DEXPI Process Modelling of Process Systems and their Documentation

Authors: David Cameron, Wilhelm Otten, Heiner Temmen, Gregor Tolksdorf
DEXPI+ Project Team: David Cameron (University of Oslo), Andreas Schüller (NAMUR), Anselm Klose (TU Dresden), Behnam Ghahraman (Aucotec), Eric Carnet (Aveva), Iskandar Halim (ISCEE), Leon Hanke (Aucotec), Maged Selim (Aveva), Manfred Theißen (PNB), Martin te Lintelo (USPI), Monica Hole (Aibel), Idar Pe Ingebrigtsen (Equinor)

Version	Date	Description
0.1	2022-11-30	First issue for review
0.2	2022-11-30	Changes from meeting 2022-11-30
0.3	2023-04-25	XML schema and additional parameters.
RC 1.0	2023-06-14	First Release Candidate
RC 1.1	2023-09-12	Second Release Candidate. Renamed to DEXPI Process.
1.0	2023-12-08	Official release with updated model diagrams

Contents

1 Introduction: Modelling of Process Block and Process Flow Diagrams

The DEXPI P&ID specification defines an information model for Piping and Instrumentation diagrams (PID). This model covers the product aspects of engineering and construction of a process plant. However, there is no corresponding standard information model applicable for Process Flow Diagrams (PFD) and Block Flow Diagrams (BFD). This is a gap in the coverage of standards in process engineering. The PFD and BFD cover the process and functional aspects of the design that are developed in early-phase engineering and provide the design constraints for detailed design, procurement and construction.

The DEXPI Process information model, described here, represents the information in PFDs and BFDs. In addition, it provides a data model for representing process design in a way that allows traceability of requirements and design choices through the engineering lifecycle. The model describes the process systems that are represented in the diagrams. It also provides a framework for defining and providing the design data that is shown on PFDs.

The information model is defined as an additional package in the DEXPI data model

A presentation layer allows this information model to be rendered as a PFD or BFD. Together, the model and the presentation layer provide a standard exchange format for these diagrams. This document describes the information model and demonstrates its use to represent four reference diagrams. Further work is needed to define the presentation layer and its relation to the information model.

2 Building the Information Model: Process Systems and Activities

2.1 Overview: Modelling the Process

The information model required to support PBDs and PFDs needs to model the *process* in a facility, as distinct from the *plant*, which is the realization of these processes.

We model the process as a set of interconnected *process steps,* where a process step is a *system* that performs a *process activity*. This approach is consistent with the OntoCAPE ontology for Chemical Process Engineering.

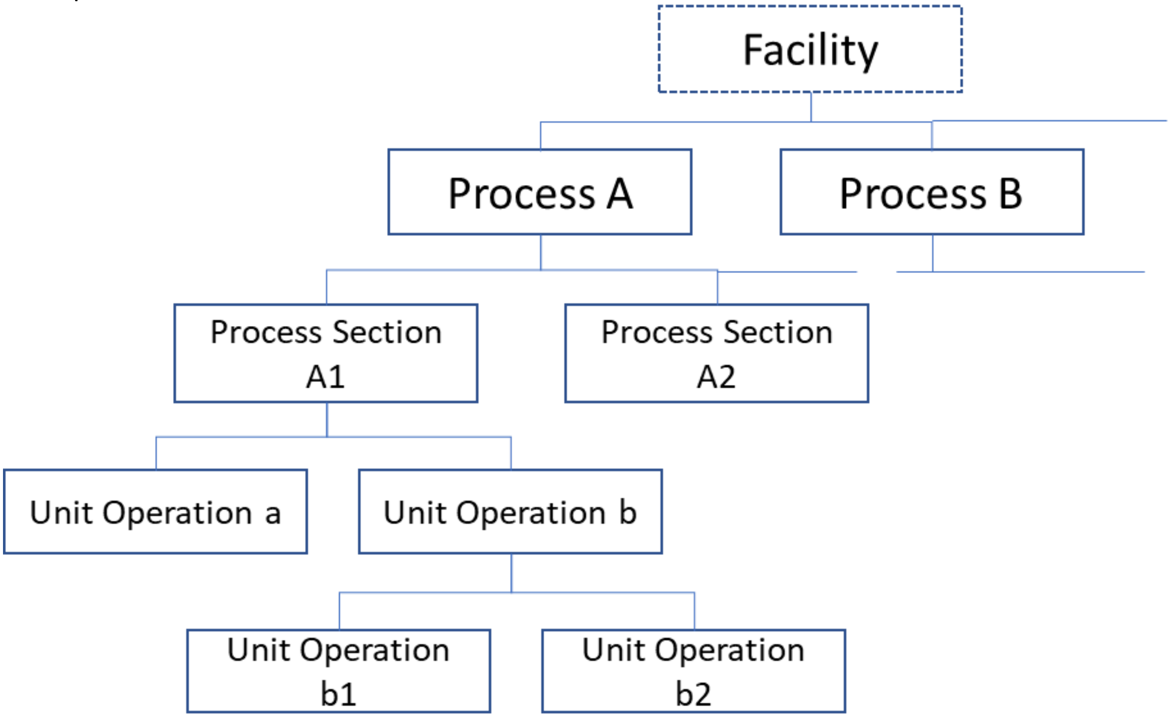
A (process) *activity* provides the semantic description, the type, of a process step. It is therefore the point of linkage to ISO15926 parts 4 and 14. This also provides a point of linkage with the Reference

Designation Systems based on ISO/IEC 81346, either as described in part 2 of that standard or in the DIN6779-13.

Process activities are characterized by a verb. In naming our classes we have chosen to use the present participle form (e.g. **Pumping**, **Regulating**). Names of unit operations are formed by adding the noun for the object of the action (e.g. **RegulatingFlow**) or an adverbial phrase about how the activity is done (e.g. **SeparatingByCentrifugalMotion**).

2.2 Hierarchy of Process Steps

The conceptual design of a processing facility is done by defining a hierarchy of process steps, presented in DIN6779-13. We start at a high level, defining a single **process** to model a whole production system. We then break the process into smaller steps, **process sections**, producing the PBD. Each of these sections are then broken down further into smaller steps, **unit operations**. These unit operations are shown on the PFD.



This hierarchy corresponds to the ISO/IEC 81346 hierarchy of systems, with the following relations.

DIN6779-13	ISO/IEC81346
Process	Functional System
Process Section	Technical System
Unit Operation	Component System

2.3 Representation of Systems

Each DEXPI Process system class inherits a base structure of a system block: a **ProcessStep** that hosts **Port** and **Parameter** objects.

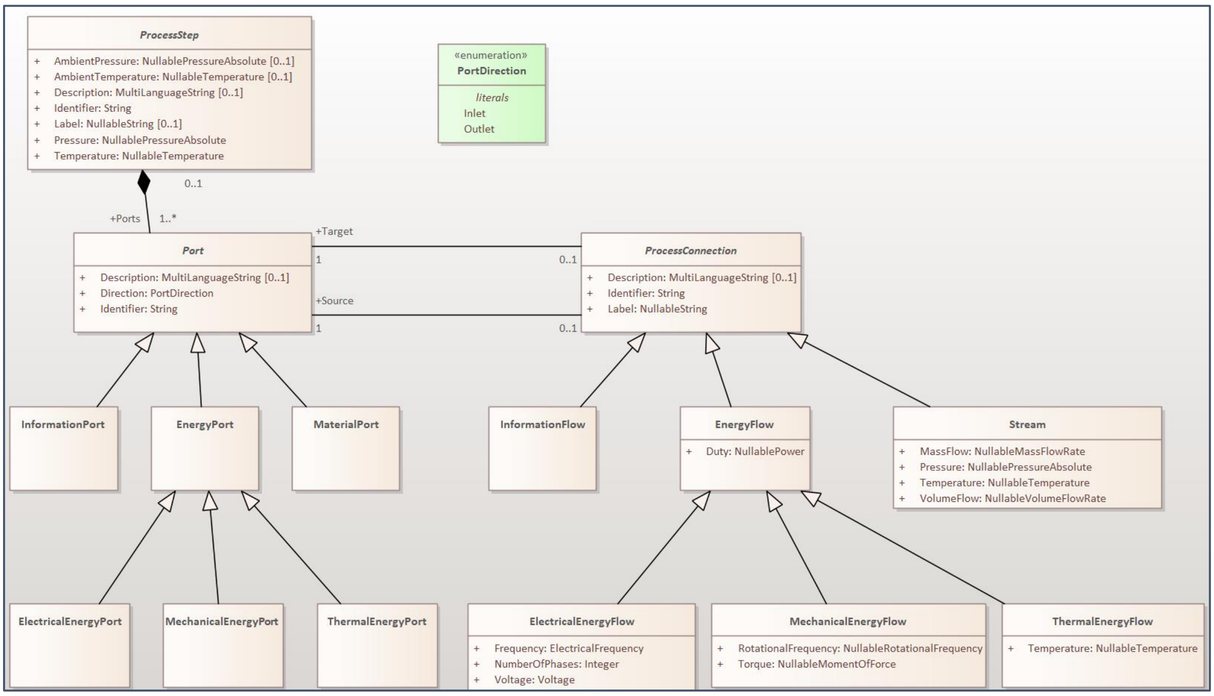
2.3.1 Ports

A **Port** provides an interface, where a system can interact with another system. Each port has a nominal direction, given by the *PortDirection* enumeration, namely **Inlet** or **Outlet**. Process systems interact through exchange of material, energy, or information. The ports provide reference points for information about these exchanges: their *state*. Exchange of material corresponds to the concept of a **stream** in ISO15926. An exchange of energy corresponds to an **energy stream**. An exchange of information corresponds to a **process signal**.

In the DEXPI Process model we have named ports using a variant of the coding proposes by DIN 6779-13. These codes are shown in the following table.

Port Type	DEXPI+ Flow Class	ISO15926 Concept	DIN 6779-13 code
MaterialPort	**Stream**	Stream	XL
EnergyPort	**EnergyFlow**	Energy Stream	XN
ElectricalEnergyPort	**ElectricalEnergyFlow**		XD
ThermalEnergyPort	**ThermalEnergyFlow**		XQ
MechanicalEnergyPort	**MechanicalEnergyFlow**		XN
InformationPort	**InformationFlow**	Process Signal	XG

The exchange is represented by a **ProcessConnection** object. This links an inlet port and an outlet port and provides a reference to the **Stream**, **EnergyFlow** or **InformationFlow** that is exchanged between these ports.



Note that a process connection has no physical or logical extent. It specifies that the physical state of the material, energy and information is the same at each of the connected ports.

2.3.2 Process Connections

This also means that *transport, or conveying,* of energy, material or information is not modelled by the **ProcessConnection** object. Where a process design needs to model transport explicitly, we use process steps such as **TransportingFluids**, **TransportingSolids** or **TransportingElectricalEnergy** to represent these steps. Here it is assumed that the state of the inlet to the system is not the same as the state of the outlet.

This model fits with the model described in VDI/VDE 3682 Blatt 1 [2], where a material is described as a “product”.

2.3.3 Linking Ports between Models with Varying Levels of Detail

Ports also provide a way of linking between models at different levels of process hierarchy. A port can be related to ports in a block in a model with lower detail or higher detail.

We model this using two optional attributes in a Port:

- **SuperReference**: this port corresponds to the indicated port in a higher-level, less detailed model.
- **SubReference**: this port corresponds to the indicated port in a lower-level, more detailed model.

This allows the embedding of one or more PFDs within a BFD. The blocks in the BFD provide a frame for the blocks in the PFD. Ports on the blocks in the BFD are aligned using **SubReference** links with ports in blocks on the boundary of the PFD. These ports in the PFD in turn contain **SuperReference** links to the BFD.

2.3.4 Parameters

Parameters contain information about the process step itself. A parameter may be embedded in the process step itself, or it is part of a flow structure at a port.

This model is consistent with the SysML modelling language for systems engineering. This means that a DEXPI Plus information model can be built in SysML using Model-based System Engineering tools.

2.4 Structure of Streams, Energy Flows and Information Flows.

Streams and energy flows need to be shown in stream tables and referred to by labels on the connections in the model. DEXPI plus uses a model that is derived from the OntoCAPE ontology and the CAPE-OPEN data model for simulation data.

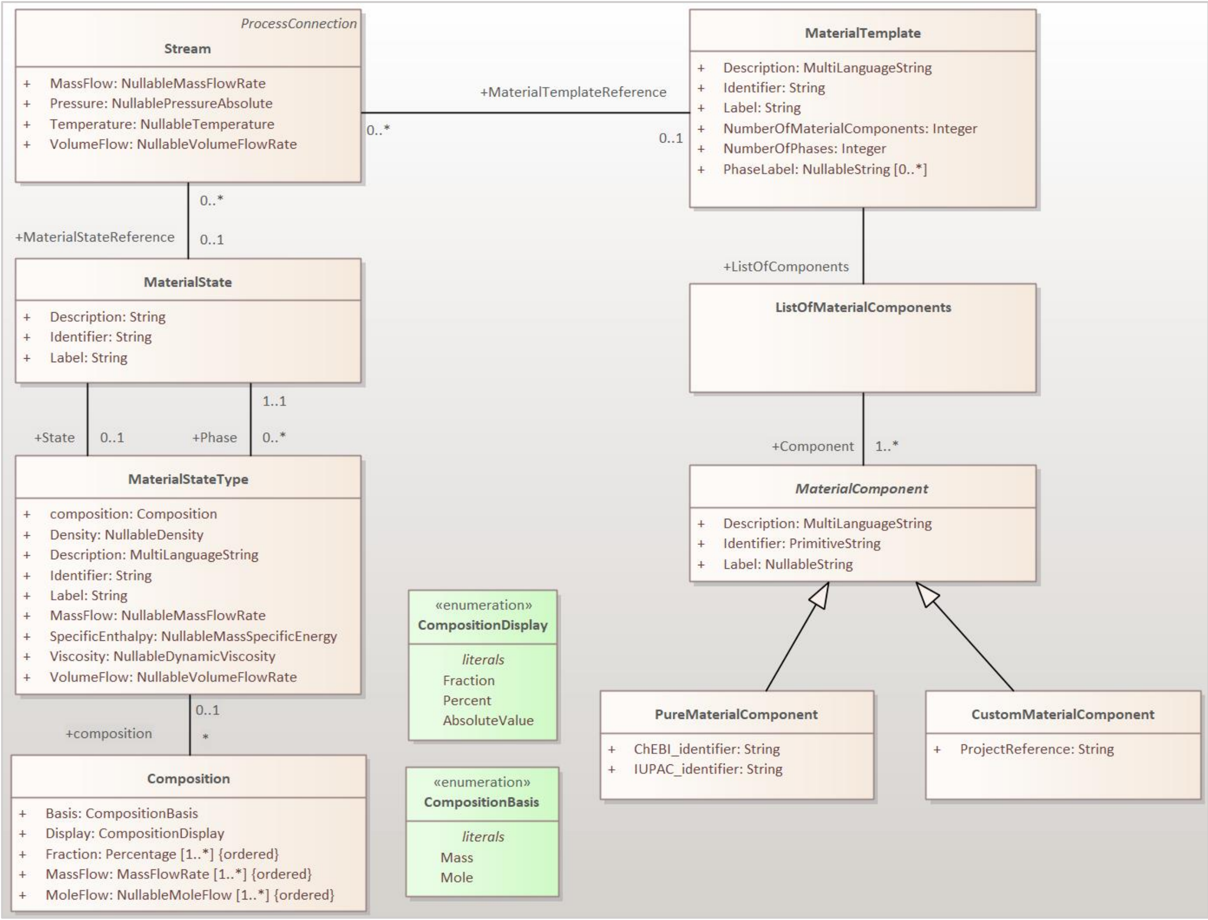
There are three types of flow between blocks:

- **Stream**: flows of material.
- **EnergyFlow**: flows of thermal, electrical or mechanical energy. These are modelled by three subclasses: **ThermalEnergyFlow**, **ElectricalEnergyFlow** and **MechanicalEnergyFlow**.
- **InformationFlow**: flows of structured information such as measurements or control signals.

The structure of information in these flows must be able to be modified for each project. This is done by defining template objects that define the structure for a stream, information flow or energy flow.

2.4.1 Material Flows: Streams

The **MaterialTemplate** object defines the structure of **Stream** objects that are connected to **MaterialPort** objects.



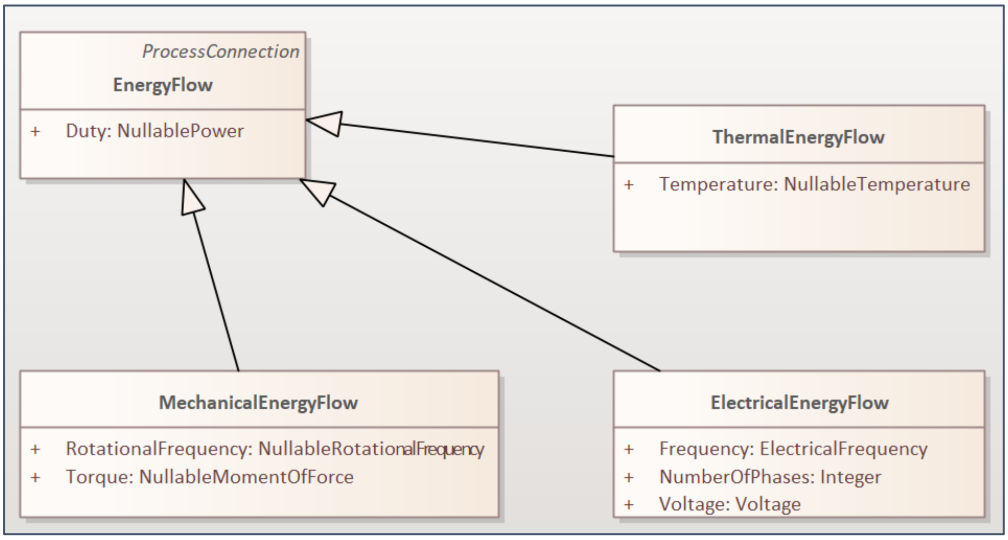
The **MaterialTemplate** defines the number of material components, *NumberOfMaterialComponents*, the number of possible phases, *NumberOfPhases*, and a list of material components, *ListOfComponents*. A **MaterialComponent** object can be either a pure chemical, defined by a **PureMaterialComponent** object, or a project-specific pseudo-component, defined by a **CustomMaterialComponent** object.

These objects provide the information needed to create the header block for a stream table. The columns in the stream table are then provided by **Stream** objects. Each **Stream** object is linked to a **MaterialState** object that contains – via the **MaterialStateType** object - the overall physical properties and **Composition** for the stream. The **Composition** object allows storage of the information as an array of ratio/percentage values (interpreted as mass or mole fraction depending on the value of the *CompositionBasis* parameter), an array of mole flows, and array of mass flows for each material component. The **MaterialState** object can also contain embedded **MaterialStateType** objects that represent individual phases (depending on the *NumberOfPhases* parameter in the **MaterialTemplate**). The order and identity of the components in the **Composition** object is given by the *ListOfComponents* list in the **MaterialTemplate**.

This model has been developed using reference to the OntoCAPE ontology and the CAPE-OPEN standards for simulation. Given this, we believe that the data structures defined here will support simple and standardised data exchange with process simulators.

2.4.2 Energy Flows

As noted above, we model a generic flow of energy by an **EnergyFlow** object. We can also use specialized objects for flows of electrical, thermal and mechanical energy.



2.4.3 Information Flows

An **InformationFlow** can contain any set of **Parameters**. Usually, it will only contain a single parameter, such as a control signal or a scalar quantity.

2.5 Additional Physical Quantities

We use the physical quantities defined in the DEXPI **PhysicalQuantities** package. However, we need to define further quantities in order to be able to represent the stream tables and functional attributes of the process steps.

These quantities are defined using the same pattern as other quantities in the DEXPI **PhysicalQuantities** package. This means that for each quantity, we define a corresponding, abstract **Nullable<>** quantity, a **<>Unit** enumeration for the quantity and a **Null<>** quantity with a singleton instance.

Quantity	Dimension (ISO 80000)	Dimension (Energistics)	IEC62720 CODE	ISO 15926-4 RDL	Typical Unit	Typical Unit (UN code)
Density	$L^{-3}M$	M/L3	UAD106	RDS358874	kg/m³	KMQ
DynamicViscosity	$L^{-1}MT^{-1}$	M/LT	UAD035	RDS2227749	Pa·s	C65
ElectricConductivity	$L^{-3}M^{-1}T^{3}I^{2}$	I2T3/L3M	UAD025	RDS350639	S/m	D10
ElectricCurrent	I	I	UAD039	RDS351089	A	AMP
Energy	$L^{2}MT^{-2}$	L2M/T2	ABT563	RDS351404	kJ	KJO
HeatCapacity	$L^{2}MT^{-2}\Theta^{-1}$	L2M/DT2		RDS382184	kJ/K	
HeatTransferResistance	$M^{-1}T^{3}\Theta$	DT3/M	UAD192	RDS355994	m²·K/W	D19
KinematicViscosity	$L^{2}T^{-1}$	L2/T	UAD070	RDS356399	m²/s	S4
MagneticFieldIntensity	IL^{-1}	I/L		RDS353024	Tesla	
MagneticFluxDensity	IL^{-1}	I/L		RDS353114	A/m	
MassConcentration	ML^{-3}	M/L3	UAD106	RDS4316806585	kg/m³	KMQ
MassSpecificEnergy	$L^{2}T^{-2}$	L2/T2	UAD173	RDS382589	kJ/kg	B42
MassSpecificHeatCapacity	$L^{2}T^{-2}\ \Theta^{-1}$	L2/DT2	UAD112	RDS382229	kJ/(kg·K)	B43
MoleConcentration	$L^{-3}N$	N/L3	UAD005	RDS387989	mol/L	
MoleFlowRate	$T^{-1}N$	N/T	UAD122	RDS351539	kmol/h	
MoleSpecificEnergy	$L^{2}MT^{-2}N^{-1}$	L2M/NT2	UAD124	RDS2229548	kJ/mol	B15
MomentOfForce	$L^{2}MT^{-2}$	L2M/T2	UAD200	RDS380069	N·m	NU
ParticleSize	L	L		RDS403019	mm	MMT
pH	$LOG(L^{-3}N)$			RDS2229993	pH	
SurfaceTension	MT^{-2}	M/T2	UAD184	RDS355769	N/m	4P
ThermalConductivity	$LMT^{-3}\Theta^{-1}$	LM/DT3	UAD190	RDS355949	$W\cdot m^{-1}\cdot K^{-1}$	D53
TimeInterval	T	T		RDS17850935	s	
Velocity	LT^{-1}	L/T		RDS2220032	m/s	

2.6 Sources, Sinks, Emitters and Off-Page Connectors

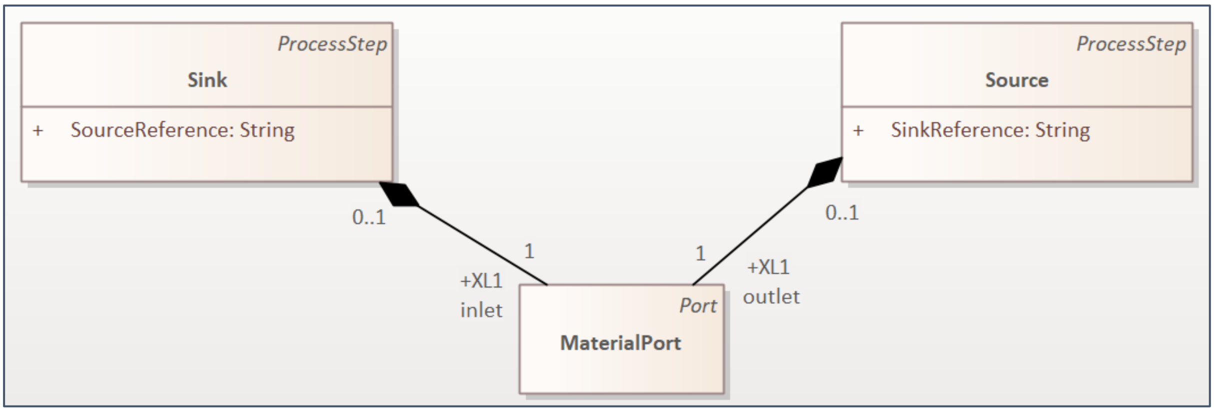
The diagrams we are working with have a limited extent. This means that we need **Source** and **Sink** blocks. These blocks provide the ports needed to define the connections for streams into and out of a diagram or drawing sheet.

An **Emitting** block represents a discharge to the environment. This is useful for accounting for emissions in process design.

These blocks can function as off-page connecters. A source block can contain a reference to a sink block in another diagram. A sink block can contain a reference to a source block in another diagram.

Sources and sinks have no process function. They are simply placeholders for a connection.

It is important to note that sources and sinks are metadata rather than process data. They state that the same material flow is occurring on across a pair of connectors.

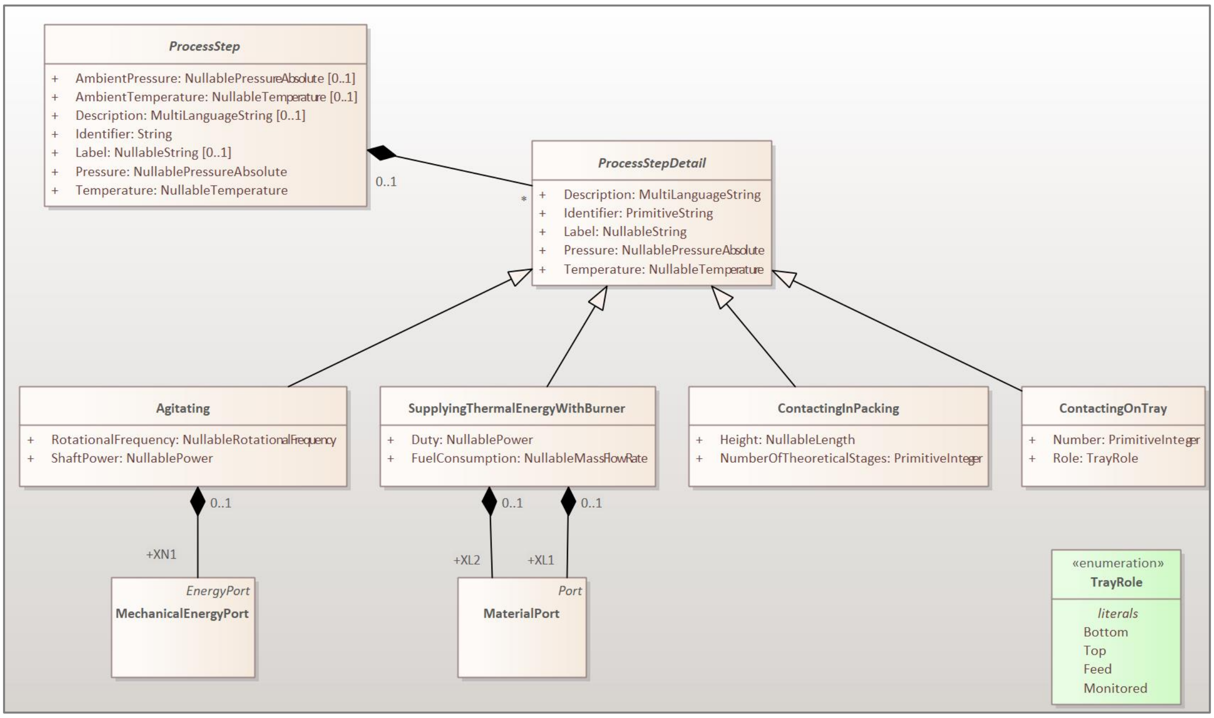


2.7 Details on Process Steps

The **ProcessStepDetail** class is used to represent sub-systems within a **ProcessStep**. Process Flow Diagrams often show details of a process step, such as:

- Compartments in a separation system.
- Packing sections or trays in a distillation or absorption system,
- Heating elements or agitation systems in a reactor or storage vessel.

These can be modelled by inserting a **ProcessStepDetail** object into the parent **ProcessStep**.



Three **ProcessStepDetail** sub-classes are defined. The **Agitate** class represents a stirring or agitating function, whereas the **ContactingOnTray** and **ContactingInPacking** classes represent separation stages in a column-based separation process. The **SupplyingThermalEnergyWithBurner** class represents a burner component in a larger system.

2.8 Modelling Control and Safety Functions on Process Flow Diagrams

2.8.1 Overall Approach

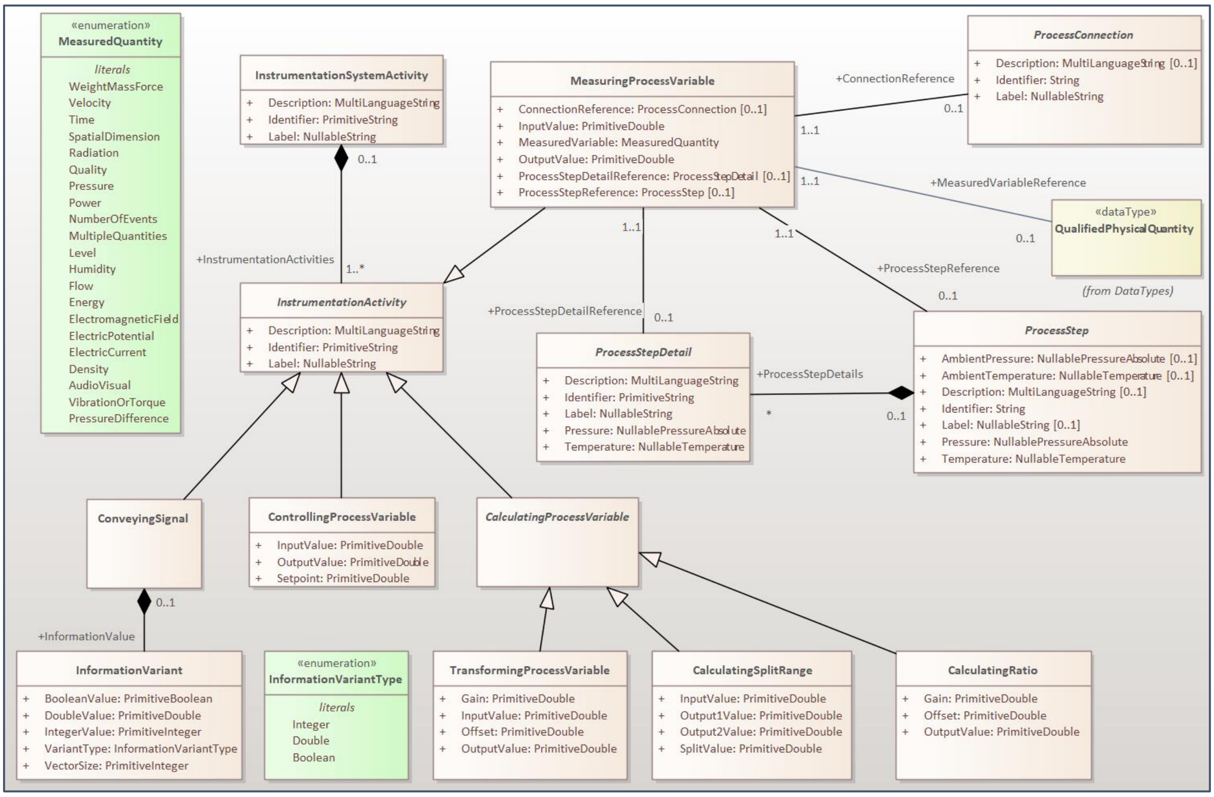
The control and instrumentation information shown on a PFD documents the process control and monitoring philosophy of the process. The information shown consists of:

- Identification of the sensed variables that are used for control and monitoring.
- Indication of the control and instrumentation algorithms and structures used, for example
    - Single-loop control.
    - Cascade control.
    - Ratio control.
- Indication of the final control element(s) used. This will usually be either a **ShuttingOffFlow** or **RegulatingFlow** process step, i.e., a valve, a **Driving** step, i.e., a motor or a **Heating** or **Cooling** step.

2.8.2 Simplified Representation of Control Functions

Note that graphically, the sensing and control functions that would be shown separately on a P&ID are combined into a single symbol. Thus, a FT and FC function will be combined as a single FC graphical element. Presentation is simplified, so that a control loop is represented in the drawing by a single bubble that is connected to the sensing point in the drawing (on a stream or a unit operation) and the final control elements. The bubble represents an **InstrumentationSystemActivity** which will be realized by a DEXPI **ProcessInstrumentationFunction** in the DEXPI *plant* model.

An **InstrumentationSystemActivity** consists of a connected set of **InstrumentationActivity** blocks.



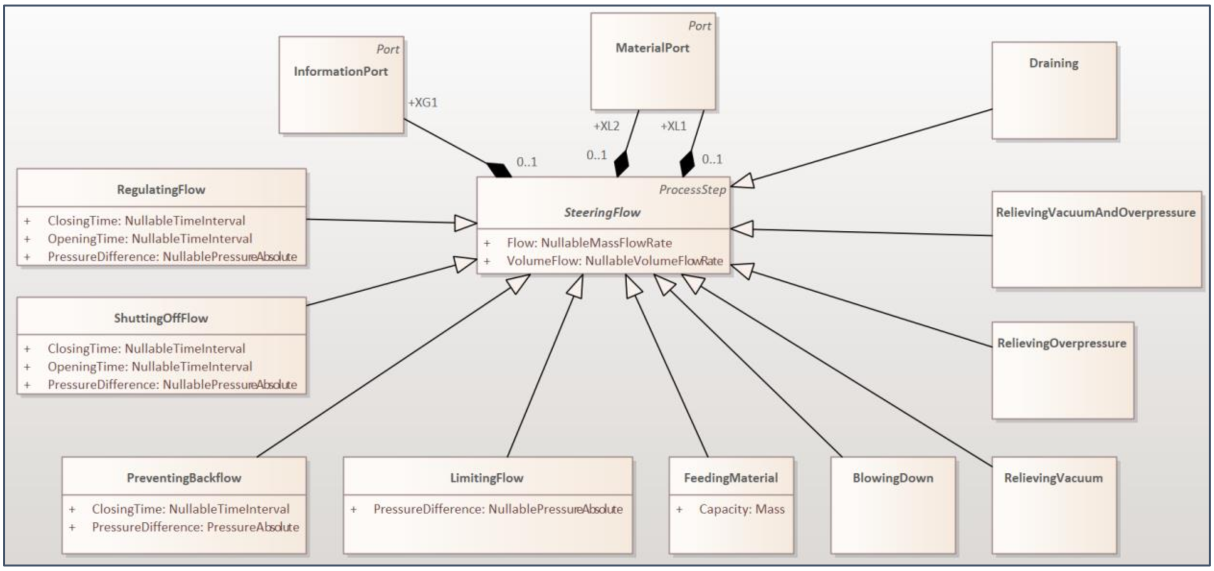
These are standard system blocks, with parameters and ports. We define three sub-types of **InstrumentationActivity**:

- **MeasuringProcessVariable**. This function can be realized by one or more DEXPI **ProcessSignalGeneratingFunction** objects.
- **ControllingProcessVariable**. This function can be realized by one or more DEXPI **ActuatingFunction** or **ActuatingElectricalFunction** objects.
- **CalculatingProcessVariable.**

The DEXPI **SignalConveying** function is represented here by a **ProcessConnection** and its associated **InformationFlow** object.

2.8.3 Flow Control

Flows in the process are manipulated to control the process and ensure safe operation. This means that control valves and shutoff valves are needed in the plant to realize these functions. These functions are indicated in a PFD using a valve icon that corresponds to the equipment that realizes the function.



These functions are modelled by sub-classes of the abstract **SteeringFlow** class. These are:

1. **RegulatingFlow.** Manipulate flow over a range of flowrates. This is the usual function for implementing process control functions.
2. **ShuttingOffFlow**. Turn flow on or off. This is the usual function for implementing safety functions for isolation.
3. **PreventingBackflow**. This is usually a safety function and is realized by a check valve.
4. **LimitingFlow**. This is usually a safety function and is realized by a flow orifice or flow-restriction valve.
5. **FeedingMaterial**. This function describes the feeding of material to or in a process at a defined rate. The material handled is usually a solid.
6. **BlowingDown**. This safety function depressurises the process system for safety or shutdown purposes.
7. **RelievingVacuum**. This safety function protects the process against vacuum.
8. **RelievingOverpressure**. This safety function protects the process against excessive pressure.
9. **RelievingVacuumAndOverpressure.** This safety function protects the process against vacuum or excessive pressure.
10. **Draining.** This function drains liquids in the process to a safe place.

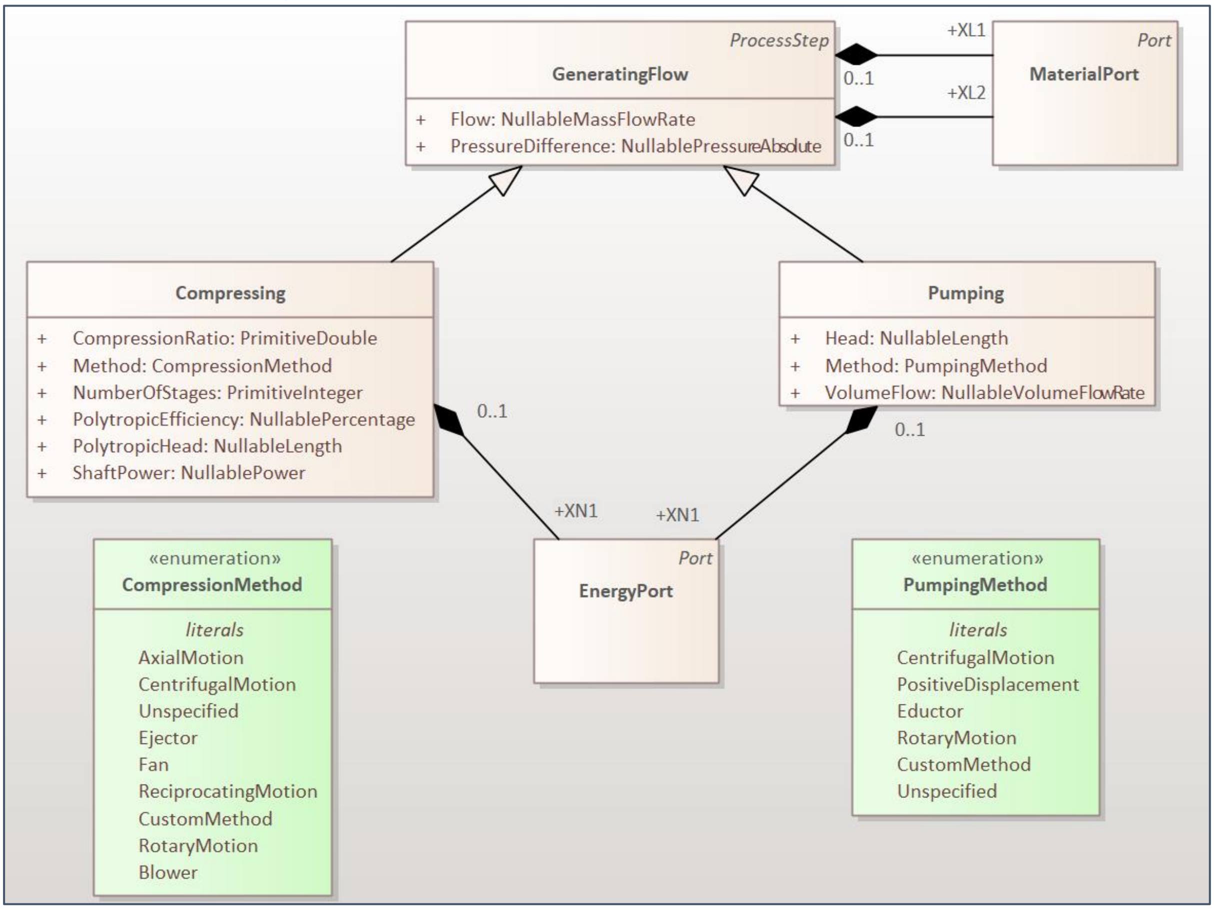
2.9 Flow Generation: Compress and Pump

Process steps for compression of gases and pumping of liquids are subclasses of **GeneratingFlow**. It is a matter of debate whether this activity should be called generate flow or increase pressure. In practice, the designer will choose a **Compressing** or **Pumping** block early in the design process, so the **GeneratingFlow** block will not be used often.

We define two **GeneratingFlow** block types.

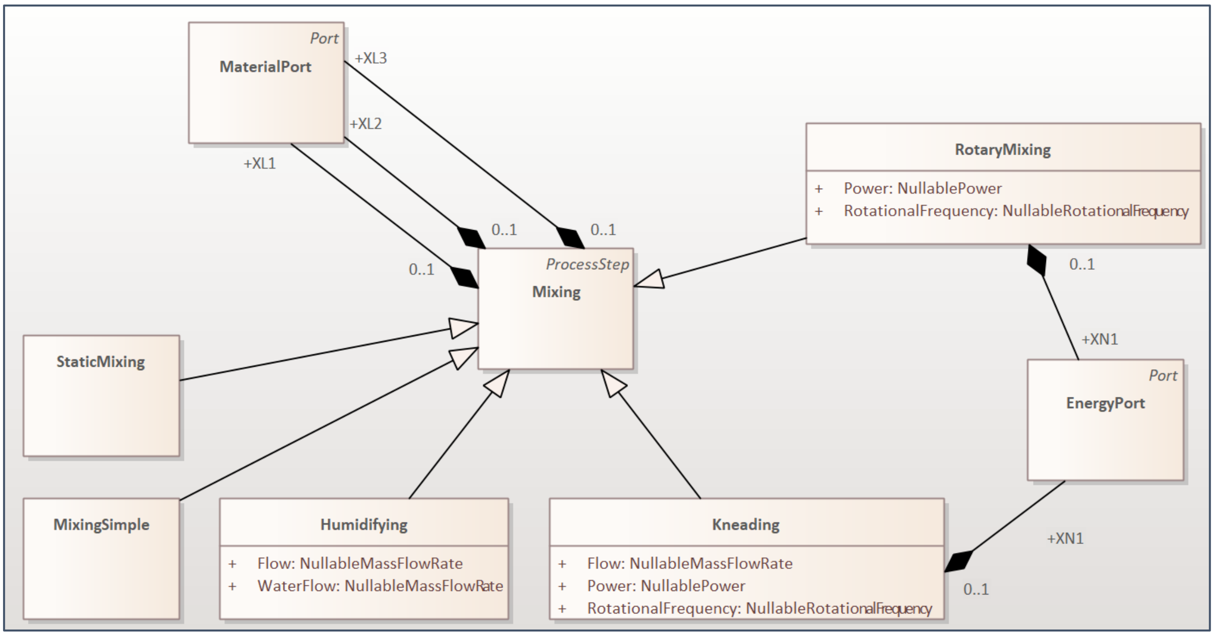
- **Compressing**. This represents a gas compression process. The *Method* attribute allows definition of the compression method:
    - *ReciprocatingMotion*.
    - *Blower.*
    - *Fan*.
    - *Ejector*.
    - *CentrifugalMotion*.
    - *RotaryMotion*.
    - *AxialMotion*.
    - *Unspecified*

- **Pumping.** This represents a liquid pumping process. The *Method* attribute allows definition of the pumping method.
    - *PositiveDisplacement*.
    - *CentrifugalMotion*.
    - *Eductor*.
    - *RotaryMotion*.
    - *Unspecified*.



2.10 Mixing

The following blocks represent mixing processes, where two or more inputs are mixed into a single output.



- **StaticMixing**: mix materials in an apparatus with a static element that obtains a homogeneous material at the outlet.
- **MixingSimple**: a block that can be used to model convergent branches in the detailed flow topology needed for instrumentation and safety design, as shown on the Process Flow diagram. These blocks are represented graphically as a line junction. However these mixing processes will need to be simulated.
- **Humidifying**: this process mixes a material with a water-containing stream in order to attain a desired water content in the outlet.
- **Kneading:** this process mixes two or more solid material inlets to a produce a homogeneous material at the outlet.
- **RotaryMixing**: mix materials with a rotating element to obtain a homogeneous material at the outlet.

2.11 Splitting

A **Splitting** block is used to split an inlet flow into two or more outlet flows. This is needed to model the topology in process flow diagrams. This block has two sub-types:

- **SplittingMaterial**. Split a material flow into two or more outlet flows.
- **SplittingEnergy**. Split an energy flow into two or more outlet flows.

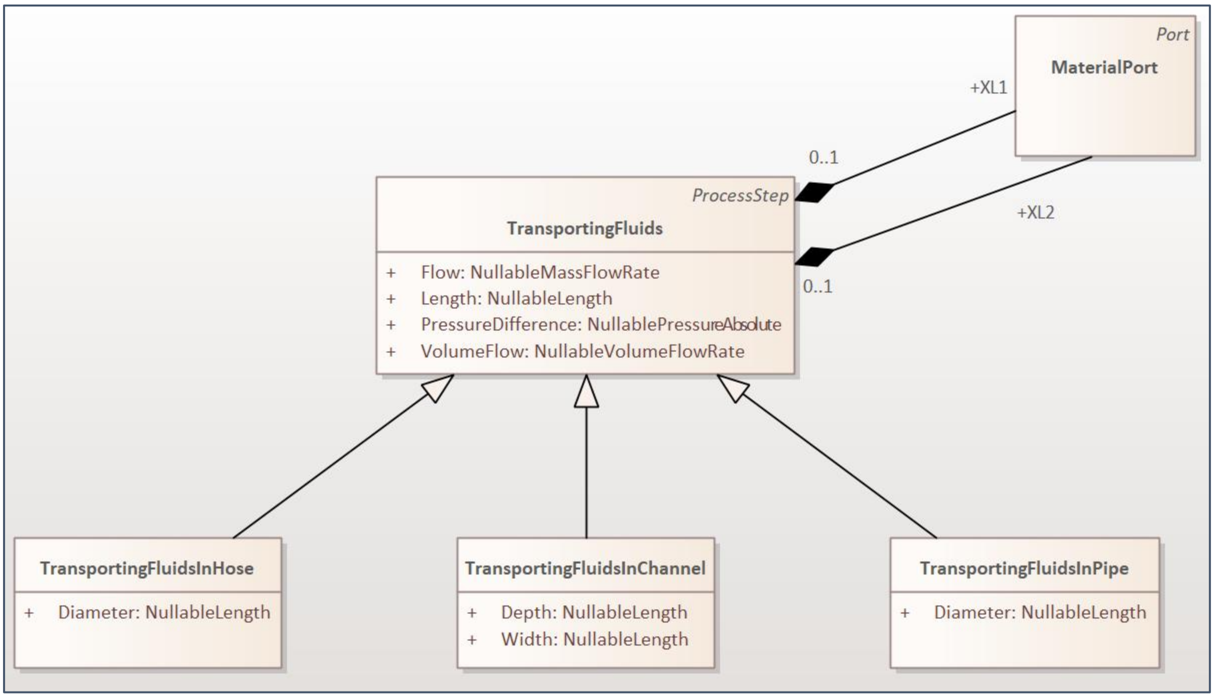
2.12 Transporting of Material and Energy

In process modelling, we only consider transport of material and energy where (1) there is a difference in state between the inlet and the outlet of the system or (2) when process-related calculations are associated with the transporting of the material. The connections between ports in a process model do not represent transport. They describe the state of a material, energy or information at a system boundary.

2.12.1 Transporting of Fluids

Transport of fluids, including slurries, can be modelled by a **TransportingFluids** block, or one of the following sub-classes:

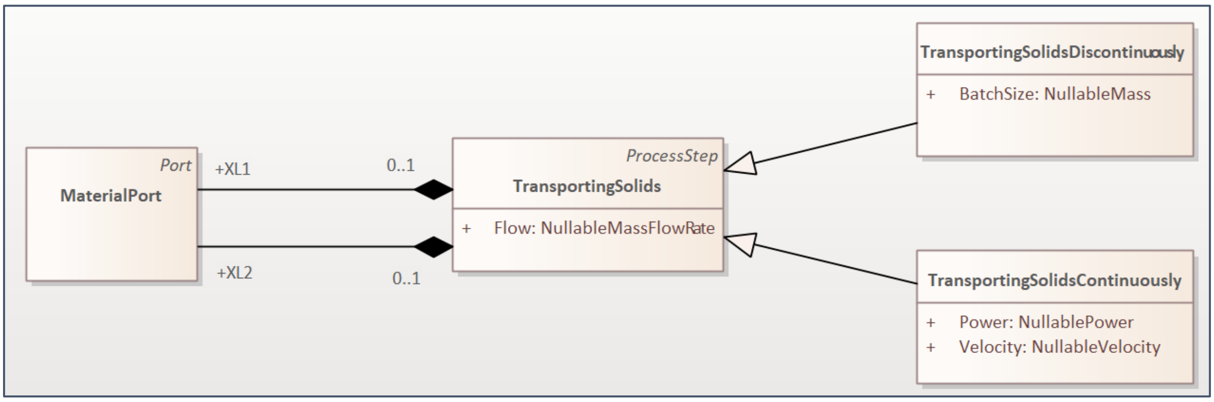
- **TransportingFluidsInPiping**. Transport fluids in a closed, rigid conduit.
- **TransportingFluidsInChannel**. Transport fluids in an open conduit.
- **TransportingFluidsInHose**. Transport fluids in a closed, flexible conduit.



2.12.2 Transporting of Solids

Transport of solids can be modelled by a **TransportingSolids** block, or one of the following sub-classes:

- **TransportingSolidsContinuously**, i.e., by a conveyor or similar apparatus.
- **TransportingSolidsDiscontinously,** i.e., using a vehicle or other batch transport apparatus.

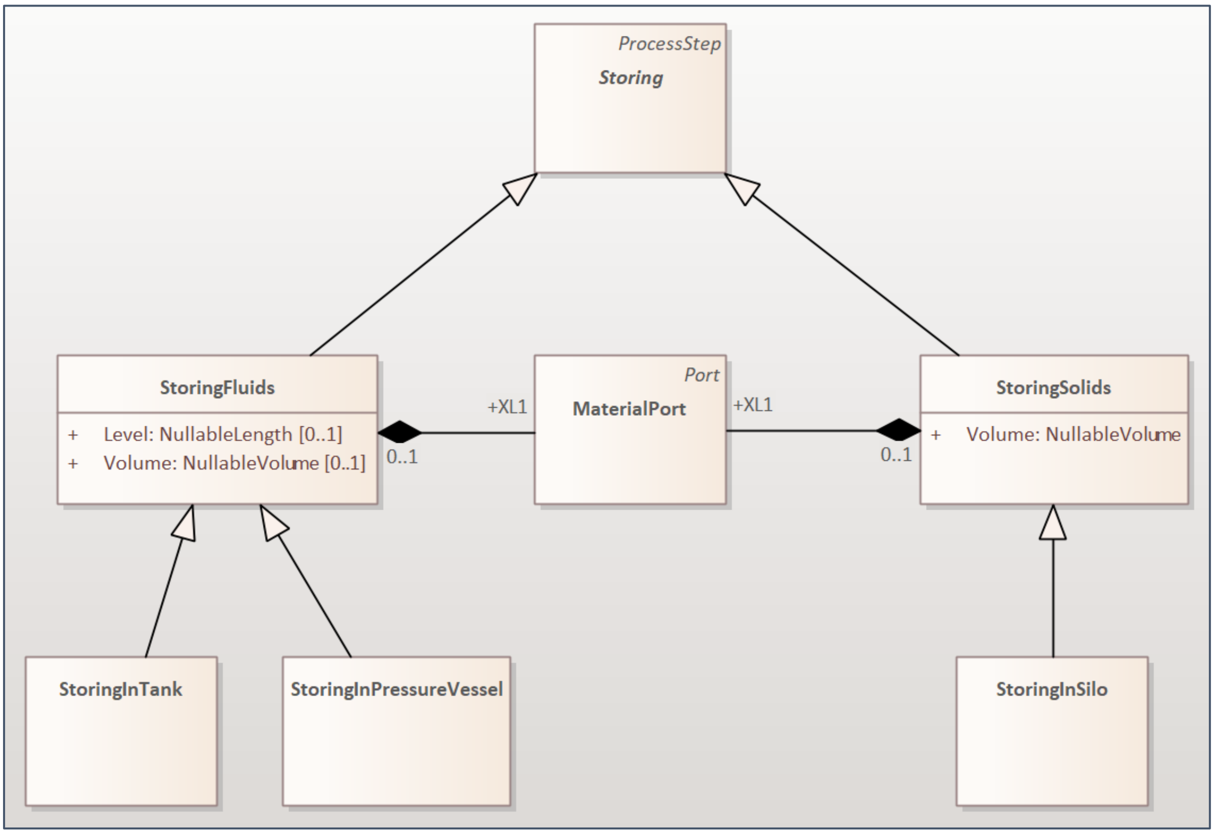


2.12.3 Transporting of Electrical Energy

Modelling of transport of electrical energy is necessary when transmission losses are needed to be accounted for or if the electrical connection has a high capital cost.
This is done using the **TransportingElectricalEnergy** block.

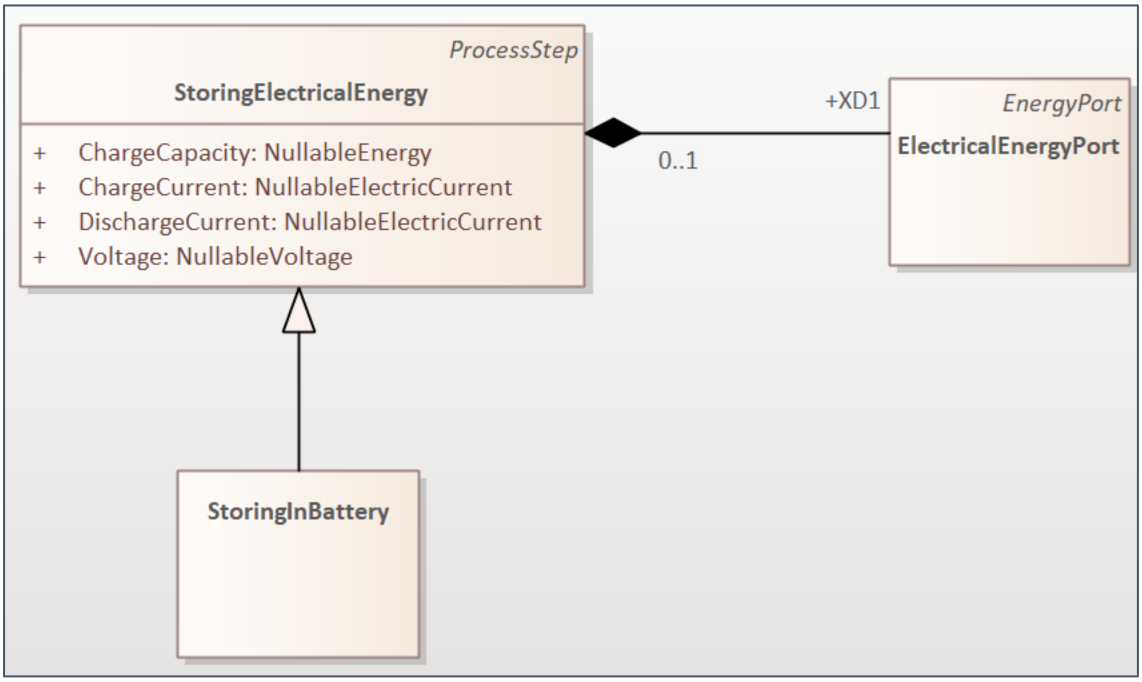
2.13 Storage

2.13.1 Storage of Fluids and Solids



2.13.2 Storage of Electrical Energy

Storage of electrical energy can be modelled by a generic **StoringElectricalEnergy** block, or a more specific **StoringInBattery** block.



2.14 Supplying of Materials and Energy

2.14.1 Supplying of Materials

The **SupplyingFluids** and **SupplyingSolids** blocks may be used to model a supply of materials. These blocks can be used instead of a **Source** block to model the starting point of a process.

2.14.2 Supply of Electrical Energy

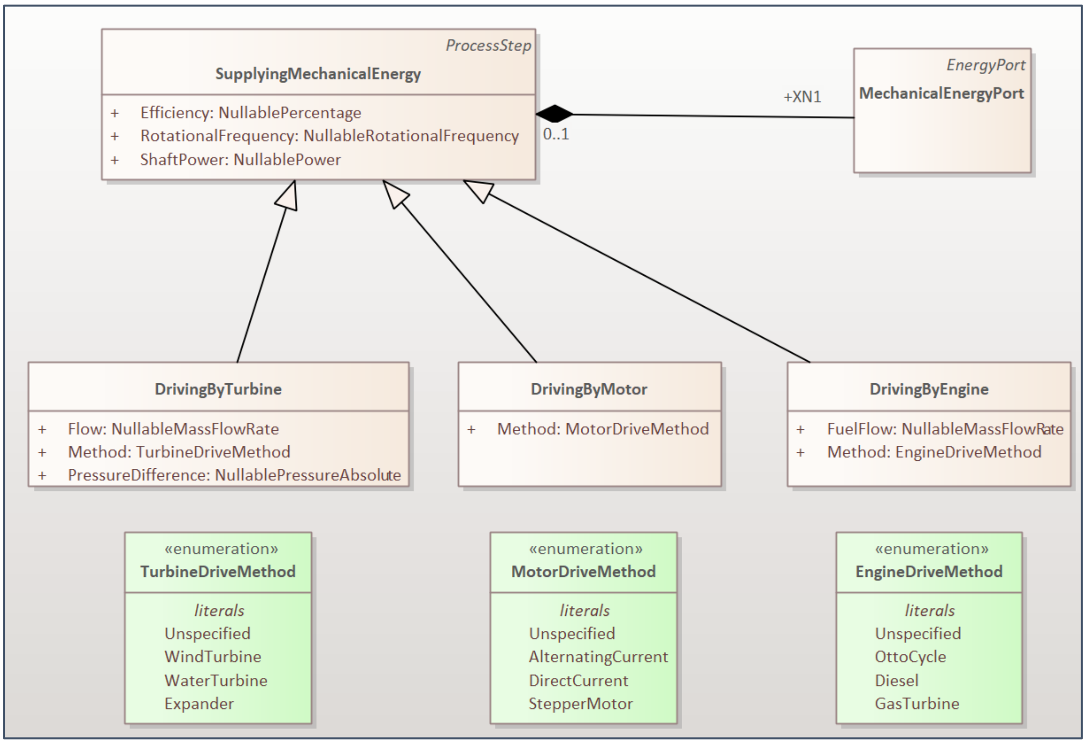
The **SupplyingElectricalEnergy** can be used to model a source of electricity. This function can be further differentiated into:

- **GeneratingACPower**, i.e., an AC generation process.
- **GeneratingDCPower**, i.e., a DC generation process.
- **GeneratingInFuelCell**, i.e., a fuel cell electricity source.
- **GeneratingCustom**, a custom source of electricity.

2.14.3 Supply of Mechanical Energy

The **SupplyingMechanicalEnergy** block models a generic source of mechanical energy. We further differentiate these into:

- **DrivingByMotor**, where the *Method* parameter specifies the type of motor:
  - *AlternatingCurrent*, i.e. one- or three-phase AC motor
  - *Stepper*
  - *DirectCurrent*
  - *Unspecifed*
- **DrivingByTurbine**, where the *Method* parameter specifies the type of turbine process:
  - *WindTubine*
  - *HydraulicTurbine*
  - *Expander*
  - *Unspecified*
- **DrivingByEngine**, where the *Method* parameter specifies the type of engine:
  - *OttoCycle*, i.e., a petrol-driven engine
  - *DieselEngine*
  - *GasTurbine*
  - *Unspecifed*



2.15 Supplying and Exchange of Thermal Energy: Flaring, Heat Exchange, Heating and Cooling

2.15.1 Overview

Systems for heat exchange, heating and cooling can be represented by three process step blocks:

- **SupplyingThermalEnergy**. This models the heating of materials, usually with an external or utility heating source.
- **RemovingThermalEnergy**. This models the cooling of materials, usually with an external or utility cooling medium.
- **ExchangingThermalEnergy**. This models the exchange of energy with another process stream. It can also be used to model exchange with heating or cooling utility streams.

2.15.2 Heat Exchange System Types

All thermal energy blocks have a *Method* parameter. This can be set to indicate the type of heat exchange system used. Options are:

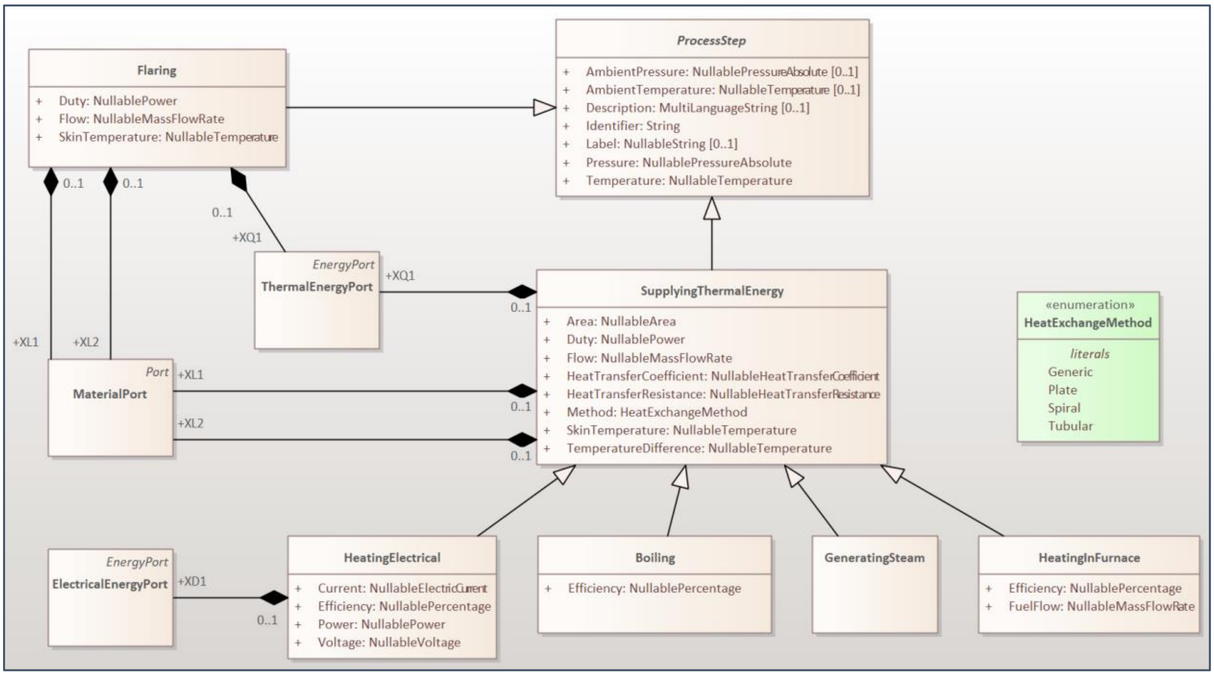
- *Generic*
- *Plate*
- *Spiral*
- *Tubular*

This parameter can be used to determine the type of symbol used to represent the block.

2.15.3 Flaring and Supplying Thermal Energy (Heating)

The **Flaring** block represents a process where a material is burnt, usually for disposal.
The **SupplyingThermalEnergy** block has four sub-types.

- **HeatingElectrical**. This represents a process where a material is heated up by electrical energy.
- **Boiling**. This represents boiling of a fluid.
- **GeneratingSteam**. This represents production of steam.
- **HeatingInFurnace**. This represents heating in a fired heater or furnace.



2.15.4 Remove Thermal Energy: Cooling

The **RemovingThermalEnergy** block has a single sub-type:

- **Cooling**. This represents cooling by contact with an external or utility cooling fluid.

2.15.5 Exchange Thermal Energy

The **ExchangingThermalEnergy** block is used to represent all heat exchange processes. The primary purpose of this block is to model processes where heat is exchanged between streams in a process. A modeller can also choose to use this block for cooling and heating processes.

2.16 Separation

2.16.1 Overview

The classification of separation processes follows that given in DIN6779, with some modification.

This means that we identify the following top-level separation processes:

- **SeparatingByPhaseSeparation**.
- **SeparatingByThermalProcess**.
- **SeparatingByFiltering**.
- **SeparatingByElectromagneticForce**.
- **SeparatingByPhysicalProcess**.

2.16.2 Separating by Phase Separation

The **SeparatingByPhaseSeparation** block has the following sub-types:

- **SeparatingByGravity**. This models gravitational separation processes, such as settling.
- **SeparatingByCentrifugalForce**. This models centrifugal processes with externally supplied motion.
- **SeparatingByCyclonicMotion**. This models cyclonic processes, where centrifugal motion is generated by the flow of the fluid itself.
- **SeparatingByGasLiquidSeparation**. This models phase separator and knock-out drum processes.
- **SeparatingByScrubbing**. This models processes where a fluid is contacted with a counter-flowing fluid.
- **SeparatingByFlash**. This models processes where a fluid is flashed in order to separate phases.

2.16.3 Separation by Thermal Processes

The **SeparatingByThermalProcess** block has the following sub-types:

- **Drying**. This models processes where a solid material is dried.
- **Distilling**. This models a generic distillation process.
- **Evaporating**. This models an evaporation process, where a fluid or mixture is concentrated through the application of heat.
- **StrippingDistilling**. This models a stripping distillation process.
- **VacuumDistilling**. This models a vacuum distillation process.
- **StabilizingDistilling**. This models a stabilizing distillation process.

2.16.4 Separation by Mechanical Process

The **SeparatingMechanically** block represents mechanical separation processes. It has the following sub-types:

- **Skimming**. This models processes where a material is skimmed off a surface.
- **Filtering**. This models processes where materials are separated by encountering a permeable barrier.
- **Sieving**. This models processes where solid materials are separated according to particle size or shape by passing through a screen or sieve.

2.16.5 Separating by Electromagnetic Force

The **SeparatingByElectromagneticForce** block represents electrostatic and magnetic separation processes. It has the following sub-types:

- **SeparatingByElectrostaticForce**. This models electrostatic separation processes.
- **SeparatingByMagneticForce**. This models magnetic separation processes.

2.16.6 Separating by Physical Processes

The **SeparatingByPhysicalProcess** block represents separation processes that use a physical principle, such as mass transfer, ion exchange or surface tension, as the basis for separation. It has the following sub-types:

- **SeparatingByIonExchange**. This models ion exchange separation processes.
- **Absorbing**. This models absorption/desorption processes.
- **Adsorbing**. This models adsorption/desorption processes.
- **SeparatingByContact**. This models separation by contact or liquid-liquid extraction.
- **SeparatingBySurfaceTension**. This models separation processes based on surface tension, such as froth flotation.

2.17 Solids Processing

Solids processing processes are organized following the classification proposed by DIN6779. There are three top-level processes:

- **ReducingParticleSize**
- **IncreasingParticleSize**
- **FormingSolidMaterial**

2.17.1 Reducing Particle Size

The **ReducingParticleSize** block represents processes that reduce the size of particles. It has the following sub-types.

- **Cutting**
- **Milling**
- **Crushing**
- **Grinding**
- **CustomMiling**

2.17.2 Increasing Particle Size

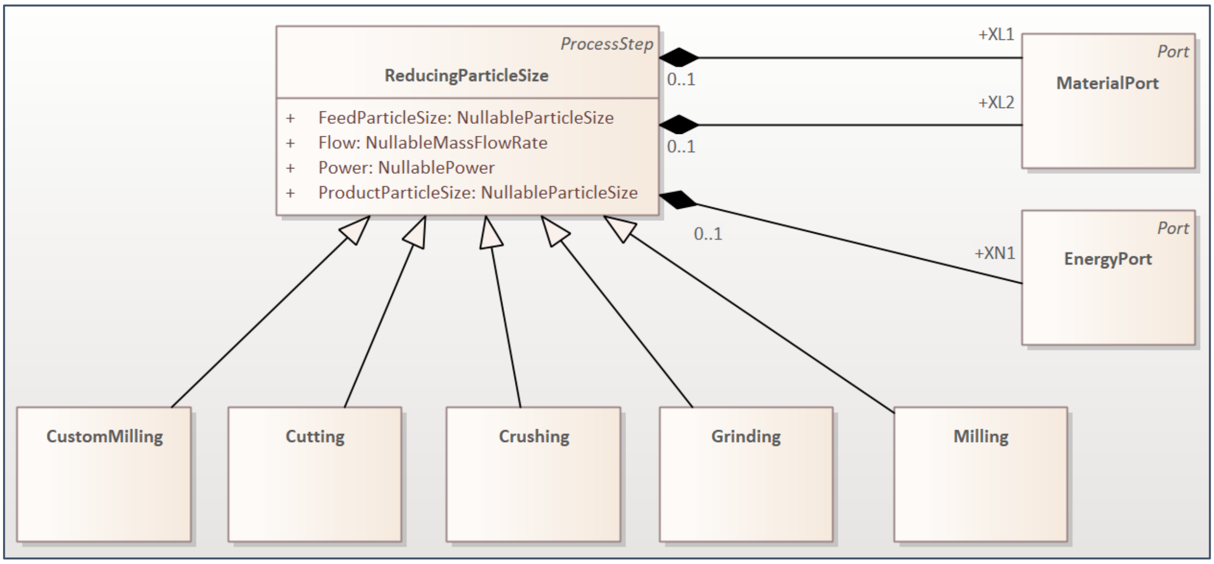
The **IncreasingParticleSize** block represents processes that increase the size of particles. It has the following sub-types.

- **Agglomerating**. This models processes that increase size through contacting particles.
- **Flocculating**. This models processes that increase size through flocculation.
- **Crystallizing**. This models processes that increase size through crystallization.

2.17.3 Forming Solid Material

The **FormingSolidMaterial** block represents processes that produce solid matter with a specific size and shape. It has the following sub-types.

- **Pelletizing**. This models processes that create particles with a specified size and shape.
- **Extruding**. This models processes that create formed shapes by extrusion.



2.18 Chemical Reactions

The **ReactingChemicals** block is used to model chemical reaction processes. The *Method* parameter allows the modeller to specify the type of reaction process.

- *Unspecified*
- *Tubular*
- *Tank*
- *PackedBed*
- *FluidizedBed*

2.19 Packaging

The **Packaging** block models processes where a material is packaged for transport or sale.

3 Parameters

3.1 Introduction

The process package proposed here provides a data model that can be used to organize and present the process design data and constraints for a facility. This means that we need a richer set of parameters than is provided in the existing version of DEXPI P&ID. We also note that each project will have its own standards and practices as to which parameters are displayed and how they are displayed.

Our ambition in the DEXPI Process specification is to propose a set of parameters that describe and constrain the *process* behaviour and design. Note that these parameters are not modified by terms such as “design”, “operating”, “nominal”, “maximum” or “lower limit”. This is done so that the parameter sets proposed for blocks are generally usable and contain parameters with a basis in engineering and physics.

The default option in DEXPI Process models and documents is therefore that the block parameters, material state and other flow objects contain the design values for the case that is to be presented in the BFD and PFD. Other values may be added to the model by adding qualified parameter values that refer to a block parameter or parameter in a flow object.

3.2 Qualified Parameters

Each parameter value includes a documentation block that is used to specify the meaning and purpose of the specific value. This block contains three attributes.

1. A **Mode** (what POSC Caesar PLM RDL calls Scope)
    - Design
    - Allowable – see the Absolute or Continuous Regularity in POSC Caesar PLM RDL).
    - Incidental
    - Expected
    - Operating
    - Test
2. A **Provenance** (what POSC Caesar PLM RDL calls Provenance):
    - Set
    - Specified
    - Estimated
    - Calculated
    - Observed
3. A **Range**
    - Actual
    - Average
    - Nominal
    - Normal
    - Upper Limit
    - Lower Limit

The parameter can also contain a Description, Label and URIs for the data source, provenance information and the RDL value that defines the qualified parameter.

3.3 Parameter Dictionary

The DEXPI process package uses the following parameter definitions. We have tried to align these definitions with those provided by ISO15926-4 and the IEC CDD. These cross-references are given in the table shown in Appendix 1.

A direct correlation with CFIHOS parameters is not possible, since the CFIHOS parameters are all qualified values. There is not a *Temperature* for a unit, rather a “tube side normal operating inlet temperature”.

3.4 Block Parameters

Every **ProcessStep** block defines the following block parameters:

- *Identifier*, a **String** that contains the unique local identifier of this block in the model.
- *Label,* a **String** that contains the identifier used for the block in the PFD or BFD.
- *Description*, a **String** that contains a description of the block.
- *Temperature,* a **Temperature** that defines the temperature at which the system represented by the block is designed to operate.
- *Pressure,* a **PressureAbsolute** that defines the pressure at which the system represented by the block is designed to operate.
- *AmbientTemperature,* a **Temperature** that defines the ambient temperature at which the system represented by the block is designed to operate.
- *AmbientPressure,* a **PressureAbsolute** that defines the ambient pressure at which the system represented by the block is designed to operate.

Each block that inherits from **ProcessStep** will then have additional block parameters. These characterise the activity carried out in the block.

4 Reference Diagrams

4.1 Overview

This model has been developed and validated using four reference diagrams. Three of these are defined in ISO10628-1 [1].

1. A block flow diagram.
2. A simplified process flow diagram, without control elements.
3. A detailed process flow diagram, with a stream table and control elements.
4. A simple process flow diagram derived from the DEXPI reference P&ID.

4.2 Reference Block Flow Diagram from ISO10628

The reference block flow diagram is shown in Figure 1.

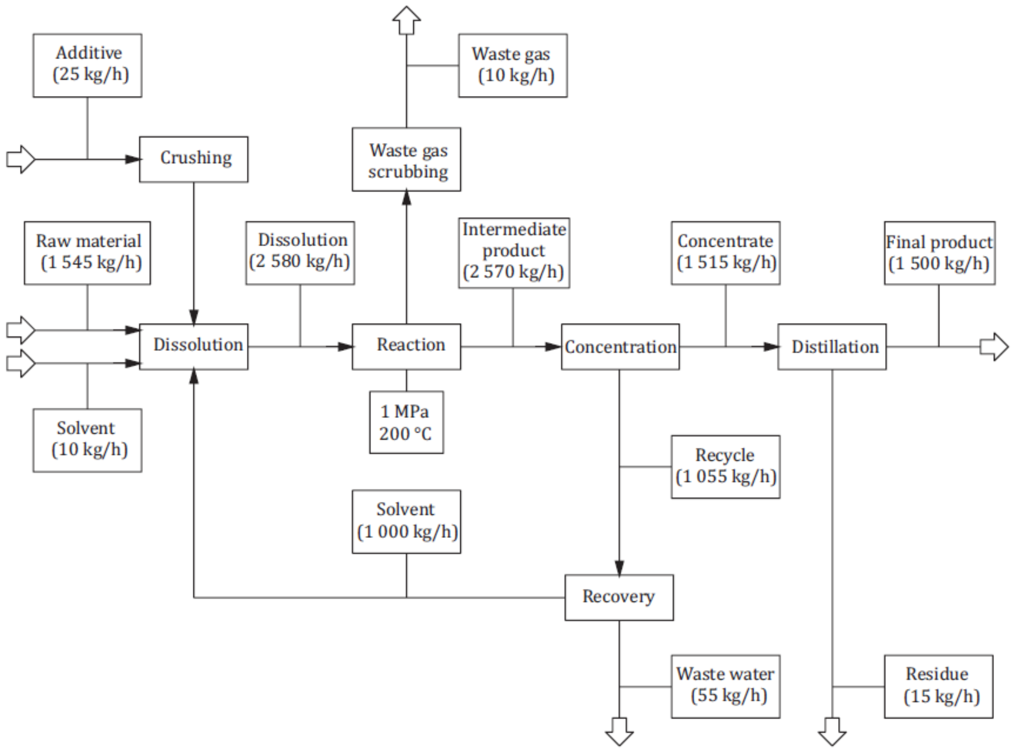
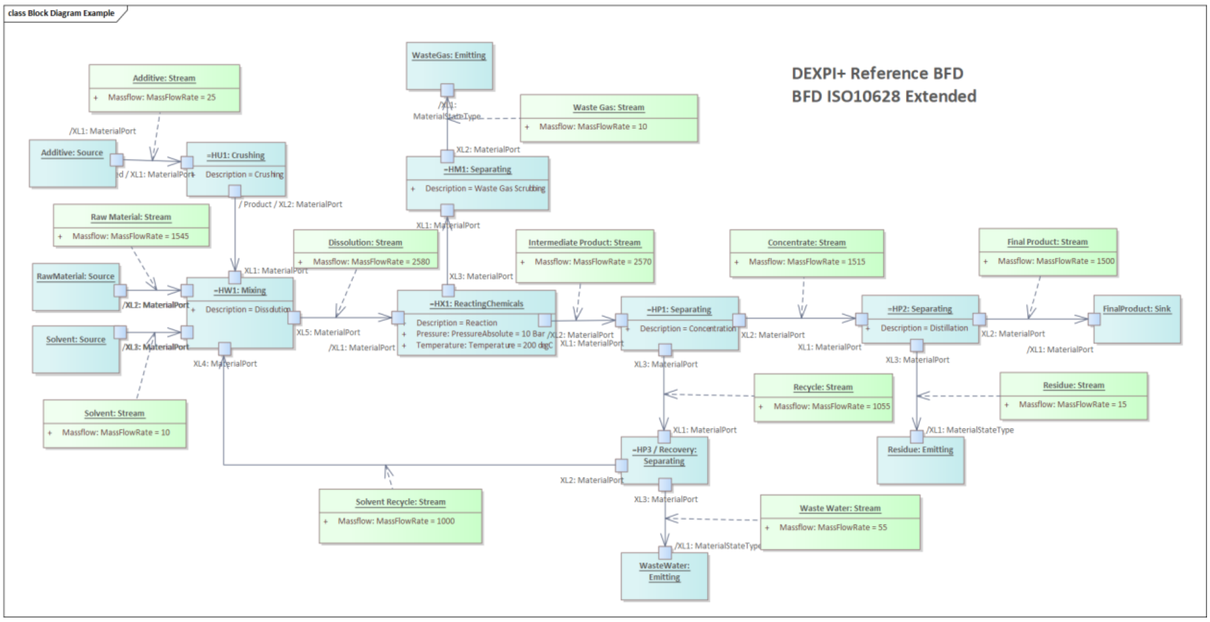


Figure 1. Reference Block Flow Diagram from ISO10628-1.

This diagram contains the following information elements.

- Blocks that represent the processing steps or activities in the process.
- Arrows that represent the flow of material to and from processing steps.
- Block arrows that represent sources and sinks for material flow. These may function as off-page connectors.
- Labels that identify the material flows and specify properties of these flows (in this case, the mass flow rate).
- Labels that specify chosen parameters for a processing activity. Here, the *Reaction* step will operate at 1 MPa and 200 °C.
- Lines that indicate the arrow or block to which a label refers. These should have a different line weight, colour or style to the arrows that are used to represent material flow.

The model is shown in the following figure.



4.3 Reference Process Flow Diagrams from ISO10628

ISO 10628 presents two sample flow diagrams for the same process. The first is a simplified diagram, showing the processing steps. It is shown in Figure 2. The second is a detailed diagram, with a table of stream properties, a presentation of control strategy and more detail of the waste air system. It is shown in Figure 3.

4.3.1 Simple Process Flow Diagram

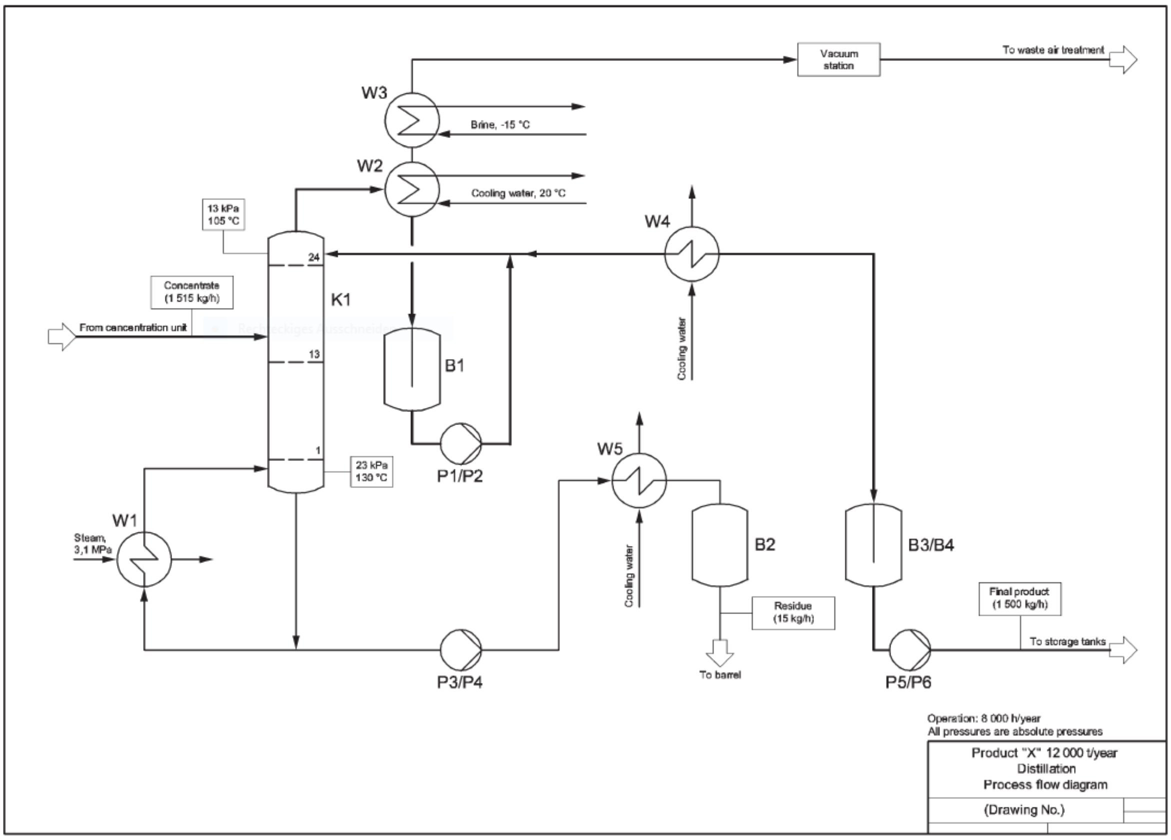


Figure 2. Simple Process Flow Diagram from ISO10628-1.

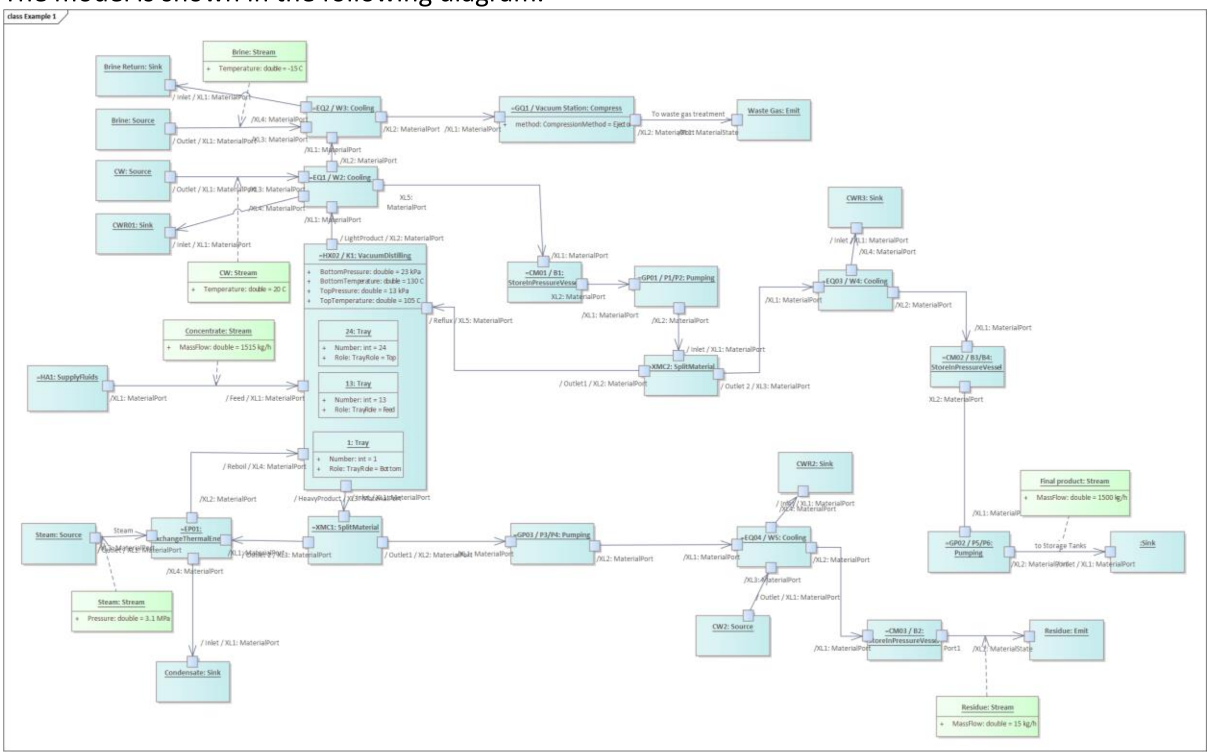
This diagram contains the following elements:

- Symbols that represent processing steps.
- Arrows that represent the flow of material between the processing steps.
- Block arrows that represent sources and sinks of material. These may represent off page connectors. These arrows can have labels, e.g. “To waste treatment”, “From concentration unit”.
- Arrows that represent the flow of heating or cooling media (steam and water) to heat exchange steps.
- Labels that identify each processing step.
- Labels that identity material flows and provide information about the flow.
- Labels that provide information about the parameters of specific processing steps.
- Lines that connect labels to the symbol or arrow to which they refer.
- Notes about the diagram and data model.
- A title block.

Note that the symbol for the distillation process step shows internal details of the system, namely the number of trays and the location of the feed tray. Similarly, the icons for systems B1 and B3/B4 show an inlet piping system to ensure submerged feed of material.

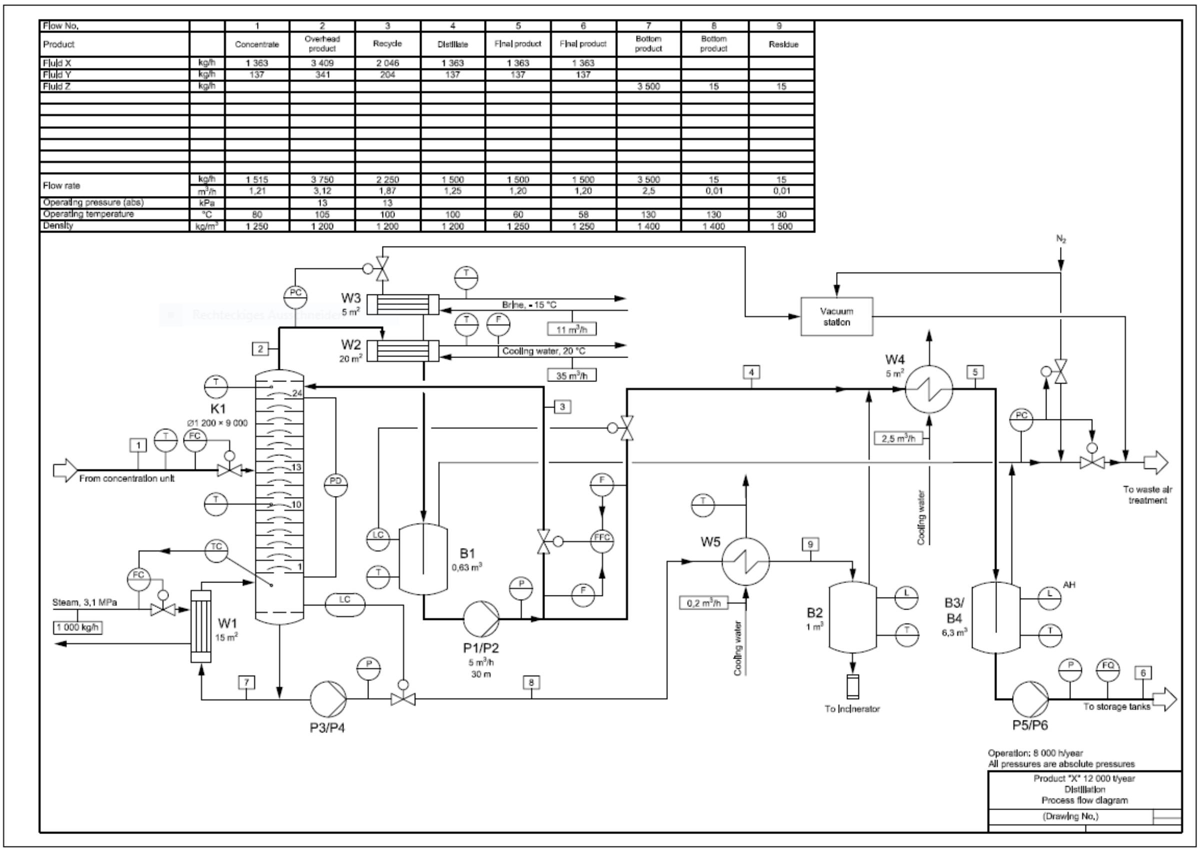
Note that there is an error in the above diagram. The arrow out of W4 towards K1 is in the wrong direction. It must be reversed, showing towards W4.

The model is shown in the following diagram.



4.3.2 Detailed Process Flow Diagram

The detailed variant of the process flow diagram is shown in Figure 3.



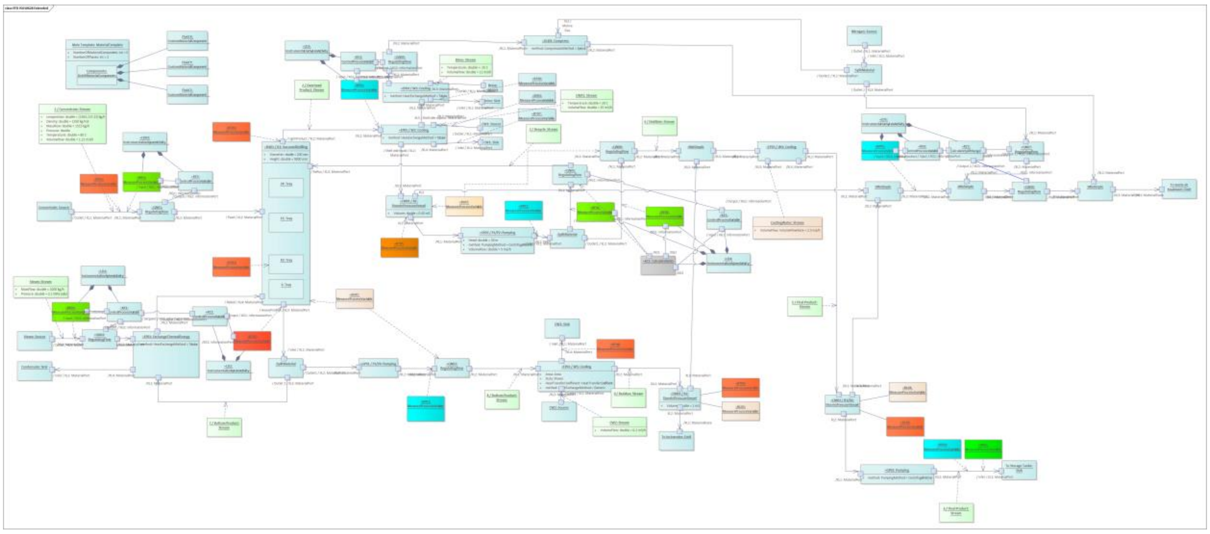
Flow No.		1	2	3	4	5	6	7	8	9
Product		Concentrate	Overhead product	Recycle	Distillate	Final product	Final product	Bottom product	Bottom product	Residue
Fluid X	kg/h	1 363	3 409	2 046	1 363	1 363	1 363			
Fluid Y	kg/h	137	341	204	137	137	137			
Fluid Z	kg/h							3 500	15	15
Flow rate	kg/h	1 515	3 750	2 250	1 500	1 500	1 500	3 500	15	15
	m^3/h	1,21	3,12	1,87	1,25	1,20	1,20	2,5	0,01	0,01
Operating pressure (abs)	kPa		13	13						
Operating temperature	°C	80	105	100	100	60	58	130	130	30
Density	kg/m^3	1 250	1 200	1 200	1 200	1 250	1 250	1 400	1 400	1 500

Figure 3. Detailed Process Flow Diagram from ISO10628-1

This diagram contains the same elements as the simple version, with the following additional elements:

- The basic control and monitoring strategy for the plant.
  - Bubbles represent measurements, controllers and calculation elements.
  - These bubbles can be labelled to show control function and alarms.
  - Control valves (flow control systems) are used to manipulate material flows.
- A stream table of material properties, flows and compositions is presented.
- The stream table refers to labels on the material flows in the diagram.
- Detailed symbols are used for the heat exchange systems W1, W2, W3 and the distillation system K1. These show internal structure for the systems.

The resulting model is shown in the following diagram.



4.4 DEXPI Reference PFD

The final example is based on the conversion of an existing DEXPI reference P&ID, shown in Figure 4.

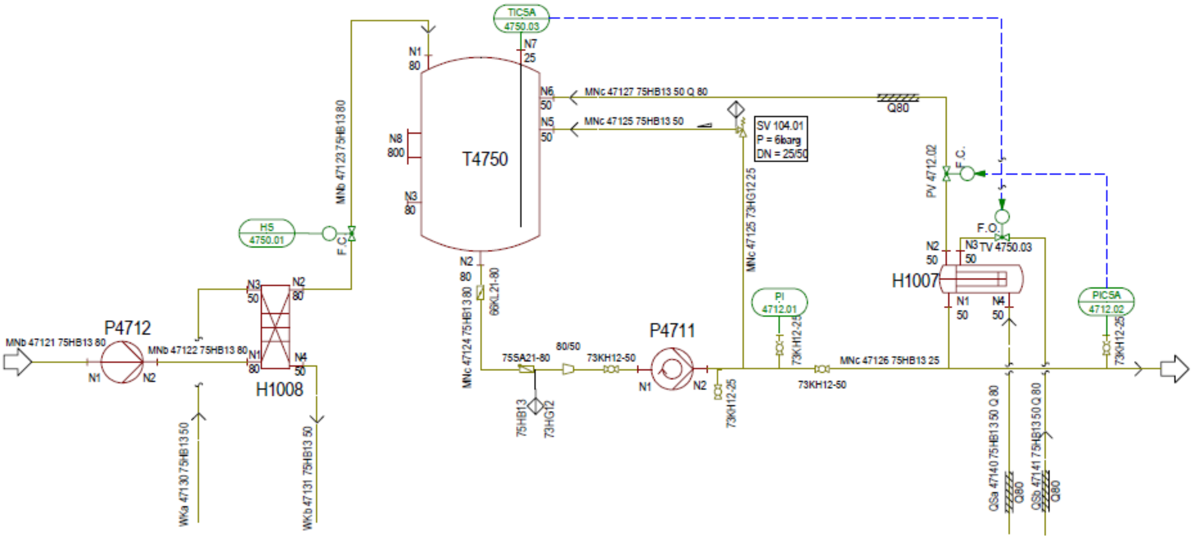


Figure 4. Demonstration P&ID used to generate Demonstration PFD.

The demonstration PFD was created by removing all piping details and labels. The safety valve was also omitted. A stream table and title block was added. The resulting diagram is shown in Figure 5.

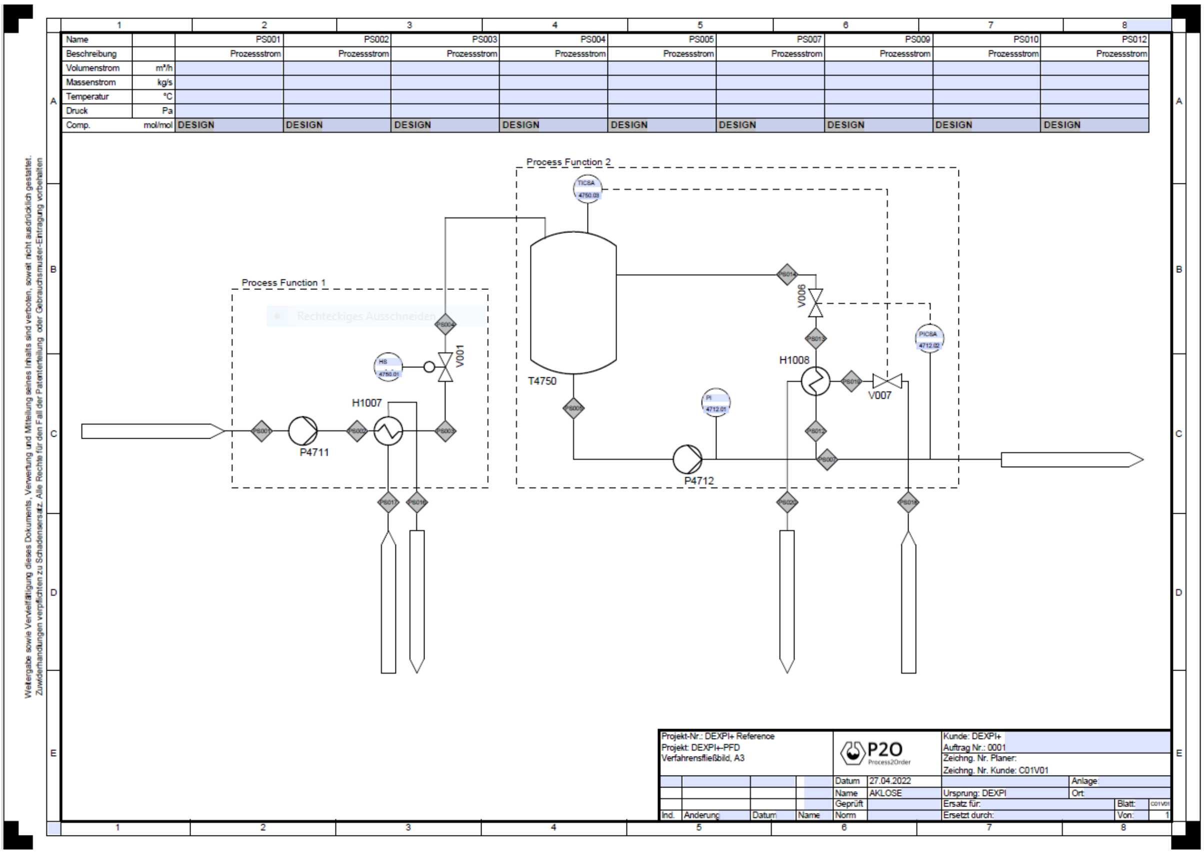
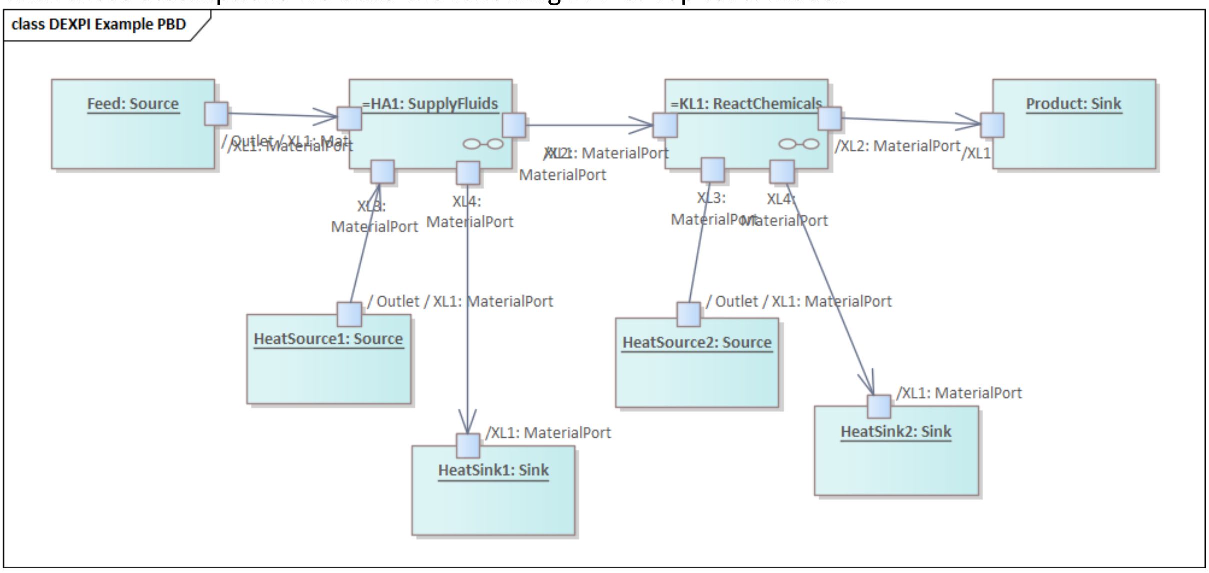


Figure 5. Demonstration PFD.

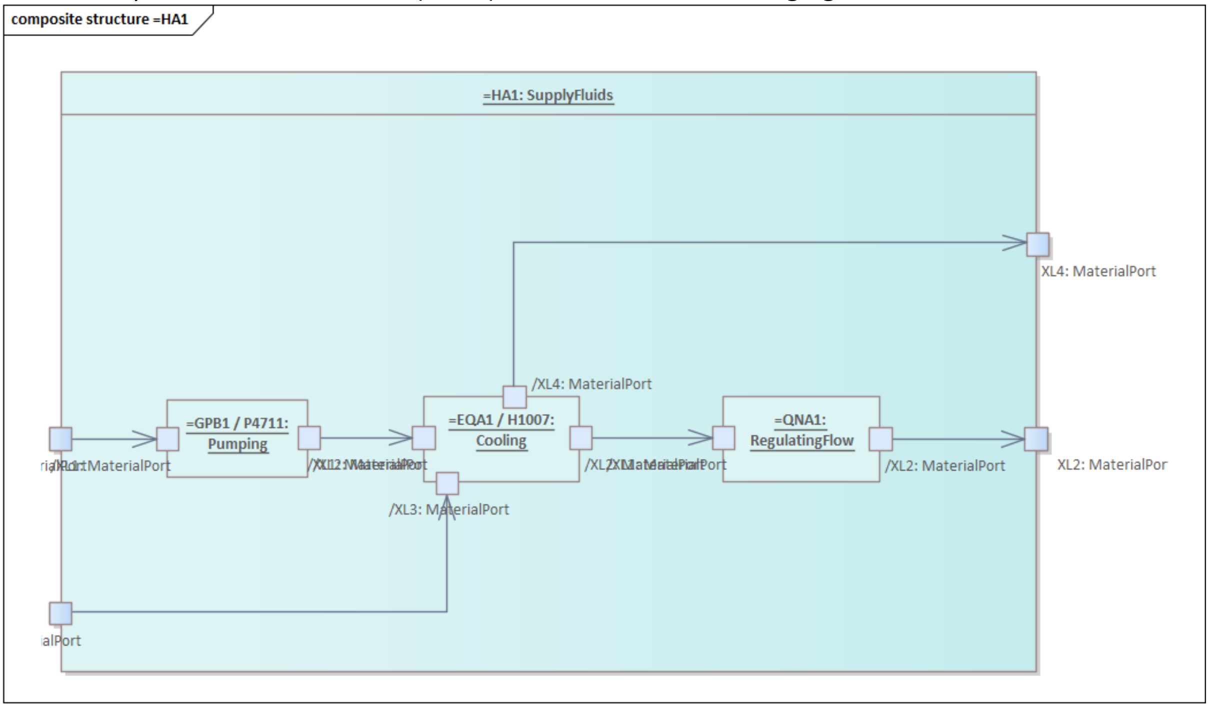
This diagram also shows a way of dividing the process into two blocks that could be shown in a process block diagram.

We model the system in two steps. We first identify the blocks that can be put on a BFD. We do not have enough information as to the primary purpose of each block, so an arbitrary choice has been made. The first block has a purpose to supply fluids at a specified rate and temperature. The second block is chosen to be a reactor system.

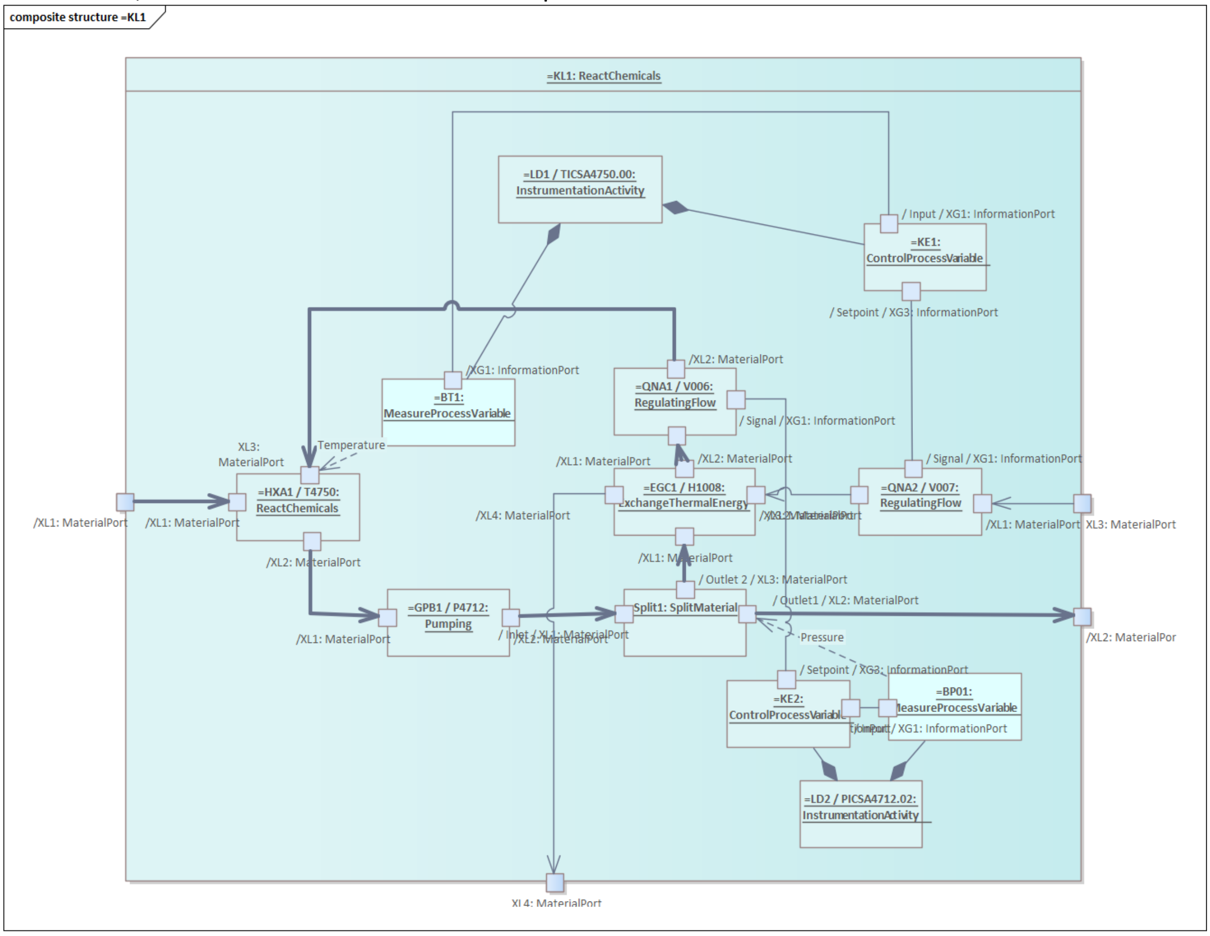
With these assumptions we build the following BFD or top-level model.



The PFD is then modelled by building models within these defined blocks. The segment of the PFD that corresponds to the first block (=HA1) is shown in the following figure.



The second block is then broken down into a reaction system, a pumping system and a heat exchange system. There are two instrumentation functions modelled, one to control temperature in the reactor, and the second to control outlet pressure.



5 XML Schema

The above model has been implemented as an experimental XML schema. This schema supports the following types of file:

- Files that specify only the structure of the model.
- Files that specify only sets of qualified parameters for models that have been defined elsewhere. The parameters in the file are grouped by **ProcessItem** and **ProcessConnection**.
- Files that contain a mixture of structure and parameters. This is the transfer format for BFD or PFD with process data and stream tables.

The schema builds on the ISO 10303 common schema, as described in ISO/TS10303-15.

The Proteus schema is *not* referenced, but we have copied some units of measurement from this schema.

ID and IDREF tags are used to provide unique identifiers for elements in the model. These identifiers are used for cross-referencing and connecting objects.

A package of the schema and representations of example files is supplied as additional material. The contents of the package are:

- DexpiPlus.xsd – the schema file.
- Common.xsd – the ISO/TS10303-15 common schema file.
- ISO10628ReferenceBFD.xml – the Block Flow Diagram example from section 4.2.
- SimplerPFDFromISO10628.xml – the simplified PFD example from section 4.3.1.
- DetailedPFDFromISO10628.xml – the detailed PFD example from section 4.3.2.
- SimpleCombinedExample.xml – the example in section 4.4, with a combined, linked set of BFDs and PFDs in a single XML file.
- TennesseEastman.xml – the Tennessee Eastman example process.

The purpose of this work was to verify the feasibility of modelling the UML representation as an XML schema.

6 The Presentation Layer

6.1 Layering of the Presentation

The data model described above determines the topology of the BFD and PFD. However, it says nothing about the graphical presentation of the same. It is essential that graphical formatting of the drawings is independent of the process data model.

6.2 Symbols

6.2.1 Symbols in Block Flow Diagrams

Each block in a block flow diagram is usually represented by a rectangular symbol with space for a label.

6.2.2 Symbols in Process Flow Diagrams

Each block in a process flow diagram is represented by a symbol that indicates the activity carried out in that block. The symbol is usually based on the corresponding equipment symbol used in a P&ID. Note that the symbol in a PFD does not directly represent a piece of equipment, rather the activity that will be carried out in the process.

6.3 Connectors

The connectors in the drawing follow the topology defined by the data model. They need to provide a clean, simple and easy-to-read connection between two symbols. The connector should reference a **ProcessConnection** object in the data model. The end points for the connection are defined points on a symbol and correspond to a **Port** in the data model.

7 Implementation using AutomationML

7.1 Introduction

AutomationML provides a serialization format for the UML model presented above. By using AutomationML, we avoid the need to develop a new, non-standard XML or JSON schema.

The DEXPI schema is represented through four AML libraries:

- A **Role Class Library: DEXPIPlusRoleClassLib.** This defines the process step classes, the process connection classes, and the instrumentation activity classes.
- An **Interface Class Library: DEXPIPLusInterfaceLib.** This defines the ports in the DEXPI Plus model.
- An **Attribute Type Library: DEXPIPlusAttributeTypeLib**. This defines the qualified parameters we can use in a DEXPI plus model.
- A **System Unit Class Lib: DEXPIPlusSystemUnitClassLib**. This defines the MaterialTemplate and Case objects.

As an aid to interoperability, the DEXPI classes inherit from relevant classes in the AutomationML base libraries. This allows us to link DEXPI Plus semantics to AutomationML and also provides possibilities to use AutomationML patterns to work with DEXPI Plus files.

In particular, the AutomationML Product-Process-Resource pattern can be used to organize the relationships between **ProcessStep** blocks and **ProcessConnection** objects. The topological connections between **ProcessStep** blocks and **ProcessConnections** can be modelled using AutomationML's standard **PPRConnector** interface class.

A DEXPI **ProcessStep** inherits from the AutomationML **Process** role and a ProcessConnection inherits from AutomationML **Product**. Note that, in AutomationML, a product is something used by or produced by a process. This differs from the product aspect in ISO/IEC81346, which corresponds to the resource role in AutomationML.

7.2 Ports

The DEXPI material and energy port classes are all derived from the AutomationML **Port** interface class. The **InformationPort**, however, is derived from the AutomationML **SignalInterface** class. This reflects the different behaviours of these types of port, in that flows of material and energy are directional, whereas information flows.

The classes defined in the **DEXPIPlusInterfaceClassLib** are:

- **MaterialPort**.
- **EnergyPort**
    - **ElectricalEnergyPort**
    - **MechanicalEnergyPort**
    - **ThermalEnergyPort**
- **InformationPort**.

The **MaterialPort** and **EnergyPort** classes implement three interface classes.

- **FlowReference**, derived from **PPRConnector**. This allows connection to the **ProcessConnection** object that is connected to this port.

- **SubReference,** of type **HierachyConnector**. This interface can be connected to the **SuperReference** interface in another port. This allows downward hierarchical alignment of ports.
- **SuperReference,** of type **HierachyConnector**. This interface can be connected to the **SubReference** interface in another port. This allows upward hierarchical alignment of ports.

7.3 Process Step and Process Step Detail

A DEXPI **ProcessStep** is modelled as a Role Class. It is derived from the AutomationML Base Role class **Process.**
Each **ProcessStep** child class in DEXPI is then derived in a straightforward way from the **ProcessSteo** top class.

An instance with the **ProcessStep** role can own any number of **MaterialPort**, **EnergyPort** (or its children) and **InformationPort** interfaces.

A **ProcessStepDetail** role is also derived from the base **Process** role.

7.4 Instrumentation System Activity and Instrumentation Activity

The **InstrumentationSystemActivity** and **InstrumentationActivity** classes are both role classes derived from the **Process** AutomationML base class.

The child classes of **InstrumentationActivity** are derived in a straightforward way from **InstrumentationActivity**.

7.5 Process Connections: Stream, Energy Flow and Information Flow

The flows of material, energy and information between blocks are modelled by objects with a **ProcessConnection** role. This role is derived from the AutomationML **Product** role.

The DEXPI plus **ProcessConnection** classes: **Stream**, **EnergyFlow**, **ElectricalEnergyFlow**, **ThermalEnergyFlow**, **MechanicalEnergyFlow** and **InformationFlow** are then defined by specialization from the **ProcessConnection** role.

All **ProcessConnection** role classes expose two interfaces that allow connection to the **FlowReference** interface in a port. Each of these interfaces are derived from the AutomationML **PPRConnector** class. The **Source** interface is connected to the upstream port while the **Target** is connected to the downstream port.

7.6 Modelling Material Properties

The DEXPI classes for modelling the properties of **Stream** objects are implemented in the **DEXPIPlusAttributeTypeLib** and **DEXPIPlusSystemUnitClass** AutomationML libraries.

When building a model, a **MaterialTemplate** object is created for each material type in the model by making an instance of the **MaterialTemplate** class from the **DEXPIPlusSystemUnitClass** library. This object is then parameterised with an *Identifier* (of AutomationML type **Category**), *Description*, *Label*, *NumberOfComponents*, *NumberOfPhases* and a *ListOfMaterialComponents*.

Each **Stream** object in the model will then have a *MaterialTemplate* attribute, of AutomationML base type **Category**. This is the *Identifier* attribute of the stream's material template object. This then defines the structure of the complex *MaterialState* attribute, with type **MaterialState**.

The **MaterialState** attribute can contain all the information in a detailed stream table, with composition and physical properties for the total stream and each phase in the stream.

7.7 Example File and Library

The file DEXPIPlusBFDExample.aml file contains the DEXPI Process AML libraries and a representation of the BFD example.

8 References

[1] ISO TC10, 2014. ISO 10628-1 Diagrams for the chemical and petrochemical industry Part 1: Specification of diagrams (International Standard No. ISO 10628-1:2014(E)). ISO, Geneva.

[2] VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA), Fachbereich Engineering und Betrieb automatisierter Anlagen, 2015. VDI/VDE 3682 Blatt 1 Formalisierte Prozessbeschreibungen Konzept und grafische Darstellung Part 1 Formalised process descriptions Concept and graphic representation (VDI/VDE-Richtlinien No. VDI/VDE 3682 Blatt 1). VDI Verein Deutscher Ingenieure, Berlin.

[3] Marquardt, W., Morbach, J., Wiesner, A., Yang, A., 2010, “OntoCAPE A Re-Usable Ontology for Chemical Process Engineering”, Berlin, Springer.

[4] DIN-Normenausschuss Technische Grundlagen (NATG), DIN-Normenausschuss Rohrleitungen und Dampfkesselanlagen (NARD), 2018. DIN 6779-13 Kennzeichnungssystematik für technische Produkte und technische Produktdokumentation – Teil 13: Chemieanlagen (Deutsche Norm). DIN Deutsches Institut für Normung e. V., Berlin.

9 Appendix 1: Parameter Dictionary

Proposed name	Variable Type (DEXPI)	IEC CDD	ISO15926-4 RDS Link	ISO15926-4 RDS Label
ActivePower	Power	0112/2///61987#ABN713 - active power	http://data.posccaesar.org/rdl/RDS384389	ACTIVE POWER
AirFlow	MassFlowRate			
AirVolumeFlow	VolumeFlow		http://data.posccaesar.org/rdl/RDS14261660	AIR VOLUME FLOW RATE
ApparentPower	ApparentPower	0112/2///61987#ABN714 - apparent power	http://data.posccaesar.org/rdl/RDS384434	APPARENT POWER
Area	Area			
ComponentSeparationEfficiency	Percentage			
CompressionRatio	Fraction			
Current	Current	0112/2///61987#ABN620 - current	http://data.posccaesar.org/rdl/RDS351089	ELECTRIC CURRENT
CycleTime	TimeInterval		http://data.posccaesar.org/rdl/RDS16806816	CYCLE TIME
Density	Density		http://data.posccaesar.org/rdl/RDS358874	DENSITY
Depth	Length	0112/2///61987#ABA195 - depth	http://data.posccaesar.org/rdl/RDS373139	DEPTH
Diameter	Length	0112/2///61987#ABB952 - diameter	http://data.posccaesar.org/rdl/RDS350954	DIAMETER
Duty	Power			
DynamicViscosity	DynamicViscosity		http://data.posccaesar.org/rdl/RDS357614	VISCOSITY
Efficiency	Percentage		http://data.posccaesar.org/rdl/RDS362654	EFFICIENCY
ElectricalConductivity	ElectricalConductivity	0112/2///61987#ABA560 electrical conductivity	http://data.posccaesar.org/rdl/RDS350639	CONDUCTIVITY
ElectricalFrequency	ElectricalFrequency	0112/2///61987#ABN624 - frequency	http://data.posccaesar.org/rdl/RDS351989	FREQUENCY
ElectricalPower	Power			

Proposed name	Variable Type (DEXPI)	IEC CDD	ISO15926-4 RDS Link	ISO15926-4 RDS Label
ElectricalResistance	ElectricalResistance		http://data.posccaesar.org/rdl/RDS7923008	ELECTRICAL RESISTANCE
EvaporationRate	MassFlowRate			
ExtentOfReaction	Percentage			
Flow	MassFlowRate	0112/2///61987#ABB290 - mass flow rate	http://data.posccaesar.org/rdl/RDS380789	MASS FLOW RATE
FoulingFactor	HeatTransferResistance		http://data.posccaesar.org/rdl/RDS364364	FOULING FACTOR
FuelConsumption	MassFlowRate			
GasMassFlow	MassFlowRate			
Head	Length			
HeatCapacity	HeatCapacity		http://data.posccaesar.org/rdl/RDS382184	HEAT CAPACITY
HeatTransferCoefficient	HeatTransferCoefficient		http://data.posccaesar.org/rdl/RDS381959	COEFFICIENT OF HEAT TRANSFER
HeatTransferResistance	HeatTransferResistance			
Height	Length	0112/2///61987#ABA574 - height	http://data.posccaesar.org/rdl/RDS357704	HEIGHT
HydraulicEfficiency	Percentage			
HydraulicHead	Length			
KinematicViscosity	KinematicViscosity	0112/2///61987#ABB300 - kinematic viscosity	http://data.posccaesar.org/rdl/RDS356399	KINEMATIC VISCOSITY
LatentHeatOfFusion	MassSpecificEnergy		http://data.posccaesar.org/rdl/RDS366074	LATENT HEAT OF FUSION
LatentHeatOfVaporization	MassSpecificEnergy		http://data.posccaesar.org/rdl/RDS352574	LATENT HEAT OF VAPORIZATION
Length	Length	0112/2///61987#ABA640 - length	http://data.posccaesar.org/rdl/RDS373094	LENGTH
Level	Length	0112/2///61987#ABH329 - level	http://data.posccaesar.org/rdl/RDS7966320	OPERATING LEVEL

Proposed name	Variable Type (DEXPI)	IEC CDD	ISO15926-4 RDS Link	ISO15926-4 RDS Label
Mass	Mass	0112/2///61987#ABA553 weight	http://data.posccaesar.org/rdl/RDS353339	MASS
MassConcentration	MassConcentration	0112/2///61987#ABN628 - mass concentration (per volume)	http://data.posccaesar.org/rdl/RDS388034	CONCENTRATION MASS PER VOLUME
MassFlow	MassFlowRate	0112/2///61987#ABB290 - mass flow rate	http://data.posccaesar.org/rdl/RDS380789	MASS FLOW RATE
MassSpecificEnthalpy	MassSpecificEnergy		http://data.posccaesar.org/rdl/RDS382679	SPECIFIC ENTHALPY
MolarFlow	MoleFlow			
MolarHeatCapacity	MolarHeatCapacity		http://data.posccaesar.org/rdl/RDS387854	MOLAR HEAT CAPACITY
MolarSpecificEnthalpy	MoleSpecificEnergy			
MolarWeight	MolarMass		http://data.posccaesar.org/rdl/RDS387719	MOLAR MASS
MoleConcentration	MoleConcentration	0112/2///61987#ABN701 - molar concentration (per volume)	http://data.posccaesar.org/rdl/RDS388124	CONCENTRATION MOLE PER VOLUME
Number	Integer			
NumberOfStages	Integer			
NumberOfTheoreticalStages	Integer			
ParticleSize	ParticleSize		http://data.posccaesar.org/rdl/RDS403019	PARTICLE DIAMETER
pH	pH	0112/2///61987#ABN617 - pH value	http://data.posccaesar.org/rdl/RDS1060066911	pH
PhaseSeparationEfficiency	Percentage			
PolytropicEfficiency	Percentage			
Power	Power	0112/2///61987#ABN625 - power	http://data.posccaesar.org/rdl/RDS354104	POWER
Pressure	AbsolutePressure	0112/2///61987#ABN616 - pressure	http://data.posccaesar.org/rdl/RDS354194	PRESSURE

Proposed name	Variable Type (DEXPI)	IEC CDD	ISO15926-4 RDS Link	ISO15926-4 RDS Label
PressureDifference	AbsolutePressure		http://data.posccaesar.org/rdl/RDS361574	DIFFERENTIAL PRESSURE
ReactionEnthalpy	MoleSpecificEnergy			
ReactivePower	ApparentPower	0112/2///61987#ABN715 - reactive power	http://data.posccaesar.org/rdl/RDS384479	REACTIVE POWER
RelativeHumidity	Percentage	0112/2///61987#ABN638 - relative humidity		
RetentionTime	TimeInterval		http://data.posccaesar.org/rdl/RDS365939	RESIDENCE TIME
RotationalFrequency	RotationalFrequency	0112/2///61987#ABF194 - speed of rotation	http://data.posccaesar.org/rdl/RDS361034	ROTATIONAL SPEED
ShaftPower	Power		http://data.posccaesar.org/rdl/RDS361259	SHAFT POWER
SolidsMassFlow	MassFlowRate			
SurfaceTension	SurfaceTension		http://data.posccaesar.org/rdl/RDS355769	SURFACE TENSION
Temperature	Temperature	0112/2///61987#ABA927 - temperature	http://data.posccaesar.org/rdl/RDS355859	TEMPERATURE
TemperatureDifference	Temperature		http://data.posccaesar.org/rdl/RDS14118235	TEMPERATURE DIFFERENCE
ThermalConductivity	ThermalConductivity	0112/2///61987#ABB094 - thermal conductivity	http://data.posccaesar.org/rdl/RDS355949	THERMAL CONDUCTIVITY
Time	TimeInterval	0112/2///61987#ABN626 - time	http://data.posccaesar.org/rdl/RDS356039	TIME
Torque	MomentOfForce		http://data.posccaesar.org/rdl/RDS356084	TORQUE
VapourPressure	AbsolutePressure		http://data.posccaesar.org/rdl/RDS358739	VAPOUR PRESSURE
Velocity	Velocity	0112/2///61987#ABA948 - flow velocity of fluid	http://data.posccaesar.org/rdl/RDS369494	LINEAR VELOCITY
Viscosity	DynamicViscosity	0112/2///61987#ABB067 - dynamic viscosity	http://data.posccaesar.org/rdl/RDS357614	VISCOSITY

Proposed name	Variable Type (DEXPI)	IEC CDD	ISO15926-4 RDS Link	ISO15926-4 RDS Label
Voltage	Voltage	0112/2///61987#ABN618 - voltage	http://data.posccaesar.org/rdl/RDS372374	VOLTAGE
Volume	Volume		http://data.posccaesar.org/rdl/RDS356444	VOLUME
VolumeFlow	VolumeFlow	0112/2///61987#ABN698 - volume flow rate	http://data.posccaesar.org/rdl/RDS380834	VOLUME FLOW RATE
WaterFlow	MassFlow			
Width	Length	0112/2///61987#ABA573 - width	http://data.posccaesar.org/rdl/RDS361709	WIDTH