises in over 100 countries rely on AVEVA to help them deliver life's essentials: safe and reliable energy, food, medicines, infrastructure and more. By connecting people with trusted information and AI-enriched insights, AVEVA enables teams to engineer efficiently and optimize operations, driving growth and sustainability.

Named as one of the world's most innovative companies, AVEVA supports customers with open solutions and the expertise of more than 6,400 employees, 5,000 partners and 5,700 certified developers. The company is headquartered in Cambridge, UK.

Learn more at www.aveva.com