Filter plugin iec104 to Pivot

This plugin can be used to convert IEC 104 ASDU 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 IEC104 south plugin. This section is mandatory so that the filter plugin can convert the PIVOT type from IEC 104 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
PIVOTTS.GTIS	Status Point
PIVOTTM.GTIM	Measurement
PIVOTTC.GTIC	Command or Set Point

`<type>` conversion table:

Root	CDC Class	IEC 104 Type ID	Type ID with timetag	Alternative format type id
GTIS	SpsTyp	M_SP_NA_1	M_SP_TA_1,M_SP_TB_1	M_PS_NA_1
GTIS	DpsTyp	M_DP_NA_1	M_DP_TA_1,M_DP_TB_1	M_EP_TA_1,M_EP_TD_1
GTIM	MvTyp	M_ME_NA_1	M_ME_TA_1,M_ME_TD_1	M_ME_ND_1
GTIM	MvTyp	M_ME_NB_1	M_ME_TB_1,M_ME_TE_1	
GTIM	MvTyp	M_ME_NC_1	M_ME_TC_1,M_ME_TF_1	
GTIM	BscTyp	M_ST_NA	M_ST_TA_1,M_ST_TB	
GTIC	SpcTyp	C_SC_NA_1	C_SC_TA_1	
GTIC	DpcTyp	C_DC_NA_1	C_DC_TA_1	
GTIC	IncTyp	C_SE_NB_1	C_SE_TB_1	
GTIC	ApcTyp	C_SE_NA_1, C_SE_NC_1	C_SE_TA_1, C_SE_TC_1	
GTIC	BscTyp	C_RC_NA_1	C_RC_TA_1	

Converting monitoring data types

The content under <Root> will convert the IEC 104 data objet to a pivot object as follow:

Key	Type	Default Value	iec104.data_object.<...>	Note
<Root>.Cause.stVal	Integer		do_cot	See Cause of Transmission
<Root>.Confirmation.stVal	Boolean	false	do_negative	
<Root>.ComingFrom	String			This plugin should always use the value "iec104" when converting to pivot
<Root>.Identifier	String			The plugin will populate the "exchanged_data.datapoints.pivot_id" using the CA-IOA from the input iec104 data object and the "datapoints.protocols.address" from the exchanged_data configuration
<Root>.<type>	{CDC}		do_type	The plugin will populate the "exchanged_data.datapoints.pivot_type" using the ASDU type from the input iec104 data object and the "datapoints.protocols.typeid" from the exchanged_data configuration (see <type> conversion table above)
<Root>.<type>.q.Validity = "questionable" <Root>.<type>.q.DetailQuality.overflow = true	Boolean	false	do_quality_ov = true	
<Root>.<type>.q.test	Boolean	false	do_test	
<Root>.<type>.q.operatorBlocked	Boolean	false	do_quality_bl	
<Root>.<type>.q.Source = "substituted"	String	"process"	do_quality_sb = true	"process" | "substituted"
<Root>.<type>.q.Validity = "invalid"	String	"good"	do_quality_iv = true	"good" | "invalid" | "reserved" | "questionable"
<Root>.<type>.t.SecondSinceEpoch	Integer		do_ts	
<Root>.TmOrg.stVal	String		do_ts_org	"genuine" | "substituted"
<Root>.TmValidity.stVal = "invalid"	String	"good"	do_ts_iv = true	"good" | "invalid" | "reserved" | "questionable"
<Root>.<type>.q.Validity = "questionable" <Root>.<type>.q.DetailQuality.oldData = true	String	"good"	do_quality_nt = true	"good" | "invalid" | "reserved" | "questionable"
<Root>.SpsTyp.stVal	Boolean		do_value	
<Root>.DpsTyp.stVal	String		do_value	intermediate-state | off | on | bad-state
<Root>.MvTyp.mag.f	Float		do_value	Float 32
<Root>.MvTyp.mag.i	Integer		do_value	Int 32
<Root>.BscTyp.valWTr.posVal	Integer		do_value	Int 8
<Root>.BscTyp.valWTr.transInd	Boolean		do_value	Boolean

Filter rules

STATION => (ASDU) IEC104 SOUTH => (IEC104 DP) IEC104TOPIVOT =>(PIVOT DP) PIVOTTOIEC104 => (IEC104 DP) IEC104 NORTH => (ASDU) CENTER

Rule 1: if the incoming IEC104 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:

A the IEC104TOPIVOT step:
If the received ASDU is without timestamp Then
    We create a pivot object With timestamp = current system time
    And <>.TmOrg.stVal = substituted

On the PIVOTTOIEC104 step:
We create an IEC104 datapoint with timestamp and do_ts_sub = true

IEC104 NORTH step:
If the Datapoint configured in the exchanged data has not IEC104 timestamp typeid
Then we create an ASDU with a typeid with no timestamp
If the Datapoint configured in the exchanged data has IEC104 timestamp typeid
Then we create an ASDU with a timestamp typeid, ASDU timestamp = do_ts And subtituted = do_ts_sub

Converting commands data types

The content under <Root> will convert the IEC 104 command objet to a pivot object as follow:

Key	Order	Type	Default Value	iec104.data_object.<...>	Note
`<Root>.Cause.stVal`		Integer		co_cot	See Cause of Transmission
`<Root>.ComingFrom`		String			This plugin should always use the value "iec104" when converting to pivot
`<Root>.Identifier`		String			The plugin will populate the "exchanged_data.datapoints.pivot_id" using the CA-IOA from the input iec104 command object and the "datapoints.protocols.address" from the exchanged_data configuration
`<Root>.Select.stVal`		Boolean	false	co_se	- 0 is mapped with false, for Execute - 1 is mapped with true, for Select before Execute
`<Root>.<type>`		`{CDC}`		co_type	The plugin will populate the "exchanged_data.datapoints.pivot_type" using the ASDU type from the input iec104 command object and the "datapoints.protocols.typeid" from the exchanged_data configuration (see `<type>` conversion table above)
`<Root>.<type>.q.test`		Boolean	false	co_test	
`<Root>.<type>.t.SecondSinceEpoch`		Integer		co_ts	
`<Root>.SpcTyp.ctlVal`		Boolean		co_value	0 or 1
`<Root>.DpcTyp.ctlVal`		String		co_value	intermediate-state | off | on | bad-state
`<Root>.IncTyp.ctlVal`		Integer		co_value	Int 32
`<Root>.ApcTyp.ctlVal`		Float		co_value	Float 32
`<Root>.BscTyp.ctlVal`		String		co_value	stop | lower | higher | reserved

Filter rules

CENTER => (ASDU) IEC104 NORTH => (IEC104 DP) IEC104TOPIVOT =>(PIVOT DP) PIVOTTOIEC104 => (IEC104 DP) IEC104 SOUTH=> (ASDU) STATION

Rule 1: if the incoming IEC104 command 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 command 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 IEC104 command object is rejected with error message.

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

A the IEC104TOPIVOT step:
If the received ASDU is without timestamp Then We create a pivot object With timestamp = current system time