Filter plugin HNZ to Pivot

This plugin can be used to convert HNZ objects to FledgePower pivot model objects.

The filter implements the Fledge Filter plugin interface (see filter_plugins).

Common rules

Filter configuration

See Plugins configuration design for examples and details.

The "`config`" parameter of "`plugin_init`" call shall include :

- A "`exchanged_data`" category with the same content as provided to the HNZ south plugin. This section is mandatory so that the filter plugin can convert the PIVOT type from HNZ type.

Notes :

- All types not listed in this table are not supported in current version.
- The current implementation provides a default mapping rule for each known type, but some new rules might be added in the future and configured using the "`alternate_mapping_rule`" option in its parameters.

Filter interface

The "`plugin_ingest`" call will convert each "`reading`" of "`reading_set`" as follow:

- The Reading fields "id", "TimeStamp" and "userTimestamp" are unchanged.
- The "asset_name" field shall be mapped with "`exchanged_data.datapoints.label`"
- The Reading field "reading" is updated with a JSON object `{'<Root> key' : {...}}` . The content of object under `'<Root> key'` is given below.

The <Root> key of PIVOT object is:

Reading key	Content
PIVOT.GTIS	Tele Signalisation
PIVOT.GTIM	Tele Measurement
PIVOT.GTIC	Tele command

`<type>` conversion table:

CDC Class	HNZ Type ID
SpsTyp/DpsTyp	Status point (TS)
MvTyp	Measurement (TM)
SpcTyp/DpcTyp/IncTyp	Command (TC/TVC)

Converting monitoring data types

The content under <Root> will convert the HNZ data objet to a pivot object as follow:

Tele-Informations

Key	Type	Default Value	HNZ.data_object. <...>	Note	Mandatory (of do field)
`<Root>.<type>`	{CDC}		do_type	exchanged_data.datapoints.pivot_type (see <type> conversion table above)	YES
`<Root>.Cause.stVal`	Integer		do_type, do_cg	3 (TS on CE) | 20 (TS on CG) | 1 (TM cyclic) | 7 (TC ACK or TVC ACK)	YES

`<Root>.Confirmation.stVal`	Boolean	false	do_valid	[0..1] (0 => ACK, 1 => NACK)	YES (if TC /TVC ACK)
`<Root>.ComingFrom`	String	"hnzip"			NO
`<Root>.SpsTyp.stVal`	Boolean		do_value	[0..1] (0 => OFF, 1 => ON)	YES (if TS Simple)
`<Root>.DpsTyp.stVal`	String		do_value	off | on	YES (if TS Double)
`<Root>.MvTyp.mag.i`	Integer		do_value	Int 32	YES (if TM)
`<Root>.Identifier`	String		do_id	exchanged_data.datapoints.pivot_id	YES
`<Root>.<type>.TmOrg.stVal`	String	"genuine"		If the timestamp is not from the HNZ frame => "substituted" Else => "genuine" eg: timestamps created artifically for any TI that doesn't have one will be "substitued".	YES
`<Root>.<type>.q.Validity`	String	"good"	do_valid, do_ts_c, do_ts_s, do_outdated	By order of priority: do_valid = 1 => "invalid" do_outdated = 1 => "questionable" do_ts_c = 1 => "questionable" do_ts_s = 1 => "questionable" else "good"	NO
`<Root>.<type>.t.SecondSinceEpoch` `<Root>.<type>.t.FractionOfSecond`	Integer Integer		do_ts	SecondSinceEpoch is the time in seconds since 1970- 01-01 00:00:00 UTC FractionOfSecond is the fraction of second (Computation method described in 61850 protocol documentation).	YES
`<Root>.TmValidity.stVal`	String	"good"	do_ts_iv	good (0) | invalid (1)	NO
`<Root>.<type>.q.DetailQuality.oldData`	Boolean	<undefined>	do_ts_c, do_outdated	By order of priority: do_outdated = 1 => true do_ts_c = 1 => true else <undefined>	NO
`<Root>.<type>.t.TimeQuality.clockNotSynchronized`	Boolean	<undefined>	do_ts_s	do_ts_s = 1 => true else <undefined>	NO

Filter rules

STATION => (ASDU) HNZ SOUTH => (HNZ DP) HNZTOPIVOT =>(PIVOT DP) PIVOTTOIEC104 => (IEC104 DP) IEC104 NORTH => (ASDU) CENTER

Rule 1: if the incoming HNZ data object has not the attribute or has the default value then we don't have to create the corresponding attribute in the pivot object.

Rule 2: If the received pivot object has not an expected attribute then we create the attribute of the protocol specific datapoint with default value.

Rule 3: Case when ASDU timestamp is not received:

The HNZTOPIVOT step:
If the received ASDU is without timestamp Then
  We create a pivot object With timestamp = current system time
  And <>.TmOrg.stVal = substituted

Converting commands data types

The content under <Root> will convert the HNZ data objet to a pivot object as follow:

Key	Type	Default Value	HNZ. command_object.<...>	Note	Mandatory (of co field)

`<Root>.ComingFrom`	String	"hnzip"			NO
`<Root>.Identifier`	String		co_type + co_addr	exchanged_data.datapoints.pivot_id	YES
`<Root>.<type>`	{CDC}		co_type	SpcTyp (TC) | DpcTyp (TC) | IncTyp (TVC)	YES
`<Root>.SpcTyp.ctlVal`	Boolean		co_value	[0..1]	YES (if TC Simple)
`<Root>.DpcTyp.ctlVal`	String		co_value	off | on	YES (if TC Double)
`<Root>.IncTyp.ctlVal`	Int		co_value	Int 32	YES (if TVC)

Filter rules

CENTER => (ASDU) IEC104 NORTH => (IEC104 DP) IEC104TOPIVOT =>(PIVOT DP) PIVOTTOHNZ => (HNZ DP) HNZ SOUTH=> (ASDU) STATION

Rule 1: if the incoming HNZ data object has not the attribute or has the default value then we don't have to create the corresponding attribute in the pivot object.

Rule 2: If the received pivot object has not an expected attribute then we create the attribute of the protocol specific datapoint with default value.

Rule 3: the timestamp of the original command issued by the Center must be transmitted as is to the Station.

Rule 4: Case when IEC104 command object timestamp is not received for ASDU type with timestamp, then HNZ command object is rejected with error message.

Rule 5: Case when IEC104 command object for ASDU type without timestamp is received then:

The PIVOTTOHNZ step:
If the received ASDU is without timestamp Then We create a pivot object With timestamp = current system time

Converting Pivot object to HNZ data

The conversion of a Pivot object to HNZ data is the inverse operation of the rules above:

Filter rules

Rules 1 : GTIC pivot object is converted to a TC HNZ object