



European Network of  
Transmission System Operators  
for Electricity

---

# STEADY STATE HYPOTHESIS SCHEDULE PROFILE SPECIFICATION

---

2024-10-16

---

ICTC APPROVED  
VERSION 1.0.1

## 1 Copyright notice:

2 **Copyright © ENTSO-E. All Rights Reserved.**

3 This document and its whole translations may be copied and furnished to others, and derivative  
4 works that comment on or otherwise explain it or assist in its implementation may be prepared,  
5 copied, published and distributed, in whole or in part, without restriction of any kind, provided  
6 that the above copyright notice and this paragraph are included on all such copies and  
7 derivative works. However, this document itself may not be modified in any way, except for  
8 literal and whole translation into languages other than English and under all circumstances, the  
9 copyright notice or references to ENTSO-E may not be removed.

10 This document and the information contained herein is provided on an "as is" basis.

11 **ENTSO-E DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT**  
12 **LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT**  
13 **INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR**  
14 **FITNESS FOR A PARTICULAR PURPOSE.**

15 **This document is maintained by the ENTSO-E CIM WG. Comments or remarks are to be**  
16 **provided at [cim@entsoe.eu](mailto:cim@entsoe.eu)**

17 **NOTE CONCERNING WORDING USED IN THIS DOCUMENT**

18 The force of the following words is modified by the requirement level of the document in which  
19 they are used.

- 20 • SHALL: This word, or the terms "REQUIRED" or "MUST", means that the definition is an  
21 absolute requirement of the specification.
- 22 • SHALL NOT: This phrase, or the phrase "MUST NOT", means that the definition is an  
23 absolute prohibition of the specification.
- 24 • SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid  
25 reasons in particular circumstances to ignore a particular item, but the full implications must  
26 be understood and carefully weighed before choosing a different course.
- 27 • SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may  
28 exist valid reasons in particular circumstances when the particular behaviour is acceptable  
29 or even useful, but the full implications should be understood and the case carefully weighed  
30 before implementing any behaviour described with this label.
- 31 • MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional.

32

33

## Revision History

Version	Date	Paragraph	Comments
1.0.0-alpha	2024-03-20		For CIM WG review.
1.0.0-beta	2024-04-08		For StG Strategy review.
1.0.1-alpha	2024-09-07		For CIM WG review.

34	<b>CONTENTS</b>		
35	Copyright notice:		2
36	Revision History		3
37	CONTENTS		4
38	1	Introduction	14
39	2	Application profile specification	14
40	2.1	Version information	14
41	2.2	Constraints naming convention	14
42	2.3	Profile constraints	15
43	2.4	Metadata	17
44	2.4.1	Constraints	18
45	2.4.2	Reference metadata	18
46	3	Package SteadyStateHypothesisScheduleProfile	18
47	3.1	General	18
48	3.2	(abstract) IdentifiedObject root class	19
49	3.3	(abstract,NC) BaseTimeSeries	20
50	3.4	Season	20
51	3.5	(NC) HourPattern	20
52	3.6	(abstract,NC) BaseRegularIntervalSchedule	21
53	3.7	(NC) HourPeriod root class	21
54	3.8	(NC) BaseIrregularTimeSeries	22
55	3.9	(NC) BaseTimeSeriesKind enumeration	22
56	3.10	(NC) TimeSeriesInterpolationKind enumeration	23
57	3.11	(NC) RegulatingControlRegularSchedule	23
58	3.12	(abstract) RegulatingControl root class	24
59	3.13	(NC) TapChangerControlRegularSchedule	25
60	3.14	(abstract) TapChangerControl root class	26
61	3.15	VsQpccControlKind enumeration	26
62	3.16	PerCent datatype	26
63	3.17	(NC) EnergyConnectionRegularSchedule	26
64	3.18	(abstract) EnergyConnection root class	27
65	3.19	(NC) TapRegularSchedule	27
66	3.20	(abstract) TapChanger root class	28
67	3.21	(NC) SwitchRegularSchedule	28
68	3.22	(abstract) Switch root class	29
69	3.23	(NC) InServiceRegularSchedule	29
70	3.24	(abstract) Equipment root class	30
71	3.25	(NC) ControlAreaRegularSchedule	30
72	3.26	(abstract) ControlArea root class	31
73	3.27	(NC) SynchronousMachineRegularSchedule	31
74	3.28	(abstract) SynchronousMachine root class	32
75	3.29	(NC) AsynchronousMachineRegularSchedule	32
76	3.30	(abstract) AsynchronousMachine root class	33
77	3.31	(NC) ExternalNetworkInjectionRegularSchedule	33

78	3.32	(abstract) ExternalNetworkInjection root class.....	34
79	3.33	(abstract,NC) ACDCConverterRegularSchedule .....	34
80	3.34	(NC) VsConverterRegularSchedule .....	35
81	3.35	(NC) CsConverterRegularSchedule .....	37
82	3.36	(abstract) VsConverter root class .....	38
83	3.37	(abstract) CsConverter root class .....	38
84	3.38	(NC) EquivalentInjectionRegularSchedule .....	38
85	3.39	(abstract) EquivalentInjection root class .....	39
86	3.40	(NC) EquivalentInjectionSchedule .....	39
87	3.41	(NC) EquivalentInjectionTimePoint root class .....	40
88	3.42	(NC) EnergyConnectionSchedule .....	41
89	3.43	(NC) EnergyConnectionTimePoint root class .....	41
90	3.44	(NC) TapSchedule .....	42
91	3.45	(NC) TapScheduleTimePoint root class .....	42
92	3.46	(NC) RegulatingControlSchedule .....	43
93	3.47	(NC) RegulatingControlTimePoint root class .....	43
94	3.48	(NC) TapChangerControlSchedule .....	44
95	3.49	(NC) VsConverterSchedule .....	45
96	3.50	(abstract,NC) ACDCTimePoint root class .....	45
97	3.51	(NC) VsConverterTimePoint .....	45
98	3.52	(NC) CsConverterTimePoint.....	46
99	3.53	(NC) CsConverterSchedule .....	47
100	3.54	(NC) SwitchSchedule .....	48
101	3.55	(NC) SwitchTimePoint root class .....	48
102	3.56	(NC) InServiceSchedule .....	49
103	3.57	(NC) InServiceTimePoint root class.....	49
104	3.58	(NC) ControlAreaSchedule .....	50
105	3.59	(NC) ControlAreaTimePoint root class.....	50
106	3.60	(NC) SynchronousMachineSchedule .....	51
107	3.61	(NC) SynchronousMachineTimePoint root class .....	51
108	3.62	(NC) AsynchronousMachineSchedule .....	52
109	3.63	(NC) AsynchronousMachineTimePoint root class .....	52
110	3.64	(NC) ExternalNetworkInjectionSchedule .....	53
111	3.65	(NC) ExternalNetworkInjectionTimePoint root class .....	53
112	3.66	(NC) BatteryUnitSchedule .....	54
113	3.67	(NC) BatteryUnitTimePoint root class .....	55
114	3.68	(abstract) BatteryUnit root class.....	55
115	3.69	(NC) VoltageLimitSchedule .....	55
116	3.70	(abstract) VoltageLimit root class .....	56
117	3.71	(NC) VoltageLimitTimePoint root class .....	56
118	3.72	(NC) ActivePowerLimitSchedule .....	56
119	3.73	(abstract) ActivePowerLimit root class .....	57
120	3.74	(NC) ActivePowerLimitTimePoint root class .....	57
121	3.75	(NC) ApparentPowerLimitSchedule .....	57
122	3.76	(abstract) ApparentPowerLimit root class .....	58
123	3.77	(NC) ApparentPowerLimitTimePoint root class .....	58

124	3.78	(NC) CurrentLimitSchedule .....	58
125	3.79	(abstract) CurrentLimit root class .....	59
126	3.80	(NC) CurrentLimitTimePoint root class .....	59
127	3.81	(NC) ShuntCompensatorSchedule.....	59
128	3.82	(NC) ShuntCompensatorTimePoint root class .....	60
129	3.83	(abstract) ShuntCompensator root class .....	60
130	3.84	(NC) StaticVarCompensatorSchedule .....	61
131	3.85	(NC) StaticVarCompensatorTimePoint root class .....	61
132	3.86	(abstract) StaticVarCompensator root class .....	62
133	3.87	(NC) GeneratingUnitSchedule .....	62
134	3.88	(NC) GeneratingUnitTimePoint root class .....	62
135	3.89	(abstract) GeneratingUnit root class .....	63
136	3.90	MonthDay primitive .....	63
137	3.91	ActivePower datatype .....	63
138	3.92	Float primitive .....	63
139	3.93	UnitMultiplier enumeration .....	63
140	3.94	UnitSymbol enumeration .....	64
141	3.95	ReactivePower datatype .....	65
142	3.96	Voltage datatype .....	65
143	3.97	DateTime primitive .....	65
144	3.98	ApparentPower datatype .....	65
145	3.99	AsynchronousMachineKind enumeration .....	65
146	3.100	(NC) DayOfWeekKind enumeration .....	66
147	3.101	Integer primitive .....	66
148	3.102	BatteryStateKind enumeration.....	66
149	3.103	RealEnergy datatype.....	67
150	3.104	CsOperatingModeKind enumeration .....	67
151	3.105	CsPpccControlKind enumeration.....	67
152	3.106	AngleDegrees datatype .....	67
153	3.107	CurrentFlow datatype .....	67
154	3.108	Boolean primitive .....	68
155	3.109	(NC) PeakKind enumeration.....	68
156	3.110	(NC) EnergyDemandKind enumeration .....	68
157	3.111	String primitive.....	68
158	3.112	Time primitive .....	68
159	3.113	SynchronousMachineOperatingMode enumeration .....	68
160	3.114	PU datatype .....	69
161	3.115	Resistance datatype .....	69
162	3.116	VsPpccControlKind enumeration .....	69
163		Annex A (informative): Sample data .....	71
164	A.1	General.....	71
165	A.2	Sample instance data.....	71
166			
167		<b>List of figures</b>	

168	Figure 1 – Class diagram SteadyStateHypothesisScheduleProfile::IrregularSchedule .....	18
169	Figure 2 – Class diagram SteadyStateHypothesisScheduleProfile::RegularSchedule .....	19
170	Figure 3 – Class diagram SteadyStateHypothesisScheduleProfile::Core .....	19
171		
172	<b>List of tables</b>	
173	Table 1 – Attributes of SteadyStateHypothesisScheduleProfile::IdentifiedObject .....	19
174	Table 2 – Attributes of SteadyStateHypothesisScheduleProfile::BaseTimeSeries .....	20
175	Table 3 – Attributes of SteadyStateHypothesisScheduleProfile::Season .....	20
176	Table 4 – Attributes of SteadyStateHypothesisScheduleProfile::HourPattern .....	21
177	Table 5 – Attributes of	
178	SteadyStateHypothesisScheduleProfile::BaseRegularIntervalSchedule.....	21
179	Table 6 – Association ends of	
180	SteadyStateHypothesisScheduleProfile::BaseRegularIntervalSchedule with other	
181	classes .....	21
182	Table 7 – Attributes of SteadyStateHypothesisScheduleProfile::HourPeriod.....	22
183	Table 8 – Association ends of SteadyStateHypothesisScheduleProfile::HourPeriod with	
184	other classes .....	22
185	Table 9 – Attributes of	
186	SteadyStateHypothesisScheduleProfile::BaseIrregularTimeSeries .....	22
187	Table 10 – Literals of SteadyStateHypothesisScheduleProfile::BaseTimeSeriesKind .....	23
188	Table 11 – Literals of	
189	SteadyStateHypothesisScheduleProfile::TimeSeriesInterpolationKind .....	23
190	Table 12 – Attributes of	
191	SteadyStateHypothesisScheduleProfile::RegulatingControlRegularSchedule .....	23
192	Table 13 – Association ends of	
193	SteadyStateHypothesisScheduleProfile::RegulatingControlRegularSchedule with other	
194	classes .....	24
195	Table 14 – Attributes of	
196	SteadyStateHypothesisScheduleProfile::TapChangerControlRegularSchedule .....	25
197	Table 15 – Association ends of	
198	SteadyStateHypothesisScheduleProfile::TapChangerControlRegularSchedule with	
199	other classes .....	25
200	Table 16 – Literals of SteadyStateHypothesisScheduleProfile::VsQpccControlKind .....	26
201	Table 17 – Attributes of SteadyStateHypothesisScheduleProfile::PerCent .....	26
202	Table 18 – Attributes of	
203	SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule .....	26
204	Table 19 – Association ends of	
205	SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule with other	
206	classes .....	27
207	Table 20 – Attributes of SteadyStateHypothesisScheduleProfile::TapRegularSchedule .....	27
208	Table 21 – Association ends of	
209	SteadyStateHypothesisScheduleProfile::TapRegularSchedule with other classes .....	28
210	Table 22 – Attributes of	
211	SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule .....	28

212	Table 23 – Association ends of	
213	SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule with other classes .....	29
214	Table 24 – Attributes of	
215	SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule .....	29
216	Table 25 – Association ends of	
217	SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule with other classes .....	30
218	Table 26 – Attributes of	
219	SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule .....	30
220	Table 27 – Association ends of	
221	SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule with other	
222	classes .....	31
223	Table 28 – Attributes of	
224	SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule .....	32
225	Table 29 – Association ends of	
226	SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule with	
227	other classes .....	32
228	Table 30 – Attributes of	
229	SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule .....	33
230	Table 31 – Association ends of	
231	SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule with	
232	other classes .....	33
233	Table 32 – Attributes of	
234	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule .....	33
235	Table 33 – Association ends of	
236	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule	
237	with other classes .....	34
238	Table 34 – Attributes of	
239	SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule .....	35
240	Table 35 – Association ends of	
241	SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule with other	
242	classes .....	35
243	Table 36 – Attributes of	
244	SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule .....	36
245	Table 37 – Association ends of	
246	SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule with other	
247	classes .....	37
248	Table 38 – Attributes of	
249	SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule .....	37
250	Table 39 – Association ends of	
251	SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule with other	
252	classes .....	38
253	Table 40 – Attributes of	
254	SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule .....	39
255	Table 41 – Association ends of	
256	SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule with	
257	other classes .....	39
258	Table 42 – Attributes of	
259	SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule .....	40



260	Table 43 – Association ends of	
261	SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule with other	
262	classes .....	40
263	Table 44 – Attributes of	
264	SteadyStateHypothesisScheduleProfile::EquivalentInjectionTimePoint .....	40
265	Table 45 – Association ends of	
266	SteadyStateHypothesisScheduleProfile::EquivalentInjectionTimePoint with other	
267	classes .....	41
268	Table 46 – Attributes of	
269	SteadyStateHypothesisScheduleProfile::EnergyConnectionSchedule .....	41
270	Table 47 – Association ends of	
271	SteadyStateHypothesisScheduleProfile::EnergyConnectionSchedule with other classes .....	41
272	Table 48 – Attributes of	
273	SteadyStateHypothesisScheduleProfile::EnergyConnectionTimePoint .....	41
274	Table 49 – Association ends of	
275	SteadyStateHypothesisScheduleProfile::EnergyConnectionTimePoint with other	
276	classes .....	42
277	Table 50 – Attributes of SteadyStateHypothesisScheduleProfile::TapSchedule .....	42
278	Table 51 – Association ends of SteadyStateHypothesisScheduleProfile::TapSchedule	
279	with other classes .....	42
280	Table 52 – Attributes of	
281	SteadyStateHypothesisScheduleProfile::TapScheduleTimePoint .....	42
282	Table 53 – Association ends of	
283	SteadyStateHypothesisScheduleProfile::TapScheduleTimePoint with other classes .....	43
284	Table 54 – Attributes of	
285	SteadyStateHypothesisScheduleProfile::RegulatingControlSchedule .....	43
286	Table 55 – Association ends of	
287	SteadyStateHypothesisScheduleProfile::RegulatingControlSchedule with other classes .....	43
288	Table 56 – Attributes of	
289	SteadyStateHypothesisScheduleProfile::RegulatingControlTimePoint .....	43
290	Table 57 – Association ends of	
291	SteadyStateHypothesisScheduleProfile::RegulatingControlTimePoint with other	
292	classes .....	44
293	Table 58 – Attributes of	
294	SteadyStateHypothesisScheduleProfile::TapChangerControlSchedule .....	44
295	Table 59 – Association ends of	
296	SteadyStateHypothesisScheduleProfile::TapChangerControlSchedule with other	
297	classes .....	44
298	Table 60 – Attributes of SteadyStateHypothesisScheduleProfile::VsConverterSchedule .....	45
299	Table 61 – Association ends of	
300	SteadyStateHypothesisScheduleProfile::VsConverterSchedule with other classes .....	45
301	Table 62 – Attributes of SteadyStateHypothesisScheduleProfile::ACDCTimePoint .....	45
302	Table 63 – Attributes of	
303	SteadyStateHypothesisScheduleProfile::VsConverterTimePoint .....	46
304	Table 64 – Association ends of	
305	SteadyStateHypothesisScheduleProfile::VsConverterTimePoint with other classes .....	46
306	Table 65 – Attributes of	
307	SteadyStateHypothesisScheduleProfile::CsConverterTimePoint .....	47

308	Table 66 – Association ends of	
309	SteadyStateHypothesisScheduleProfile::CsConverterTimePoint with other classes .....	47
310	Table 67 – Attributes of SteadyStateHypothesisScheduleProfile::CsConverterSchedule .....	47
311	Table 68 – Association ends of	
312	SteadyStateHypothesisScheduleProfile::CsConverterSchedule with other classes .....	48
313	Table 69 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchSchedule .....	48
314	Table 70 – Association ends of	
315	SteadyStateHypothesisScheduleProfile::SwitchSchedule with other classes .....	48
316	Table 71 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchTimePoint .....	48
317	Table 72 – Association ends of	
318	SteadyStateHypothesisScheduleProfile::SwitchTimePoint with other classes .....	49
319	Table 73 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceSchedule .....	49
320	Table 74 – Association ends of	
321	SteadyStateHypothesisScheduleProfile::InServiceSchedule with other classes .....	49
322	Table 75 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceTimePoint .....	49
323	Table 76 – Association ends of	
324	SteadyStateHypothesisScheduleProfile::InServiceTimePoint with other classes .....	50
325	Table 77 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaSchedule .....	50
326	Table 78 – Association ends of	
327	SteadyStateHypothesisScheduleProfile::ControlAreaSchedule with other classes .....	50
328	Table 79 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint .....	50
329	Table 80 – Association ends of	
330	SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint with other classes .....	51
331	Table 81 – Attributes of	
332	SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule .....	51
333	Table 82 – Association ends of	
334	SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule with other	
335	classes .....	51
336	Table 83 – Attributes of	
337	SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint .....	51
338	Table 84 – Association ends of	
339	SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint with other	
340	classes .....	52
341	Table 85 – Attributes of	
342	SteadyStateHypothesisScheduleProfile::AsynchronousMachineSchedule .....	52
343	Table 86 – Association ends of	
344	SteadyStateHypothesisScheduleProfile::AsynchronousMachineSchedule with other	
345	classes .....	52
346	Table 87 – Attributes of	
347	SteadyStateHypothesisScheduleProfile::AsynchronousMachineTimePoint .....	53
348	Table 88 – Association ends of	
349	SteadyStateHypothesisScheduleProfile::AsynchronousMachineTimePoint with other	
350	classes .....	53
351	Table 89 – Attributes of	
352	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionSchedule .....	53

353	Table 90 – Association ends of	
354	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionSchedule with other	
355	classes .....	53
356	Table 91 – Attributes of	
357	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint .....	54
358	Table 92 – Association ends of	
359	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint with other	
360	classes .....	54
361	Table 93 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule .....	54
362	Table 94 – Association ends of	
363	SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule with other classes .....	54
364	Table 95 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint .....	55
365	Table 96 – Association ends of	
366	SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint with other classes .....	55
367	Table 97 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule .....	55
368	Table 98 – Association ends of	
369	SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule with other classes .....	55
370	Table 99 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint .....	56
371	Table 100 – Association ends of	
372	SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint with other classes .....	56
373	Table 101 – Attributes of	
374	SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule .....	56
375	Table 102 – Association ends of	
376	SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule with other classes.....	57
377	Table 103 – Attributes of	
378	SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint .....	57
379	Table 104 – Association ends of	
380	SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint with other classes .....	57
381	Table 105 – Attributes of	
382	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule .....	57
383	Table 106 – Association ends of	
384	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule with other	
385	classes .....	58
386	Table 107 – Attributes of	
387	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint.....	58
388	Table 108 – Association ends of	
389	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint with other	
390	classes .....	58
391	Table 109 – Attributes of	
392	SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule .....	58
393	Table 110 – Association ends of	
394	SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule with other classes .....	59
395	Table 111 – Attributes of	
396	SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint .....	59
397	Table 112 – Association ends of	
398	SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint with other classes .....	59

399	Table 113 – Attributes of	
400	SteadyStateHypothesisScheduleProfile::ShuntCompensatorSchedule .....	59
401	Table 114 – Association ends of	
402	SteadyStateHypothesisScheduleProfile::ShuntCompensatorSchedule with other	
403	classes .....	60
404	Table 115 – Attributes of	
405	SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint .....	60
406	Table 116 – Association ends of	
407	SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint with other	
408	classes .....	60
409	Table 117 – Attributes of	
410	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule .....	61
411	Table 118 – Association ends of	
412	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule with other	
413	classes .....	61
414	Table 119 – Attributes of	
415	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint .....	61
416	Table 120 – Association ends of	
417	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint with other	
418	classes .....	61
419	Table 121 – Attributes of	
420	SteadyStateHypothesisScheduleProfile::GeneratingUnitSchedule .....	62
421	Table 122 – Association ends of	
422	SteadyStateHypothesisScheduleProfile::GeneratingUnitSchedule with other classes .....	62
423	Table 123 – Attributes of	
424	SteadyStateHypothesisScheduleProfile::GeneratingUnitTimePoint .....	62
425	Table 124 – Association ends of	
426	SteadyStateHypothesisScheduleProfile::GeneratingUnitTimePoint with other classes .....	63
427	Table 125 – Attributes of SteadyStateHypothesisScheduleProfile::ActivePower .....	63
428	Table 126 – Literals of SteadyStateHypothesisScheduleProfile::UnitMultiplier .....	64
429	Table 127 – Literals of SteadyStateHypothesisScheduleProfile::UnitSymbol .....	64
430	Table 128 – Attributes of SteadyStateHypothesisScheduleProfile::ReactivePower .....	65
431	Table 129 – Attributes of SteadyStateHypothesisScheduleProfile::Voltage .....	65
432	Table 130 – Attributes of SteadyStateHypothesisScheduleProfile::ApparentPower .....	65
433	Table 131 – Literals of	
434	SteadyStateHypothesisScheduleProfile::AsynchronousMachineKind.....	66
435	Table 132 – Literals of SteadyStateHypothesisScheduleProfile::DayOfWeekKind .....	66
436	Table 133 – Literals of SteadyStateHypothesisScheduleProfile::BatteryStateKind.....	66
437	Table 134 – Attributes of SteadyStateHypothesisScheduleProfile::RealEnergy .....	67
438	Table 135 – Literals of SteadyStateHypothesisScheduleProfile::CsOperatingModeKind.....	67
439	Table 136 – Literals of SteadyStateHypothesisScheduleProfile::CsPpccControlKind.....	67
440	Table 137 – Attributes of SteadyStateHypothesisScheduleProfile::AngleDegrees .....	67
441	Table 138 – Attributes of SteadyStateHypothesisScheduleProfile::CurrentFlow .....	68
442	Table 139 – Literals of SteadyStateHypothesisScheduleProfile::PeakKind .....	68
443	Table 140 – Literals of SteadyStateHypothesisScheduleProfile::EnergyDemandKind .....	68

444	Table 141 – Literals of	
445	SteadyStateHypothesisScheduleProfile::SynchronousMachineOperatingMode .....	69
446	Table 142 – Attributes of SteadyStateHypothesisScheduleProfile::PU .....	69
447	Table 143 – Attributes of SteadyStateHypothesisScheduleProfile::Resistance .....	69
448	Table 144 – Literals of SteadyStateHypothesisScheduleProfile::VsPpccControlKind .....	69
449		

## 450 1 Introduction

451 The steady state hypothesis schedule profile enables an exchange of schedules of operating  
452 point (steady state hypothesis).

## 453 2 Application profile specification

### 454 2.1 Version information

455 The content is generated from UML model file CIM17-2\_CGMES31v01\_PROF-  
456 20v02\_NC23v65\_MS10v01\_DES10v01.eap.

457 This edition is based on the IEC 61970 UML version 'IEC61970CIM17v40', dated '2020-08-24'.

- 458 - Title: Steady State Hypothesis Schedule Vocabulary
- 459 - Keyword: SHS
- 460 - Description: This vocabulary is describing the steady state hypothesis schedule.
- 461 - Version IRI: <https://ap-voc.cim4.eu/SteadyStateHypothesisSchedule/1.0>
- 462 - Version info: 1.0.1
- 463 - Prior version:
- 464 - Conforms to: urn:iso:std:iec:61970-600-2:ed-1|urn:iso:std:iec:61970-301:ed-  
465 7:amd1|file://iec61970cim17v40\_iec61968cim13v13a\_iec62325cim03v17a.eap|urn:iso:  
466 std:iec:61970-401:draft:ed-1|urn:iso:std:iec:61970-501:draft:ed-  
467 2|file://CIM100\_CGMES31v01\_501-20v02\_NC23v62\_MM10v01.eap
- 468 - Identifier: urn:uuid:0d815deb-9968-4c6f-85d7-503d49e0b81f

469

### 470 2.2 Constraints naming convention

471 The naming of the rules shall not be used for machine processing. The rule names are just a  
472 string. The naming convention of the constraints is as follows.

473 "{rule.Type}:{rule.Standard}:{rule.Profile}:{rule.Property}:{rule.Name}"

474 where

475 rule.Type: C – for constraint; R – for requirement

476 rule.Standard: the number of the standard e.g. 301 for 61970-301, 456 for 61970-456, 13 for  
477 61968-13. 61970-600 specific constraints refer to 600 although they are related to one or  
478 combination of the 61970-450 series profiles. For NC profiles, NC is used.

479 rule.Profile: the abbreviation of the profile, e.g. TP for Topology profile. If set to "ALL" the  
480 constraint is applicable to all IEC 61970-600 profiles.

481 rule.Property: for UML classes, the name of the class, for attributes and associations, the name  
482 of the class and attribute or association end, e.g. EnergyConsumer, IdentifiedObject.name, etc.  
483 If set to "NA" the property is not applicable to a specific UML element.

484 rule.Name: the name of the rule. It is unique for the same property.

485 Example: C:600:ALL:IdentifiedObject.name:stringLength

## 486 2.3 Profile constraints

487 This clause defines requirements and constraints that shall be fulfilled by applications that  
488 conform to this document.

489 This document is the master for rules and constraints tagged "NC". For the sake of self-  
490 containment, the list below also includes a copy of the relevant rules from IEC 61970-452,  
491 tagged "452".

- 492 • C:452:ALL:NA:datatypes

493 According to 61970-501, datatypes are not exchanged in the instance data. The  
494 UnitMultiplier is 1 in cases none value is specified in the profile.

- 495 • R:452:ALL:NA:exchange

496 Optional and required attributes and associations must be imported and exported if they  
497 are in the model file prior to import.

- 498 • R:452:ALL:NA:exchange1

499 If an optional attribute does not exist in the imported file, it does not have to be exported  
500 in case exactly the same data set is exported, i.e. the tool is not obliged to automatically  
501 provide this attribute. If the export is resulting from an action by the user performed after  
502 the import, e.g. data processing or model update the export can contain optional  
503 attributes.

- 504 • R:452:ALL:NA:exchange2

505 In most of the profiles the selection of optional and required attributes is made so as to  
506 ensure a minimum set of required attributes without which the exchange does not fulfil  
507 its basic purpose. Business processes governing different exchanges can require  
508 mandatory exchange of certain optional attributes or associations. Optional and required  
509 attributes and associations shall therefore be supported by applications which claim  
510 conformance with certain functionalities of the IEC 61970-452. This provides flexibility  
511 for the business processes to adapt to different business requirements and base the  
512 exchanges on IEC 61970-452 compliant applications.

- 513 • R:452:ALL:NA:exchange3

514 An exporter may, at his or her discretion, produce a serialization containing additional  
515 class data described by the CIM Schema but not required by this document provided  
516 these data adhere to the conventions established in Clause 5.

- 517 • R:452:ALL:NA:exchange4

518 From the standpoint of the model import used by a data recipient, the document  
519 describes a subset of the CIM that importing software shall be able to interpret in order  
520 to import exported models. Data providers are free to exceed the minimum requirements  
521 described herein as long as their resulting data files are compliant with the CIM Schema  
522 and the conventions established in Clause 5. The document, therefore, describes  
523 additional classes and class data that, although not required, exporters will, in all  
524 likelihood, choose to include in their data files. The additional classes and data are  
525 labelled as required (cardinality 1..1) or as optional (cardinality 0..1) to distinguish them  
526 from their required counterparts. Please note, however, that data importers could  
527 potentially receive data containing instances of any and all classes described by the  
528 CIM Schema.

- 529 • R:452:ALL:NA:cardinality



- 530 The cardinality defined in the CIM model shall be followed, unless a more restrictive  
531 cardinality is explicitly defined in this document. For instance, the cardinality on the  
532 association between VoltageLevel and BaseVoltage indicates that a VoltageLevel shall  
533 be associated with one and only one BaseVoltage, but a BaseVoltage can be associated  
534 with zero to many VoltageLevels.
- 535 • R:452:ALL:NA:associations
- 536 Associations between classes referenced in this document and classes not referenced  
537 here are not required regardless of cardinality.
- 538 • R:452:ALL:IdentifiedObject.name:rule
- 539 The attribute “name” inherited by many classes from the abstract class IdentifiedObject  
540 is not required to be unique. It must be a human readable identifier without additional  
541 embedded information that would need to be parsed. The attribute is used for purposes  
542 such as User Interface and data exchange debugging. The MRID defined in the data  
543 exchange format is the only unique and persistent identifier used for this data exchange.  
544 The attribute IdentifiedObject.name is, however, always required for CoreEquipment  
545 profile and Short Circuit profile.
- 546 • R:452:ALL:IdentifiedObject.description:rule
- 547 The attribute “description” inherited by many classes from the abstract class  
548 IdentifiedObject must contain human readable text without additional embedded  
549 information that would need to be parsed.
- 550 • R:452:ALL:NA:uniqueIdentifier
- 551 All IdentifiedObject-s shall have a persistent and globally unique identifier (Master  
552 Resource Identifier - mRID).
- 553 • R:452:ALL:NA:unitMultiplier
- 554 For exchange of attributes defined using CIM Data Types (ActivePower, Susceptance,  
555 etc.) a unit multiplier of 1 is used if the UnitMultiplier specified in this document is “none”.
- 556 • C:452:ALL:IdentifiedObject.name:stringLength
- 557 The string IdentifiedObject.name has a maximum of 128 characters.
- 558 • C:452:ALL:IdentifiedObject.description:stringLength
- 559 The string IdentifiedObject.description is maximum 256 characters.
- 560 • C:452:ALL:NA:float
- 561 An attribute that is defined as float (e.g. has a type Float or a type which is a Datatype  
562 with .value attribute of type Float) shall support ISO/IEC 60559:2020 for floating-point  
563 arithmetic using single precision floating point. A single precision float supports 7  
564 significant digits where the significant digits are described as an integer, or a decimal  
565 number with 6 decimal digits. Two float values are equal when the significant with 7  
566 digits are identical, e.g. 1234567 is equal 1.234567E6 and so are 1.2345678 and  
567 1.234567E0.
- 568 • R:NC:ALL:NA:serialization
- 569 The profiles are defined in the EnterpriseArchitect application and have multiple artifacts  
570 that describe them. The main artifacts are:



- 571 1) the EAP file (EnterpriseArchitect project file),  
572 2) the profiles' specification document and  
573 3) the application profiles (RDFS and SHACL).

574 Due to the complexity of the profiles, there are various cross profile associations that,  
575 from profiling and profile maintenance point of view, it is not practical to include the  
576 complete inheritance structure in all profiles. If this is done the documentation provided  
577 for all profiles would also include duplicated information on the description of classes  
578 defined in other profiles. The following cases are often observed in profiles:

- 579 ○ Case 1: An association end refers to an abstract class
- 580 ○ Case 2: An abstract class (stereotyped with "Description") has an association  
581 (direction to another class)
- 582 ○ Case 3: An abstract class (not stereotyped with "Description") has an  
583 association (direction to another class)
- 584 ○ Case 4: An abstract class has attributes and subclasses are not in the profile

585 In all cases, the datasets shall only include the subtypes of the abstract classes with  
586 the related properties (i.e. association or attributes) defined in the profile. The  
587 information is taken from either canonical model or the profiles where complete  
588 (expected) inheritance structure for the related abstract class is described. SHACL  
589 based constraints include constraints only for the concrete classes that are subtypes of  
590 the abstract class in the profile, and this can be used to inform which are the concrete  
591 classes expected in a dataset that conforms to this profile.

592 It should be taken into account that this approach deviates from MVAL5 (IEC 61970-  
593 600-1:2021), which creates multiple inheritance at serialization. For instance, with this  
594 more explicit exchange the serialization of the association between abstract class  
595 Equipment and abstract class Circuit for a PowerTransformer will be serialized as  
596 follows:

- 597 ○ for association
- ```
598 <cim:PowerTransformer rdf:about="_c328f787-bc17-47ad-a59f-6ba7133340d0">
599   <nc:Equipment.Circuit rdf:resource="#_9ced16ac-d076-4ef9-a241-a998a579e77b"/>
600 </cim:PowerTransformer>
```
- 601 ○ for attribute
- ```
602 <cim:ACLineSegment rdf:about="_04f681aa-6999-4fb3-9775-aca5eb7ceff">
603   <cim:Equipment.inService>true</cim:Equipment.inService>
604 </cim:ACLineSegment>
```

605 The usage of rdf:ID or rdf:about depends on the stereotype of the class. rdf:about is  
606 used if the class has the stereotype "Description".

607 An example of not allowed serialization, as the Equipment is an abstract class

```
608 <cim:Equipment rdf:about="_c328f787-bc17-47ad-a59f-6ba7133340d0">
609   <nc:Equipment.Circuit rdf:resource="#_9ced16ac-d076-4ef9-a241-a998a579e77b"/>
610 </cim:Equipment>
```

## 611 2.4 Metadata

612 ENTSO-E agreed to extend the header and metadata definitions by IEC 61970-552 Ed2. This  
613 new header definitions rely on W3C recommendations which are used worldwide and are

614 positively recognized by the European Commission. The new definitions of the header mainly  
615 use Provenance ontology (PROV-O), Time Ontology and Data Catalog Vocabulary (DCAT). The  
616 global new header applicable for this profile is included in the metadata and document header  
617 specification document.

618 The header vocabulary contains all attributes defined in IEC 61970-552. This is done only for  
619 the purpose of having one vocabulary for header and to ensure transition for data exchanges  
620 that are using IEC 61970-552:2016 header. This profile does not use IEC 61970-552:2016  
621 header attributes and relies only on the extended attributes.

### 622 2.4.1 Constraints

623 The identification of the constraints related to the metadata follows the same convention for  
624 naming of the constraints as for profile constraints.

- 625 • R:NC:ALL:wasAttributedTo:usage

626 The prov:wasAttributedTo should normally be the “X” EIC code of the actor or their URI  
627 (prov:Agent).

628

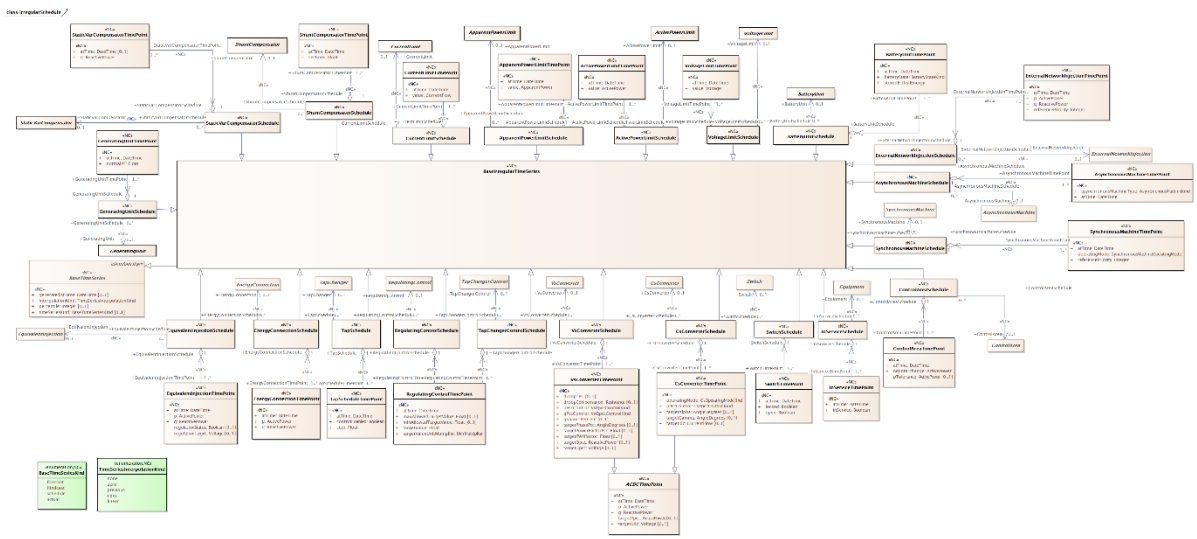
### 629 2.4.2 Reference metadata

630 The header defined for this profile requires availability of a set of reference metadata. For  
631 instance, the attribute prov:wasGeneratedBy requires a reference to an activity which produced  
632 the model or the related process. The activities are defined as reference metadata and their  
633 identifiers are referenced from the header to enable the receiving entity to retrieve the “static”  
634 (reference) information that is not modified frequently. This approach imposes a requirement  
635 that both the sending entity and the receiving entity have access to a unique version of the  
636 reference metadata. Therefore, each business process shall define which reference metadata  
637 is used and where it is located.

## 638 3 Package SteadyStateHypothesisScheduleProfile

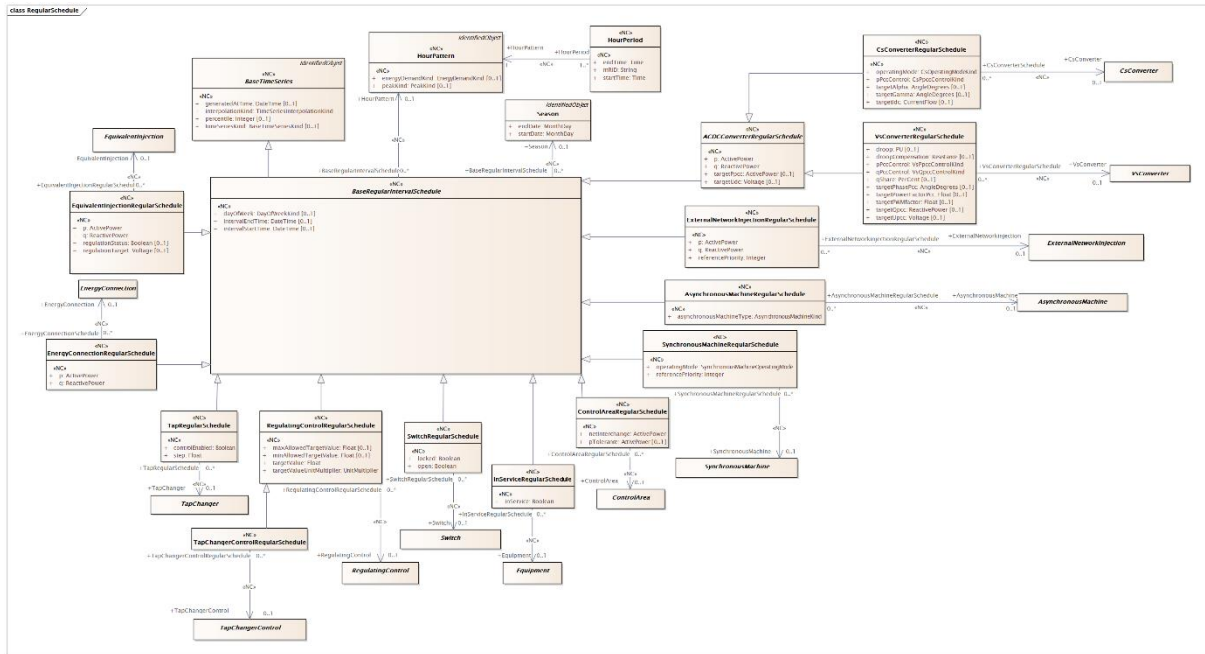
### 639 3.1 General

640 This package contains steady state hypothesis schedule profile.

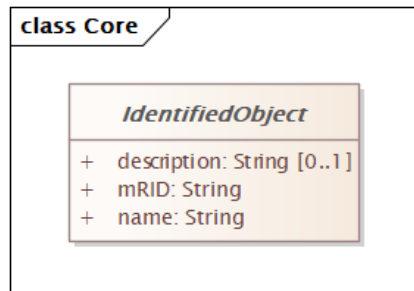


641  
642 **Figure 1 – Class diagram SteadyStateHypothesisScheduleProfile::IrregularSchedule**

643 Figure 1: The diagram shows classes related to the irregular schedule.



644  
 645 **Figure 2 – Class diagram SteadyStateHypothesisScheduleProfile::RegularSchedule**  
 646 Figure 2: The diagram shows classes related to the regular schedule.



647  
 648 **Figure 3 – Class diagram SteadyStateHypothesisScheduleProfile::Core**

649 Figure 3: The diagram shows classes from Base CIM used in the profile.

650 **3.2 (abstract) IdentifiedObject root class**

651 This is a root class to provide common identification for all classes needing identification and  
 652 naming attributes.

653 Table 1 shows all attributes of IdentifiedObject.

654 **Table 1 – Attributes of SteadyStateHypothesisScheduleProfile::IdentifiedObject**

name	mult	type	description
description	0..1	<a href="#">String</a>	The description is a free human readable text describing or naming the object. It may be non unique and may not correlate to a naming hierarchy.
mRID	1..1	<a href="#">String</a>	Master resource identifier issued by a model authority. The mRID is unique within an exchange context. Global uniqueness is easily achieved by using a UUID, as specified in RFC 4122, for the mRID. The use of UUID is strongly recommended.

name	mult	type	description
			For CIMXML data files in RDF syntax conforming to IEC 61970-552, the mRID is mapped to rdf:ID or rdf:about attributes that identify CIM object elements.
name	1..1	<a href="#">String</a>	The name is any free human readable and possibly non unique text naming the object.

655

656 **3.3 (abstract,NC) BaseTimeSeries**657 Inheritance path = [IdentifiedObject](#)

658 Time series of values at points in time.

659 Table 2 shows all attributes of BaseTimeSeries.

660

**Table 2 – Attributes of SteadyStateHypothesisScheduleProfile::BaseTimeSeries**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) Kind of interpolation done between time point.
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) Kind of base time series.
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) The time this time series (entity) come to existents and available for use.
percentile	0..1	<a href="#">Integer</a>	(NC) The percentile is a number where a certain percentage of scores/ranking/values of a sample fall below that number. This is a way for expressing uncertainty in the number provided.
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

661

662 **3.4 Season**663 Inheritance path = [IdentifiedObject](#)

664 A specified time period of the year.

665 Table 3 shows all attributes of Season.

666

**Table 3 – Attributes of SteadyStateHypothesisScheduleProfile::Season**

name	mult	type	description
endDate	1..1	<a href="#">MonthDay</a>	Date season ends.
startDate	1..1	<a href="#">MonthDay</a>	Date season starts.
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

667

668 **3.5 (NC) HourPattern**669 Inheritance path = [IdentifiedObject](#)

670 Pattern of hourly period in a day with the same kind of intensity.

671 Table 4 shows all attributes of HourPattern.

672

**Table 4 – Attributes of SteadyStateHypothesisScheduleProfile::HourPattern**

name	mult	type	description
peakKind	0..1	<a href="#">PeakKind</a>	(NC) Type of peak or intensity that the pattern is valid for.
energyDemandKind	0..1	<a href="#">EnergyDemandKind</a>	(NC) Type of energy demand that the pattern is valid for.
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

673

**3.6 (abstract,NC) BaseRegularIntervalSchedule**675 Inheritance path = [BaseTimeSeries](#) : [IdentifiedObject](#)

676 Time series that has regular points in time.

677 Table 5 shows all attributes of BaseRegularIntervalSchedule.

678

679

**Table 5 – Attributes of SteadyStateHypothesisScheduleProfile::BaseRegularIntervalSchedule**

name	mult	type	description
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) Day of the week for which the schedule is valid for.
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) Interval start time for which the schedule is valid for.
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) Interval end time for which the schedule is valid for.
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

680

681 Table 6 shows all association ends of BaseRegularIntervalSchedule with other classes.

682

683

684

**Table 6 – Association ends of SteadyStateHypothesisScheduleProfile::BaseRegularIntervalSchedule with other classes**

mult from	name	mult to	type	description
0..*	Season	0..1	<a href="#">Season</a>	(NC) Season associated with a base regular interval schedule.
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) HourPattern that has base regular interval schedule.

685

**3.7 (NC) HourPeriod root class**

687 Period of hours in a day.

688 Table 7 shows all attributes of HourPeriod.

689 **Table 7 – Attributes of SteadyStateHypothesisScheduleProfile::HourPeriod**

name	mult	type	description
mRID	1..1	<a href="#">String</a>	(NC) Master resource identifier issued by a model authority. The mRID is unique within an exchange context. Global uniqueness is easily achieved by using a UUID, as specified in RFC 4122, for the mRID. The use of UUID is strongly recommended.  For CIMXML data files in RDF syntax conforming to IEC 61970-552, the mRID is mapped to rdf:ID or rdf:about attributes that identify CIM object elements.
startTime	1..1	<a href="#">Time</a>	(NC) Time the period start and including, e.g. 12:00 which means it include the time of 12:00.
endTime	1..1	<a href="#">Time</a>	(NC) Time the period end and not including, e.g. 13:00 which means it does not include the time of 13:00 but 12:59.

690

691

Table 8 shows all association ends of HourPeriod with other classes.

692

693

**Table 8 – Association ends of SteadyStateHypothesisScheduleProfile::HourPeriod with other classes**

mult from	name	mult to	type	description
1..*	HourPattern	1..1	<a href="#">HourPattern</a>	(NC) HourPattern which has some hour periods.

694

### 695 3.8 (NC) BaselrregularTimeSeries

696 Inheritance path = [BaseTimeSeries](#) : [IdentifiedObject](#)

697 Time series that has irregular points in time.

698 Table 9 shows all attributes of BaselrregularTimeSeries.

699 **Table 9 – Attributes of SteadyStateHypothesisScheduleProfile::BaselrregularTimeSeries**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

700

### 701 3.9 (NC) BaseTimeSeriesKind enumeration

702 Kind of time series.

703 Table 10 shows all literals of BaseTimeSeriesKind.

704 **Table 10 – Literals of SteadyStateHypothesisScheduleProfile::BaseTimeSeriesKind**

literal	value	description
forecast		Time series is forecast data. The values represent the result of scientific predictions based on historical time stamped data.
hindcast		Time series is hindcast data. The value represent probable past (historic) condition given by calculation done using actual values. For instance, determine the among of wind based on the energy produced by wind. However, hindcast is typical the result of a simulated forecasts for historical periods.
schedule		Time series is schedule data. The values represent the result of a committed and plan forecast data that has been through a quality control and could incur penalty when not followed.
actual		Time series is actual data. The values represent measured or calculated values that represent the actual behaviour.

705

706 **3.10 (NC) TimeSeriesInterpolationKind enumeration**

707 Kinds of interpolation of values between two time point.

708 Table 11 shows all literals of TimeSeriesInterpolationKind.

709

710

**Table 11 – Literals of  
SteadyStateHypothesisScheduleProfile::TimeSeriesInterpolationKind**

literal	value	description
none		No interpolation is applied.
zero		The value between two time points is set to zero.
previous		The value between two time points is set to previous value.
next		The value between two time points is set to next value.
linear		Linear interpolation is applied for values between two time points.

711

712 **3.11 (NC) RegulatingControlRegularSchedule**713 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

714 Regular schedule for regulating control.

715 Table 12 shows all attributes of RegulatingControlRegularSchedule.

716

717

**Table 12 – Attributes of  
SteadyStateHypothesisScheduleProfile::RegulatingControlRegularSchedule**

name	mult	type	description
targetValue	1..1	<a href="#">Float</a>	(NC) The target value specified for case input. This value can be used for the target value without the use of schedules. The value has the units appropriate to the mode attribute.
targetValueUnitMultiplier	1..1	<a href="#">UnitMultiplier</a>	(NC) Specify the multiplier for used for the targetValue.
maxAllowedTargetValue	0..1	<a href="#">Float</a>	(NC) Maximum allowed target value (RegulatingControl.targetValue).



name	mult	type	description
minAllowedTargetValue	0..1	<a href="#">Float</a>	(NC) Minimum allowed target value (RegulatingControl.targetValue).
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

718  
719  
720  
721  
722

Table 13 shows all association ends of RegulatingControlRegularSchedule with other classes.

**Table 13 – Association ends of SteadyStateHypothesisScheduleProfile::RegulatingControlRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	RegulatingControl	0..1	<a href="#">RegulatingControl</a>	(NC) Regulating control which has RegulatingControlRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

723  
724

### 3.12 (abstract) RegulatingControl root class

725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742

Specifies a set of equipment that works together to control a power system quantity such as voltage or flow.

Remote bus voltage control is possible by specifying the controlled terminal located at some place remote from the controlling equipment.

The specified terminal shall be associated with the connectivity node of the controlled point.

The most specific subtype of RegulatingControl shall be used in case such equipment participate in the control, e.g. TapChangerControl for tap changers.

For flow control, load sign convention is used, i.e. positive sign means flow out from a TopologicalNode (bus) into the conducting equipment.

The attribute minAllowedTargetValue and maxAllowedTargetValue are required in the following cases:

- For a power generating module operated in power factor control mode to specify maximum and minimum power factor values;

- Whenever it is necessary to have an off center target voltage for the tap changer regulator.

For instance, due to long cables to off shore wind farms and the need to have a simpler setup at the off shore transformer platform, the voltage is controlled from the land at the connection point for the off shore wind farm. Since there usually is a voltage rise along the cable, there is typical and overvoltage of up 3-4 kV compared to the on shore station. Thus in normal operation



743 the tap changer on the on shore station is operated with a target set point, which is in the lower  
744 parts of the dead band.

745 The attributes minAllowedTargetValue and maxAllowedTargetValue are not related to the  
746 attribute targetDeadband and thus they are not treated as an alternative of the targetDeadband.  
747 They are needed due to limitations in the local substation controller. The attribute  
748 targetDeadband is used to prevent the power flow from move the tap position in circles (hunting)  
749 that is to be used regardless of the attributes minAllowedTargetValue and  
750 maxAllowedTargetValue.

### 751 3.13 (NC) TapChangerControlRegularSchedule

752 Inheritance path = [RegulatingControlRegularSchedule](#) : [BaseRegularIntervalSchedule](#) :  
753 [BaseTimeSeries](#) : [IdentifiedObject](#)

754 Regular schedule for tap changer control.

755 Table 14 shows all attributes of TapChangerControlRegularSchedule.

756 **Table 14 – Attributes of**  
757 **SteadyStateHypothesisScheduleProfile::TapChangerControlRegularSchedule**

name	mult	type	description
targetValue	1..1	<a href="#">Float</a>	(NC) inherited from: <a href="#">RegulatingControlRegularSchedule</a>
targetValueUnitMultiplier	1..1	<a href="#">UnitMultiplier</a>	(NC) inherited from: <a href="#">RegulatingControlRegularSchedule</a>
maxAllowedTargetValue	0..1	<a href="#">Float</a>	(NC) inherited from: <a href="#">RegulatingControlRegularSchedule</a>
minAllowedTargetValue	0..1	<a href="#">Float</a>	(NC) inherited from: <a href="#">RegulatingControlRegularSchedule</a>
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

758

759 Table 15 shows all association ends of TapChangerControlRegularSchedule with other classes.

760 **Table 15 – Association ends of**  
761 **SteadyStateHypothesisScheduleProfile::TapChangerControlRegularSchedule with other**  
762 **classes**

mult from	name	mult to	type	description
0..*	TapChangerControl	0..1	<a href="#">TapChangerControl</a>	(NC) Tap changer control which has TapChangerControlRegularSchedule.
0..*	RegulatingControl	0..1	<a href="#">RegulatingControl</a>	(NC) inherited from: <a href="#">RegulatingControlRegularSchedule</a>

mult from	name	mult to	type	description
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

763

764 **3.14 (abstract) TapChangerControl root class**

765 Describes behaviour specific to tap changers, e.g. how the voltage at the end of a line varies  
766 with the load level and compensation of the voltage drop by tap adjustment.

767 **3.15 VsQpccControlKind enumeration**

768 Kind of reactive power control at point of common coupling for a voltage source converter.  
769 Table 16 shows all literals of VsQpccControlKind.

770 **Table 16 – Literals of SteadyStateHypothesisScheduleProfile::VsQpccControlKind**

literal	value	description
reactivePcc		Control is reactive power at point of common coupling. Target is provided by VsConverter.targetQpcc.
voltagePcc		Control is voltage at point of common coupling. Target is provided by VsConverter.targetUpcc.
powerFactorPcc		Control is power factor at point of common coupling. Target is provided by VsConverter.targetPowerFactorPcc.
pulseWidthModulation		No explicit control. Pulse-modulation factor is directly set in magnitude (VsConverter.targetPWMfactor) and phase (VsConverter.targetPhasePcc).

771

772 **3.16 PerCent datatype**

773 Percentage on a defined base. For example, specify as 100 to indicate at the defined base.  
774 Table 17 shows all attributes of PerCent.

775 **Table 17 – Attributes of SteadyStateHypothesisScheduleProfile::PerCent**

name	mult	type	description
value	0..1	<a href="#">Float</a>	Normally 0 to 100 on a defined base.
unit	0..1	<a href="#">UnitSymbol</a>	(const=none)
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=none)

776

777 **3.17 (NC) EnergyConnectionRegularSchedule**

778 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)  
779 Regular schedule for energy connection.

780 Table 18 shows all attributes of EnergyConnectionRegularSchedule.

781 **Table 18 – Attributes of**  
782 **SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule**

name	mult	type	description
p	1..1	<a href="#">ActivePower</a>	(NC) Active power injection. Load sign convention is used, i.e. positive sign means flow out from a node.

name	mult	type	description
			Starting value for a steady state solution.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

783

784

Table 19 shows all association ends of EnergyConnectionRegularSchedule with other classes.

785

786

787

**Table 19 – Association ends of  
SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	EnergyConnection	0..1	<a href="#">EnergyConnection</a>	(NC) EnergyConnection which has EnergyConnectionSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

788

### 789 3.18 (abstract) EnergyConnection root class

790 A connection of energy generation or consumption on the power system model.

### 791 3.19 (NC) TapRegularSchedule

792 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

793 Regular schedule for tap.

794 Table 20 shows all attributes of TapRegularSchedule.

795 **Table 20 – Attributes of SteadyStateHypothesisScheduleProfile::TapRegularSchedule**

name	mult	type	description
controlEnabled	1..1	<a href="#">Boolean</a>	(NC) Specifies the regulation status of the equipment. True is regulating, false is not regulating.
step	1..1	<a href="#">Float</a>	(NC) Tap changer position. Starting step for a steady state solution. Non integer values are allowed to support continuous tap variables. The reasons for continuous value

name	mult	type	description
			are to support study cases where no discrete tap changer has yet been designed, a solution where a narrow voltage band forces the tap step to oscillate or to accommodate for a continuous solution as input. The attribute shall be equal to or greater than lowStep and equal to or less than highStep.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

796

797

Table 21 shows all association ends of TapRegularSchedule with other classes.

798

799

**Table 21 – Association ends of SteadyStateHypothesisScheduleProfile::TapRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	TapChanger	0..1	<a href="#">TapChanger</a>	(NC) Tap changer which has TapRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

800

### 801 3.20 (abstract) TapChanger root class

802 Mechanism for changing transformer winding tap positions.

### 803 3.21 (NC) SwitchRegularSchedule

804 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

805 Regular schedule for switch.

806 Table 22 shows all attributes of SwitchRegularSchedule.

807

808

**Table 22 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule**

name	mult	type	description
open	1..1	<a href="#">Boolean</a>	(NC) The attribute tells if the switch is considered open when used as input to topology processing.
locked	1..1	<a href="#">Boolean</a>	(NC) If true, the switch is locked. The resulting switch state is a combination of locked and Switch.open attributes as follows:

name	mult	type	description
			- locked=true and Switch.open=true. The resulting state is open and locked; - locked=false and Switch.open=true. The resulting state is open; - locked=false and Switch.open=false. The resulting state is closed.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

809

810 Table 23 shows all association ends of SwitchRegularSchedule with other classes.

811

812

**Table 23 – Association ends of  
SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	Switch	0..1	<a href="#">Switch</a>	(NC) Switch which has SwitchRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

813

### 814 3.22 (abstract) Switch root class

815 A generic device designed to close, or open, or both, one or more electric circuits. All switches  
816 are two terminal devices including grounding switches. The ACDCTerminal.connected at the  
817 two sides of the switch shall not be considered for assessing switch connectivity, i.e. only  
818 Switch.open, .normalOpen and .locked are relevant.

### 819 3.23 (NC) InServiceRegularSchedule

820 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

821 Regular schedule for elements having in service.

822 Table 24 shows all attributes of InServiceRegularSchedule.

823

824

**Table 24 – Attributes of  
SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule**

name	mult	type	description
inService	1..1	<a href="#">Boolean</a>	(NC) Specifies the availability of the equipment. True means the equipment is available for topology processing, which determines if the equipment is energized or not. False means that

name	mult	type	description
			the equipment is treated by network applications as if it is not in the model.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

825

826 Table 25 shows all association ends of InServiceRegularSchedule with other classes.

827

828 **Table 25 – Association ends of SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	Equipment	0..1	<a href="#">Equipment</a>	(NC) Equipment which has InServiceRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

829

830 **3.24 (abstract) Equipment root class**

831 The parts of a power system that are physical devices, electronic or mechanical.

832 **3.25 (NC) ControlAreaRegularSchedule**833 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

834 Regular schedule for control area.

835 Table 26 shows all attributes of ControlAreaRegularSchedule.

836

837 **Table 26 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule**

name	mult	type	description
netInterchange	1..1	<a href="#">ActivePower</a>	(NC) The specified positive net interchange into the control area, i.e. positive sign means flow into the area.
pTolerance	0..1	<a href="#">ActivePower</a>	(NC) Active power net interchange tolerance. The attribute shall be a positive value or zero.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

name	mult	type	description
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

838

839 Table 27 shows all association ends of ControlAreaRegularSchedule with other classes.

840

**Table 27 – Association ends of  
SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule with other  
classes**

841

842

mult from	name	mult to	type	description
0..*	ControlArea	0..1	<a href="#">ControlArea</a>	(NC) ControlArea which has ControlAreaRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

843

### 844 3.26 (abstract) ControlArea root class

845 A control area is a grouping of generating units and/or loads and a cutset of tie lines (as  
846 terminals) which may be used for a variety of purposes including automatic generation control,  
847 power flow solution area interchange control specification, and input to load forecasting. All  
848 generation and load within the area defined by the terminals on the border are considered in  
849 the area interchange control. Note that any number of overlapping control area specifications  
850 can be superimposed on the physical model. The following general principles apply to  
851 ControlArea:

852 1. The control area orientation for net interchange is positive for an import, negative for an  
853 export.

854 2. The control area net interchange is determined by summing flows in Terminals. The  
855 Terminals are identified by creating a set of TieFlow objects associated with a ControlArea  
856 object. Each TieFlow object identifies one Terminal.

857 3. In a single network model, a tie between two control areas must be modelled in both control  
858 area specifications, such that the two representations of the tie flow sum to zero.

859 4. The normal orientation of Terminal flow is positive for flow into the conducting equipment  
860 that owns the Terminal. (i.e. flow from a bus into a device is positive.) However, the orientation  
861 of each flow in the control area specification must align with the control area convention, i.e.  
862 import is positive. If the orientation of the Terminal flow referenced by a TieFlow is positive into  
863 the control area, then this is confirmed by setting TieFlow.positiveFlowIn flag TRUE. If not, the  
864 orientation must be reversed by setting the TieFlow.positiveFlowIn flag FALSE.

### 865 3.27 (NC) SynchronousMachineRegularSchedule

866 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

867 Regular schedule for synchronous machine.

868 Table 28 shows all attributes of SynchronousMachineRegularSchedule.



869  
870**Table 28 – Attributes of  
SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule**

name	mult	type	description
operatingMode	1..1	<a href="#">SynchronousMachineOperatingMode</a>	(NC) Current mode of operation.
referencePriority	1..1	<a href="#">Integer</a>	(NC) Priority of unit for use as powerflow voltage phase angle reference bus selection. 0 = don't care (default) 1 = highest priority. 2 is less than 1 and so on.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

871  
872  
873

Table 29 shows all association ends of SynchronousMachineRegularSchedule with other classes.

874  
875  
876**Table 29 – Association ends of  
SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule with  
other classes**

mult from	name	mult to	type	description
0..*	SynchronousMachine	0..1	<a href="#">SynchronousMachine</a>	(NC) SynchronousMachine which has SynchronousMachineRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

877

**3.28 (abstract) SynchronousMachine root class**

878 An electromechanical device that operates with shaft rotating synchronously with the network.  
879 It is a single machine operating either as a generator or synchronous condenser or pump.  
880

**3.29 (NC) AsynchronousMachineRegularSchedule**

881 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

882 Regular schedule for asynchronous machine.

883 Table 30 shows all attributes of AsynchronousMachineRegularSchedule.  
884



885  
886**Table 30 – Attributes of  
SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule**

name	mult	type	description
asynchronousMachineType	1..1	<a href="#">AsynchronousMachineKind</a>	(NC) Indicates the type of Asynchronous Machine (motor or generator).
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

887  
888  
889  
890  
891  
892

Table 31 shows all association ends of AsynchronousMachineRegularSchedule with other classes.

**Table 31 – Association ends of  
SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule with  
other classes**

mult from	name	mult to	type	description
0..*	AsynchronousMachine	0..1	<a href="#">AsynchronousMachine</a>	(NC) AsynchronousMachine which has AsynchronousMachineRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903**3.30 (abstract) AsynchronousMachine root class**

A rotating machine whose shaft rotates asynchronously with the electrical field. Also known as an induction machine with no external connection to the rotor windings, e.g. squirrel-cage induction machine.

**3.31 (NC) ExternalNetworkInjectionRegularSchedule**

Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

Regular schedule for external network injection.

Table 32 shows all attributes of ExternalNetworkInjectionRegularSchedule.

**Table 32 – Attributes of  
SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule**

name	mult	type	description
referencePriority	1..1	<a href="#">Integer</a>	(NC) Priority of unit for use as powerflow voltage phase angle reference bus selection. 0 = don't

name	mult	type	description
			care (default) 1 = highest priority. 2 is less than 1 and so on.
p	1..1	<a href="#">ActivePower</a>	(NC) Active power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

904  
905  
906

Table 33 shows all association ends of ExternalNetworkInjectionRegularSchedule with other classes.

907  
908  
909

**Table 33 – Association ends of SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	ExternalNetworkInjection	0..1	<a href="#">ExternalNetworkInjection</a>	(NC) External network injection which has ExternalNetworkInjectionRegularSchedule
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

910

### 911 3.32 (abstract) ExternalNetworkInjection root class

912 This class represents the external network and it is used for IEC 60909 calculations.

### 913 3.33 (abstract,NC) ACDCConverterRegularSchedule

914 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

915 Regular schedule for ACDC converter.

916 Table 34 shows all attributes of ACDCConverterRegularSchedule.

917  
918**Table 34 – Attributes of  
SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule**

name	mult	type	description
p	1..1	<a href="#">ActivePower</a>	(NC) Active power at the point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution in the case a simplified power flow model is used.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power at the point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution in the case a simplified power flow model is used.
targetPpcc	0..1	<a href="#">ActivePower</a>	(NC) Real power injection target in AC grid, at point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node.
targetUdc	0..1	<a href="#">Voltage</a>	(NC) Target value for DC voltage magnitude. The attribute shall be a positive value.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

919  
920  
921  
922  
923

Table 35 shows all association ends of ACDCConverterRegularSchedule with other classes.

**Table 35 – Association ends of  
SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

924  
925  
926  
927  
928  
929**3.34 (NC) VsConverterRegularSchedule**

Inheritance path = [ACDCConverterRegularSchedule](#) : [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

Regular schedule for VS converter.

Table 36 shows all attributes of VsConverterRegularSchedule.

930  
931**Table 36 – Attributes of  
SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule**

name	mult	type	description
droop	0..1	<a href="#">PU</a>	Droop constant. The pu value is obtained as $D [kV/MW] \times S_b / U_{bdc}$ . The attribute shall be a positive value.
droopCompensation	0..1	<a href="#">Resistance</a>	Compensation constant. Used to compensate for voltage drop when controlling voltage at a distant bus. The attribute shall be a positive value.
pPccControl	1..1	<a href="#">VsPpccControlKind</a>	Kind of control of real power and/or DC voltage.
qPccControl	1..1	<a href="#">VsQpccControlKind</a>	Kind of reactive power control.
qShare	0..1	<a href="#">PerCent</a>	Reactive power sharing factor among parallel converters on Uac control. The attribute shall be a positive value or zero.
targetQpcc	0..1	<a href="#">ReactivePower</a>	Reactive power injection target in AC grid, at point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node.
targetUpcc	0..1	<a href="#">Voltage</a>	Voltage target in AC grid, at point of common coupling. The attribute shall be a positive value.
targetPowerFactorPcc	0..1	<a href="#">Float</a>	Power factor target at the AC side, at point of common coupling. The attribute shall be a positive value.
targetPhasePcc	0..1	<a href="#">AngleDegrees</a>	Phase target at AC side, at point of common coupling. The attribute shall be a positive value.
targetPWMfactor	0..1	<a href="#">Float</a>	Magnitude of pulse-modulation factor. The attribute shall be a positive value.
p	1..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
q	1..1	<a href="#">ReactivePower</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
targetPpcc	0..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
targetUdc	0..1	<a href="#">Voltage</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

932  
933

Table 37 shows all association ends of VsConverterRegularSchedule with other classes.

934  
935  
936**Table 37 – Association ends of  
SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	VsConverter	0..1	<a href="#">VsConverter</a>	(NC) VsConverter which has VsConverterRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

937

**938 3.35 (NC) CsConverterRegularSchedule**

939 Inheritance path = [ACDCConverterRegularSchedule](#) : [BaseRegularIntervalSchedule](#) :  
940 [BaseTimeSeries](#) : [IdentifiedObject](#)

941 Regular schedule for CS converter.

942 Table 38 shows all attributes of CsConverterRegularSchedule.

943

**Table 38 – Attributes of  
SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule**

944

name	mult	type	description
operatingMode	1..1	<a href="#">CsOperatingModeKind</a>	(NC) Indicates whether the DC pole is operating as an inverter or as a rectifier. It is converter's control variable used in power flow.
pPccControl	1..1	<a href="#">CsPpccControlKind</a>	(NC) Kind of active power control.
targetAlpha	0..1	<a href="#">AngleDegrees</a>	(NC) Target firing angle. It is converter's control variable used in power flow. It is only applicable for rectifier if continuous tap changer control is used. Allowed values are within the range $\text{minAlpha} \leq \text{targetAlpha} \leq \text{maxAlpha}$ . The attribute shall be a positive value.
targetGamma	0..1	<a href="#">AngleDegrees</a>	(NC) Target extinction angle. It is converter's control variable used in power flow. It is only applicable for inverter if continuous tap changer control is used. Allowed values are within the range $\text{minGamma} \leq \text{targetGamma} \leq \text{maxGamma}$ . The attribute shall be a positive value.
targetIdc	0..1	<a href="#">CurrentFlow</a>	(NC) DC current target value. It is converter's control variable used in power flow. The attribute shall be a positive value.
p	1..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
q	1..1	<a href="#">ReactivePower</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
targetPpcc	0..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
targetUdc	0..1	<a href="#">Voltage</a>	(NC) inherited from: <a href="#">ACDCConverterRegularSchedule</a>
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

945

946

Table 39 shows all association ends of CsConverterRegularSchedule with other classes.

947

948

949

**Table 39 – Association ends of  
SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	CsConverter	0..1	<a href="#">CsConverter</a>	(NC) CsConverter which has CsConverterRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

950

951

### 3.36 (abstract) VsConverter root class

952

DC side of the voltage source converter (VSC).

953

### 3.37 (abstract) CsConverter root class

954

DC side of the current source converter (CSC).

955

The firing angle controls the dc voltage at the converter, both for rectifier and inverter. The difference between the dc voltages of the rectifier and inverter determines the dc current. The extinction angle is used to limit the dc voltage at the inverter, if needed, and is not used in active power control. The firing angle, transformer tap position and number of connected filters are the primary means to control a current source dc line. Higher level controls are built on top, e.g. dc voltage, dc current and active power. From a steady state perspective it is sufficient to specify the wanted active power transfer (ACDCConverter.targetPpcc) and the control functions will set the dc voltage, dc current, firing angle, transformer tap position and number of connected filters to meet this. Therefore attributes targetAlpha and targetGamma are not applicable in this case.

965

The reactive power consumed by the converter is a function of the firing angle, transformer tap position and number of connected filter, which can be approximated with half of the active power. The losses is a function of the dc voltage and dc current.

968

The attributes minAlpha and maxAlpha define the range of firing angles for rectifier operation between which no discrete tap changer action takes place. The range is typically 10-18 degrees. The attributes minGamma and maxGamma define the range of extinction angles for inverter operation between which no discrete tap changer action takes place. The range is typically 17-20 degrees.

973

### 3.38 (NC) EquivalentInjectionRegularSchedule

974

Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

975

Regular schedule for equivalent injection.

976

Table 40 shows all attributes of EquivalentInjectionRegularSchedule.

977  
978**Table 40 – Attributes of  
SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule**

name	mult	type	description
regulationStatus	0..1	<a href="#">Boolean</a>	(NC) Specifies the regulation status of the EquivalentInjection. True is regulating. False is not regulating.
regulationTarget	0..1	<a href="#">Voltage</a>	(NC) The target voltage for voltage regulation. The attribute shall be a positive value.
p	1..1	<a href="#">ActivePower</a>	(NC) Equivalent active power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.
q	1..1	<a href="#">ReactivePower</a>	(NC) Equivalent reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.
dayOfWeek	0..1	<a href="#">DayOfWeekKind</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalStartTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
intervalEndTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

979  
980  
981  
982  
983

Table 41 shows all association ends of EquivalentInjectionRegularSchedule with other classes.

**Table 41 – Association ends of  
SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	EquivalentInjection	0..1	<a href="#">EquivalentInjection</a>	(NC) EquivalentInjection which has EquivalentInjectionRegularSchedule.
0..*	Season	0..1	<a href="#">Season</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>
0..*	HourPattern	0..1	<a href="#">HourPattern</a>	(NC) inherited from: <a href="#">BaseRegularIntervalSchedule</a>

984  
985  
986  
987  
988  
989**3.39 (abstract) EquivalentInjection root class**

This class represents equivalent injections (generation or load). Voltage regulation is allowed only at the point of connection.

**3.40 (NC) EquivalentInjectionSchedule**

Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)



990 Regular schedule for equivalent injection.  
991 Table 42 shows all attributes of EquivalentInjectionSchedule.

992 **Table 42 – Attributes of**  
993 **SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

994  
995 Table 43 shows all association ends of EquivalentInjectionSchedule with other classes.

996 **Table 43 – Association ends of**  
997 **SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule with other classes**

mult from	name	mult to	type	description
0..*	EquivalentInjection	0..1	<a href="#">EquivalentInjection</a>	(NC) Equivalent injection which has equivalent injection schedules.

998  
999 **3.41 (NC) EquivalentInjectionTimePoint root class**

1000 Equivalent injection values for a given point in time.  
1001 Table 44 shows all attributes of EquivalentInjectionTimePoint.

1002 **Table 44 – Attributes of**  
1003 **SteadyStateHypothesisScheduleProfile::EquivalentInjectionTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
regulationStatus	0..1	<a href="#">Boolean</a>	(NC) Specifies the regulation status of the EquivalentInjection. True is regulating. False is not regulating.
regulationTarget	0..1	<a href="#">Voltage</a>	(NC) The target voltage for voltage regulation. The attribute shall be a positive value.
p	1..1	<a href="#">ActivePower</a>	(NC) Equivalent active power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.
q	1..1	<a href="#">ReactivePower</a>	(NC) Equivalent reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.

1004  
1005 Table 45 shows all association ends of EquivalentInjectionTimePoint with other classes.



1006  
1007  
1008

**Table 45 – Association ends of  
SteadyStateHypothesisScheduleProfile::EquivalentInjectionTimePoint with other  
classes**

mult from	name	mult to	type	description
1..*	EquivalentInjectionSchedule	1..1	<a href="#">EquivalentInjectionSchedule</a>	(NC) The EquivalentInjection schedule that has this time point.

1009

### 1010 3.42 (NC) EnergyConnectionSchedule

1011 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1012 Schedule for energy connection.

1013 Table 46 shows all attributes of EnergyConnectionSchedule.

1014

1015

**Table 46 – Attributes of  
SteadyStateHypothesisScheduleProfile::EnergyConnectionSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1016

1017 Table 47 shows all association ends of EnergyConnectionSchedule with other classes.

1018

1019

**Table 47 – Association ends of  
SteadyStateHypothesisScheduleProfile::EnergyConnectionSchedule with other classes**

mult from	name	mult to	type	description
0..*	EnergyConnection	0..1	<a href="#">EnergyConnection</a>	(NC) Energy connection which has energy connection schedules.

1020

### 1021 3.43 (NC) EnergyConnectionTimePoint root class

1022 Energy connection values for a given point in time.

1023 Table 48 shows all attributes of EnergyConnectionTimePoint.

1024

1025

**Table 48 – Attributes of  
SteadyStateHypothesisScheduleProfile::EnergyConnectionTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
p	1..1	<a href="#">ActivePower</a>	(NC) Active power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution.

1026  
1027 Table 49 shows all association ends of EnergyConnectionTimePoint with other classes.

1028 **Table 49 – Association ends of**  
1029 **SteadyStateHypothesisScheduleProfile::EnergyConnectionTimePoint with other classes**

mult from	name	mult to	type	description
1..*	EnergyConnectionSchedule	1..1	<a href="#">EnergyConnectionSchedule</a>	(NC) The energy connection schedule that has this time point.

1030

### 1031 3.44 (NC) TapSchedule

1032 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1033 Schedule for tap.

1034 Table 50 shows all attributes of TapSchedule.

1035 **Table 50 – Attributes of SteadyStateHypothesisScheduleProfile::TapSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1036

1037 Table 51 shows all association ends of TapSchedule with other classes.

1038 **Table 51 – Association ends of SteadyStateHypothesisScheduleProfile::TapSchedule**  
1039 **with other classes**

mult from	name	mult to	type	description
0..*	TapChanger	0..1	<a href="#">TapChanger</a>	(NC) Tap changer which has tap schedules.

1040

### 1041 3.45 (NC) TapScheduleTimePoint root class

1042 Tap schedule values for a given point in time.

1043 Table 52 shows all attributes of TapScheduleTimePoint.

1044 **Table 52 – Attributes of SteadyStateHypothesisScheduleProfile::TapScheduleTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
controlEnabled	1..1	<a href="#">Boolean</a>	(NC) Specifies the regulation status of the equipment. True is regulating, false is not regulating.
step	1..1	<a href="#">Float</a>	(NC) Tap changer position. Starting step for a steady state solution. Non integer values are allowed to support continuous tap variables. The reasons for continuous value are to support study cases where no discrete tap changer has yet been designed, a solution where

name	mult	type	description
			a narrow voltage band forces the tap step to oscillate or to accommodate for a continuous solution as input. The attribute shall be equal to or greater than lowStep and equal to or less than highStep.

1045

1046

Table 53 shows all association ends of TapScheduleTimePoint with other classes.

1047

1048

**Table 53 – Association ends of SteadyStateHypothesisScheduleProfile::TapScheduleTimePoint with other classes**

mult from	name	mult to	type	description
1..*	TapSchedule	1..1	<a href="#">TapSchedule</a>	(NC) The tap schedule that has this time point.

1049

1050

### 3.46 (NC) RegulatingControlSchedule

1051

Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1052

Schedule for regulating control.

1053

Table 54 shows all attributes of RegulatingControlSchedule.

1054

1055

**Table 54 – Attributes of SteadyStateHypothesisScheduleProfile::RegulatingControlSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1056

1057

Table 55 shows all association ends of RegulatingControlSchedule with other classes.

1058

1059

**Table 55 – Association ends of SteadyStateHypothesisScheduleProfile::RegulatingControlSchedule with other classes**

mult from	name	mult to	type	description
0..*	RegulatingControl	0..1	<a href="#">RegulatingControl</a>	(NC) Regulating control which has regulating control schedules.

1060

1061

### 3.47 (NC) RegulatingControlTimePoint root class

1062

Regulating control values for a given point in time.

1063

Table 56 shows all attributes of RegulatingControlTimePoint.

1064

1065

**Table 56 – Attributes of SteadyStateHypothesisScheduleProfile::RegulatingControlTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.

name	mult	type	description
targetValue	1..1	<a href="#">Float</a>	(NC) The target value specified for case input. This value can be used for the target value without the use of schedules. The value has the units appropriate to the mode attribute.
targetValueUnitMultiplier	1..1	<a href="#">UnitMultiplier</a>	(NC) Specify the multiplier for used for the targetValue.
maxAllowedTargetValue	0..1	<a href="#">Float</a>	(NC) Maximum allowed target value (RegulatingControl.targetValue).
minAllowedTargetValue	0..1	<a href="#">Float</a>	(NC) Minimum allowed target value (RegulatingControl.targetValue).

1066

1067 Table 57 shows all association ends of RegulatingControlTimePoint with other classes.

1068

1069

**Table 57 – Association ends of  
SteadyStateHypothesisScheduleProfile::RegulatingControlTimePoint with other classes**

mult from	name	mult to	type	description
1..*	RegulatingControlSchedule	1..1	<a href="#">RegulatingControlSchedule</a>	(NC) The regulating control schedule that has this time point.
1..*	TapChangerControlSchedule	1..1	<a href="#">TapChangerControlSchedule</a>	(NC) The tap changer control schedule that has this time point.

1070

### 1071 3.48 (NC) TapChangerControlSchedule

1072 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1073 Schedule for tap changer control.

1074 Table 58 shows all attributes of TapChangerControlSchedule.

1075

1076

**Table 58 – Attributes of  
SteadyStateHypothesisScheduleProfile::TapChangerControlSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1077

1078 Table 59 shows all association ends of TapChangerControlSchedule with other classes.

1079

1080

1081

**Table 59 – Association ends of  
SteadyStateHypothesisScheduleProfile::TapChangerControlSchedule with other classes**

mult from	name	mult to	type	description
0..*	TapChangerControl	0..1	<a href="#">TapChangerControl</a>	(NC)

1082

1083 **3.49 (NC) VsConverterSchedule**1084 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1085 Schedule for VS converter.

1086 Table 60 shows all attributes of VsConverterSchedule.

1087 **Table 60 – Attributes of SteadyStateHypothesisScheduleProfile::VsConverterSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1088

1089 Table 61 shows all association ends of VsConverterSchedule with other classes.

1090

1091 **Table 61 – Association ends of SteadyStateHypothesisScheduleProfile::VsConverterSchedule with other classes**

mult from	name	mult to	type	description
0..*	VsConverter	0..1	<a href="#">VsConverter</a>	(NC) Vs converter which has Vs converter schedules.

1092

1093 **3.50 (abstract,NC) ACDCTimePoint root class**

1094 ACDC values for a given point in time.

1095 Table 62 shows all attributes of ACDCTimePoint.

1096 **Table 62 – Attributes of SteadyStateHypothesisScheduleProfile::ACDCTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
p	1..1	<a href="#">ActivePower</a>	(NC) Active power at the point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution in the case a simplified power flow model is used.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power at the point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution in the case a simplified power flow model is used.
targetPpcc	0..1	<a href="#">ActivePower</a>	(NC) Real power injection target in AC grid, at point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node.
targetUdc	0..1	<a href="#">Voltage</a>	(NC) Target value for DC voltage magnitude. The attribute shall be a positive value.

1097

1098 **3.51 (NC) VsConverterTimePoint**1099 Inheritance path = [ACDCTimePoint](#)

1100 VS converter values for a given point in time.  
1101 Table 63 shows all attributes of VsConverterTimePoint.

1102 **Table 63 – Attributes of SteadyStateHypothesisScheduleProfile::VsConverterTimePoint**

name	mult	type	description
droop	0..1	<a href="#">PU</a>	(NC) Droop constant. The pu value is obtained as $D [kV/MW] \times S_b / U_{bdc}$ . The attribute shall be a positive value.
droopCompensation	0..1	<a href="#">Resistance</a>	(NC) Compensation constant. Used to compensate for voltage drop when controlling voltage at a distant bus. The attribute shall be a positive value.
pPccControl	1..1	<a href="#">VsPpccControlKind</a>	(NC) Kind of control of real power and/or DC voltage.
qPccControl	1..1	<a href="#">VsQpccControlKind</a>	(NC) Kind of reactive power control.
qShare	0..1	<a href="#">PerCent</a>	(NC) Reactive power sharing factor among parallel converters on Uac control. The attribute shall be a positive value or zero.
targetQpcc	0..1	<a href="#">ReactivePower</a>	(NC) Reactive power injection target in AC grid, at point of common coupling. Load sign convention is used, i.e. positive sign means flow out from a node.
targetUpcc	0..1	<a href="#">Voltage</a>	(NC) Voltage target in AC grid, at point of common coupling. The attribute shall be a positive value.
targetPowerFactorPcc	0..1	<a href="#">Float</a>	(NC) Power factor target at the AC side, at point of common coupling. The attribute shall be a positive value.
targetPhasePcc	0..1	<a href="#">AngleDegrees</a>	(NC) Phase target at AC side, at point of common coupling. The attribute shall be a positive value.
targetPWMfactor	0..1	<a href="#">Float</a>	(NC) Magnitude of pulse-modulation factor. The attribute shall be a positive value.
atTime	1..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
p	1..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
q	1..1	<a href="#">ReactivePower</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
targetPpcc	0..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
targetUdc	0..1	<a href="#">Voltage</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>

1103  
1104 Table 64 shows all association ends of VsConverterTimePoint with other classes.

1105 **Table 64 – Association ends of**  
1106 **SteadyStateHypothesisScheduleProfile::VsConverterTimePoint with other classes**

mult from	name	mult to	type	description
1..*	VsConverterSchedule	1..1	<a href="#">VsConverterSchedule</a>	(NC) The VS converter schedule that has this time point.

1107  
1108 **3.52 (NC) CsConverterTimePoint**  
1109 Inheritance path = [ACDCTimePoint](#)  
1110 CSConverter values for a given point in time.  
1111 Table 65 shows all attributes of CsConverterTimePoint.

1112 **Table 65 – Attributes of SteadyStateHypothesisScheduleProfile::CsConverterTimePoint**

name	mult	type	description
operatingMode	1..1	<a href="#">CsOperatingModeKind</a>	(NC) Indicates whether the DC pole is operating as an inverter or as a rectifier. It is converter's control variable used in power flow.
pPccControl	1..1	<a href="#">CsPpccControlKind</a>	(NC) Kind of active power control.
targetAlpha	0..1	<a href="#">AngleDegrees</a>	(NC) Target firing angle. It is converter's control variable used in power flow. It is only applicable for rectifier if continuous tap changer control is used. Allowed values are within the range $\text{minAlpha} \leq \text{targetAlpha} \leq \text{maxAlpha}$ . The attribute shall be a positive value.
targetGamma	0..1	<a href="#">AngleDegrees</a>	(NC) Target extinction angle. It is converter's control variable used in power flow. It is only applicable for inverter if continuous tap changer control is used. Allowed values are within the range $\text{minGamma} \leq \text{targetGamma} \leq \text{maxGamma}$ . The attribute shall be a positive value.
targetIdc	0..1	<a href="#">CurrentFlow</a>	(NC) DC current target value. It is converter's control variable used in power flow. The attribute shall be a positive value.
atTime	1..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
p	1..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
q	1..1	<a href="#">ReactivePower</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
targetPpcc	0..1	<a href="#">ActivePower</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>
targetUdc	0..1	<a href="#">Voltage</a>	(NC) inherited from: <a href="#">ACDCTimePoint</a>

1113

1114 Table 66 shows all association ends of CsConverterTimePoint with other classes.

1115

1116

**Table 66 – Association ends of  
SteadyStateHypothesisScheduleProfile::CsConverterTimePoint with other classes**

mult from	name	mult to	type	description
1..*	CsConverterSchedule	1..1	<a href="#">CsConverterSchedule</a>	(NC) The CS converter schedule that has this time point.

1117

### 1118 3.53 (NC) CsConverterSchedule

1119 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1120 Schedule for CS converter.

1121 Table 67 shows all attributes of CsConverterSchedule.

1122 **Table 67 – Attributes of SteadyStateHypothesisScheduleProfile::CsConverterSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>



1123  
1124 Table 68 shows all association ends of CsConverterSchedule with other classes.

1125 **Table 68 – Association ends of**  
1126 **SteadyStateHypothesisScheduleProfile::CsConverterSchedule with other classes**

mult from	name	mult to	type	description
0..*	CsConverter	0..1	<a href="#">CsConverter</a>	(NC) Cs converter which has Cs converter schedules.

1127  
1128 **3.54 (NC) SwitchSchedule**

1129 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1130 Schedule for switch.

1131 Table 69 shows all attributes of SwitchSchedule.

1132 **Table 69 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1133  
1134 Table 70 shows all association ends of SwitchSchedule with other classes.

1135 **Table 70 – Association ends of SteadyStateHypothesisScheduleProfile::SwitchSchedule**  
1136 **with other classes**

mult from	name	mult to	type	description
0..*	Switch	0..1	<a href="#">Switch</a>	(NC) Switch which has switch schedules.

1137  
1138 **3.55 (NC) SwitchTimePoint root class**

1139 Switch values for a given point in time.

1140 Table 71 shows all attributes of SwitchTimePoint.

1141 **Table 71 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
open	1..1	<a href="#">Boolean</a>	(NC) The attribute tells if the switch is considered open when used as input to topology processing.
locked	1..1	<a href="#">Boolean</a>	(NC) If true, the switch is locked. The resulting switch state is a combination of locked and Switch.open attributes as follows: - locked=true and Switch.open=true. The resulting state is open and locked;

name	mult	type	description
			- locked=false and Switch.open=true. The resulting state is open; - locked=false and Switch.open=false. The resulting state is closed.

1142

1143

Table 72 shows all association ends of SwitchTimePoint with other classes.

1144

1145

**Table 72 – Association ends of SteadyStateHypothesisScheduleProfile::SwitchTimePoint with other classes**

mult from	name	mult to	type	description
1..*	SwitchSchedule	1..1	<a href="#">SwitchSchedule</a>	The switch schedule that has this time point.

1146

1147

### 3.56 (NC) InServiceSchedule

1148

Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1149

Schedule for elements having in service.

1150

Table 73 shows all attributes of InServiceSchedule.

1151

**Table 73 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1152

1153

Table 74 shows all association ends of InServiceSchedule with other classes.

1154

1155

**Table 74 – Association ends of SteadyStateHypothesisScheduleProfile::InServiceSchedule with other classes**

mult from	name	mult to	type	description
0..*	Equipment	0..1	<a href="#">Equipment</a>	(NC) Equipment which has equipment schedules.

1156

1157

### 3.57 (NC) InServiceTimePoint root class

1158

In service values for a given point in time.

1159

Table 75 shows all attributes of InServiceTimePoint.

1160

**Table 75 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
inService	1..1	<a href="#">Boolean</a>	(NC) Specifies the availability of the equipment. True means the equipment is available for topology processing, which determines if the

name	mult	type	description
			equipment is energized or not. False means that the equipment is treated by network applications as if it is not in the model.

1161

1162 Table 76 shows all association ends of InServiceTimePoint with other classes.

1163 **Table 76 – Association ends of**  
1164 **SteadyStateHypothesisScheduleProfile::InServiceTimePoint with other classes**

mult from	name	mult to	type	description
1..*	InServiceSchedule	1..1	<a href="#">InServiceSchedule</a>	(NC) The in service schedule that has this time point.

1165

1166 **3.58 (NC) ControlAreaSchedule**1167 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1168 Schedule for control area.

1169 Table 77 shows all attributes of ControlAreaSchedule.

1170 **Table 77 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1171

1172 Table 78 shows all association ends of ControlAreaSchedule with other classes.

1173 **Table 78 – Association ends of**  
1174 **SteadyStateHypothesisScheduleProfile::ControlAreaSchedule with other classes**

mult from	name	mult to	type	description
0..*	ControlArea	0..1	<a href="#">ControlArea</a>	(NC) Control area which has control area schedules.

1175

1176 **3.59 (NC) ControlAreaTimePoint root class**

1177 Participation factor for a given point in time.

1178 Table 79 shows all attributes of ControlAreaTimePoint.

1179 **Table 79 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
netInterchange	1..1	<a href="#">ActivePower</a>	(NC) The specified positive net interchange into the control area, i.e. positive sign means flow into the area.

name	mult	type	description
pTolerance	0..1	<a href="#">ActivePower</a>	(NC) Active power net interchange tolerance. The attribute shall be a positive value or zero.

1180

1181 Table 80 shows all association ends of ControlAreaTimePoint with other classes.

1182

1183

**Table 80 – Association ends of  
SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint with other classes**

mult from	name	mult to	type	description
1..*	ControlAreaSchedule	1..1	<a href="#">ControlAreaSchedule</a>	(NC) The control area schedule that has this time point.

1184

### 1185 3.60 (NC) SynchronousMachineSchedule

1186 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1187 Schedule for synchronous machine.

1188 Table 81 shows all attributes of SynchronousMachineSchedule.

1189

1190

**Table 81 – Attributes of  
SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1191

1192 Table 82 shows all association ends of SynchronousMachineSchedule with other classes.

1193

1194

1195

**Table 82 – Association ends of  
SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	SynchronousMachine	0..1	<a href="#">SynchronousMachine</a>	(NC) Synchronous machine which has synchronous machine schedules.

1196

### 1197 3.61 (NC) SynchronousMachineTimePoint root class

1198 Synchronous machine values for a given point in time.

1199 Table 83 shows all attributes of SynchronousMachineTimePoint.

1200

1201

**Table 83 – Attributes of  
SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.

name	mult	type	description
operatingMode	1..1	<a href="#">SynchronousMachineOperatingMode</a>	(NC) Current mode of operation.
referencePriority	1..1	<a href="#">Integer</a>	(NC) Priority of unit for use as powerflow voltage phase angle reference bus selection. 0 = don't care (default) 1 = highest priority. 2 is less than 1 and so on.

1202

1203

Table 84 shows all association ends of SynchronousMachineTimePoint with other classes.

1204

1205

1206

**Table 84 – Association ends of SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint with other classes**

mult from	name	mult to	type	description
1..*	SynchronousMachineSchedule	1..1	<a href="#">SynchronousMachineSchedule</a>	(NC) The synchronous machine schedule that has this time point.

1207

1208

### 3.62 (NC) AsynchronousMachineSchedule

1209

Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1210

Schedule for asynchronous machine.

1211

Table 85 shows all attributes of AsynchronousMachineSchedule.

1212

1213

**Table 85 – Attributes of SteadyStateHypothesisScheduleProfile::AsynchronousMachineSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1214

1215

Table 86 shows all association ends of AsynchronousMachineSchedule with other classes.

1216

1217

1218

**Table 86 – Association ends of SteadyStateHypothesisScheduleProfile::AsynchronousMachineSchedule with other classes**

mult from	name	mult to	type	description
0..*	AsynchronousMachine	0..1	<a href="#">AsynchronousMachine</a>	(NC) Asynchronous machine which has asynchronous machine schedules.

1219

1220

### 3.63 (NC) AsynchronousMachineTimePoint root class

1221

Asynchronous machine values for a given point in time.

1222

Table 87 shows all attributes of AsynchronousMachineTimePoint.

1223  
1224

**Table 87 – Attributes of  
SteadyStateHypothesisScheduleProfile::AsynchronousMachineTimePoint**

name	mult	type	description
asynchronousMachineType	1..1	<a href="#">AsynchronousMachineKind</a>	(NC) Indicates the type of Asynchronous Machine (motor or generator).
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.

1225  
1226

Table 88 shows all association ends of AsynchronousMachineTimePoint with other classes.

1227  
1228  
1229

**Table 88 – Association ends of  
SteadyStateHypothesisScheduleProfile::AsynchronousMachineTimePoint with other  
classes**

mult from	name	mult to	type	description
1..*	AsynchronousMachineSchedule	1..1	<a href="#">AsynchronousMachineSchedule</a>	(NC) The asynchronous machine schedule that has this time point.

1230

### 1231 3.64 (NC) ExternalNetworkInjectionSchedule

1232 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1233 Schedule for external network injection.

1234 Table 89 shows all attributes of ExternalNetworkInjectionSchedule.

1235  
1236

**Table 89 – Attributes of  
SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1237  
1238

Table 90 shows all association ends of ExternalNetworkInjectionSchedule with other classes.

1239  
1240  
1241

**Table 90 – Association ends of  
SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	ExternalNetworkInjection	0..1	<a href="#">ExternalNetworkInjection</a>	(NC) External Network Injection which has External Network Injection schedules.

1242

### 1243 3.65 (NC) ExternalNetworkInjectionTimePoint root class

1244 External network injection values for a given point in time.

1245 Table 91 shows all attributes of ExternalNetworkInjectionTimePoint.

1246  
1247**Table 91 – Attributes of  
SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
referencePriority	1..1	<a href="#">Integer</a>	(NC) Priority of unit for use as powerflow voltage phase angle reference bus selection. 0 = don't care (default) 1 = highest priority. 2 is less than 1 and so on.
p	1..1	<a href="#">ActivePower</a>	(NC) Active power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for steady state solutions.

1248  
1249  
1250  
1251  
1252

Table 92 shows all association ends of ExternalNetworkInjectionTimePoint with other classes.

**Table 92 – Association ends of  
SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint with other  
classes**

mult from	name	mult to	type	description
1..*	ExternalNetworkInjectionSchedule	1..1	<a href="#">ExternalNetworkInjectionSchedule</a>	(NC) The external network injection schedule that has this time point.

1253  
1254  
1255  
1256  
1257  
1258**3.66 (NC) BatteryUnitSchedule**

Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

Schedule for battery unit.

Table 93 shows all attributes of BatteryUnitSchedule.

**Table 93 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1259  
1260  
1261  
1262

Table 94 shows all association ends of BatteryUnitSchedule with other classes.

**Table 94 – Association ends of  
SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule with other classes**

mult from	name	mult to	type	description
0..*	BatteryUnit	0..1	<a href="#">BatteryUnit</a>	(NC) Battery unit which has battery unit schedules.



1263

1264 **3.67 (NC) BatteryUnitTimePoint root class**

1265 Battery unit values for a given point in time.

1266 Table 95 shows all attributes of BatteryUnitTimePoint.

1267 **Table 95 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
batteryState	1..1	<a href="#">BatteryStateKind</a>	(NC) The current state of the battery (charging, full, etc.).
storedE	1..1	<a href="#">RealEnergy</a>	(NC) Amount of energy currently stored. The attribute shall be a positive value or zero and lower than BatteryUnit.ratedE.

1268

1269 Table 96 shows all association ends of BatteryUnitTimePoint with other classes.

1270

1271 **Table 96 – Association ends of SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	BatteryUnitSchedule	1..1	<a href="#">BatteryUnitSchedule</a>	(NC) The battery unit schedule that has this time point.

1272

1273 **3.68 (abstract) BatteryUnit root class**

1274 An electrochemical energy storage device.

1275 **3.69 (NC) VoltageLimitSchedule**1276 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1277 Schedule for voltage limit.

1278 Table 97 shows all attributes of VoltageLimitSchedule.

1279 **Table 97 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1280

1281 Table 98 shows all association ends of VoltageLimitSchedule with other classes.

1282

1283 **Table 98 – Association ends of SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	VoltageLimit	0..1	<a href="#">VoltageLimit</a>	(NC) Voltage limit which has voltage limit schedules.

1284

1285 **3.70 (abstract) VoltageLimit root class**

1286 Operational limit applied to voltage.

1287 The use of operational VoltageLimit is preferred instead of limits defined at VoltageLevel. The  
1288 operational VoltageLimits are used, if present.1289 **3.71 (NC) VoltageLimitTimePoint root class**

1290 Voltage limit values for a given point in time.

1291 Table 99 shows all attributes of VoltageLimitTimePoint.

1292 **Table 99 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
value	1..1	<a href="#">Voltage</a>	(NC) Limit on voltage. High or low limit nature of the limit depends upon the properties of the operational limit type. The attribute shall be a positive value or zero.

1293

1294 Table 100 shows all association ends of VoltageLimitTimePoint with other classes.

1295 **Table 100 – Association ends of**  
1296 **SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	VoltageLimitSchedule	1..1	<a href="#">VoltageLimitSchedule</a>	(NC) The voltage limit schedule that has this time point.

1297

1298 **3.72 (NC) ActivePowerLimitSchedule**1299 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1300 Schedule for active power limit.

1301 Table 101 shows all attributes of ActivePowerLimitSchedule.

1302 **Table 101 – Attributes of**  
1303 **SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1304

1305 Table 102 shows all association ends of ActivePowerLimitSchedule with other classes.

1306 **Table 102 – Association ends of**  
1307 **SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	ActivePowerLimit	0..1	<a href="#">ActivePowerLimit</a>	(NC) Active power limit which has active power limit schedules.

1308

1309 **3.73 (abstract) ActivePowerLimit root class**

1310 Limit on active power flow.

1311 **3.74 (NC) ActivePowerLimitTimePoint root class**

1312 Active power limit for a given point in time.

1313 Table 103 shows all attributes of ActivePowerLimitTimePoint.

1314 **Table 103 – Attributes of**  
1315 **SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
value	1..1	<a href="#">ActivePower</a>	(NC) Value of active power limit. The attribute shall be a positive value or zero.

1316

1317 Table 104 shows all association ends of ActivePowerLimitTimePoint with other classes.

1318 **Table 104 – Association ends of**  
1319 **SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	ActivePowerLimitSchedule	1..1	<a href="#">ActivePowerLimitSchedule</a>	(NC) The active power limit schedule that has this time point.

1320

1321 **3.75 (NC) ApparentPowerLimitSchedule**1322 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1323 Schedule for apparent power limit.

1324 Table 105 shows all attributes of ApparentPowerLimitSchedule.

1325 **Table 105 – Attributes of**  
1326 **SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1327

1328 Table 106 shows all association ends of ApparentPowerLimitSchedule with other classes.

1329  
1330  
1331

**Table 106 – Association ends of  
SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	ApparentPowerLimit	0..1	<a href="#">ApparentPowerLimit</a>	(NC) Apparent power limit which has apparent power limit schedules.

1332

### 1333 3.76 (abstract) ApparentPowerLimit root class

1334 Apparent power limit.

### 1335 3.77 (NC) ApparentPowerLimitTimePoint root class

1336 Apparent power limit for a given point in time.

1337 Table 107 shows all attributes of ApparentPowerLimitTimePoint.

1338

1339

**Table 107 – Attributes of  
SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
value	1..1	<a href="#">ApparentPower</a>	(NC) The apparent power limit. The attribute shall be a positive value or zero.

1340

1341 Table 108 shows all association ends of ApparentPowerLimitTimePoint with other classes.

1342

1343

1344

**Table 108 – Association ends of  
SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint with other  
classes**

mult from	name	mult to	type	description
1..*	ApparentPowerLimitSchedule	1..1	<a href="#">ApparentPowerLimitSchedule</a>	(NC) The apparent power limit schedule that has this time point.

1345

### 1346 3.78 (NC) CurrentLimitSchedule

1347 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1348 Schedule for current limit.

1349 Table 109 shows all attributes of CurrentLimitSchedule.

1350

**Table 109 – Attributes of SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1351

1352 Table 110 shows all association ends of CurrentLimitSchedule with other classes.

1353  
1354**Table 110 – Association ends of  
SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	CurrentLimit	0..1	<a href="#">CurrentLimit</a>	(NC) Current limit which has current limit schedules.

1355

**3.79 (abstract) CurrentLimit root class**

1357 Operational limit on current.

**3.80 (NC) CurrentLimitTimePoint root class**

1359 Current limit values for a given point in time.

1360 Table 111 shows all attributes of CurrentLimitTimePoint.

1361

1362

**Table 111 – Attributes of  
SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
value	1..1	<a href="#">CurrentFlow</a>	(NC) Limit on current flow. The attribute shall be a positive value or zero.

1363

1364

Table 112 shows all association ends of CurrentLimitTimePoint with other classes.

1365

1366

**Table 112 – Association ends of  
SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	CurrentLimitSchedule	1..1	<a href="#">CurrentLimitSchedule</a>	(NC) The current limit schedule that has this time point.

1367

**3.81 (NC) ShuntCompensatorSchedule**1369 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1370 Schedule for shunt compensator.

1371 Table 113 shows all attributes of ShuntCompensatorSchedule.

1372

1373

**Table 113 – Attributes of  
SteadyStateHypothesisScheduleProfile::ShuntCompensatorSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1374

1375

Table 114 shows all association ends of ShuntCompensatorSchedule with other classes.

1376 **Table 114 – Association ends of**  
1377 **SteadyStateHypothesisScheduleProfile::ShuntCompensatorSchedule with other classes**

mult from	name	mult to	type	description
0..*	ShuntCompensator	0..1	<a href="#">ShuntCompensator</a>	(NC) Shunt compensator which has shunt compensator schedules.

1378

### 1379 3.82 (NC) ShuntCompensatorTimePoint root class

1380 Shunt compensator values for a given point in time.

1381 Table 115 shows all attributes of ShuntCompensatorTimePoint.

1382 **Table 115 – Attributes of**  
1383 **SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
sections	1..1	<a href="#">Float</a>	(NC) Shunt compensator sections in use. Starting value for steady state solution. The attribute shall be a positive value or zero. Non integer values are allowed to support continuous variables. The reasons for continuous value are to support study cases where no discrete shunt compensators has yet been designed, a solutions where a narrow voltage band force the sections to oscillate or accommodate for a continuous solution as input.  For LinearShuntCompensator the value shall be between zero and ShuntCompensator.maximumSections. At value zero the shunt compensator conductance and admittance is zero. Linear interpolation of conductance and admittance between the previous and next integer section is applied in case of non-integer values.  For NonlinearShuntCompensator-s shall only be set to one of the NonlinearShuntCompensatorPoint.sectionNumber. There is no interpolation between NonlinearShuntCompensatorPoint-s.

1384

1385 Table 116 shows all association ends of ShuntCompensatorTimePoint with other classes.

1386 **Table 116 – Association ends of**  
1387 **SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint with other**  
1388 **classes**

mult from	name	mult to	type	description
1..*	ShuntCompensatorSchedule	1..1	<a href="#">ShuntCompensatorSchedule</a>	(NC) The shunt compensator schedule that has this time point.

1389

### 1390 3.83 (abstract) ShuntCompensator root class

1391 A shunt capacitor or reactor or switchable bank of shunt capacitors or reactors. A section of a  
1392 shunt compensator is an individual capacitor or reactor. A negative value for bPerSection  
1393 indicates that the compensator is a reactor. ShuntCompensator is a single terminal device.  
1394 Ground is implied.

1395 **3.84 (NC) StaticVarCompensatorSchedule**1396 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1397 Schedule for static var compensator.

1398 Table 117 shows all attributes of StaticVarCompensatorSchedule.

1399

1400

**Table 117 – Attributes of  
SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1401

1402 Table 118 shows all association ends of StaticVarCompensatorSchedule with other classes.

1403

1404

1405

**Table 118 – Association ends of  
SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule with other  
classes**

mult from	name	mult to	type	description
0..*	StaticVarCompensator	0..1	<a href="#">StaticVarCompensator</a>	(NC) Static var compensator which has static var compensator schedules.

1406

1407 **3.85 (NC) StaticVarCompensatorTimePoint root class**

1408 Static var compensator values for a given point in time.

1409 Table 119 shows all attributes of StaticVarCompensatorTimePoint.

1410

1411

**Table 119 – Attributes of  
SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint**

name	mult	type	description
atTime	0..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
q	1..1	<a href="#">ReactivePower</a>	(NC) Reactive power injection. Load sign convention is used, i.e. positive sign means flow out from a node. Starting value for a steady state solution.

1412

1413 Table 120 shows all association ends of StaticVarCompensatorTimePoint with other classes.

1414

1415

1416

**Table 120 – Association ends of  
SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint with other  
classes**

mult from	name	mult to	type	description
1..*	StaticVarCompensatorSchedule	1..1	<a href="#">StaticVarCompensatorSchedule</a>	(NC) The StaticVarCompensator schedule that has this time point.

1417



**1418 3.86 (abstract) StaticVarCompensator root class**

1419 A facility for providing variable and controllable shunt reactive power. The SVC typically  
1420 consists of a stepdown transformer, filter, thyristor-controlled reactor, and thyristor-switched  
1421 capacitor arms.

1422 The SVC may operate in fixed MVar output mode or in voltage control mode. When in voltage  
1423 control mode, the output of the SVC will be proportional to the deviation of voltage at the  
1424 controlled bus from the voltage setpoint. The SVC characteristic slope defines the proportion.  
1425 If the voltage at the controlled bus is equal to the voltage setpoint, the SVC MVar output is zero.

**1426 3.87 (NC) GeneratingUnitSchedule**

1427 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1428 Schedule for generating unit.

1429 Table 121 shows all attributes of GeneratingUnitSchedule.

1430

1431

**Table 121 – Attributes of  
SteadyStateHypothesisScheduleProfile::GeneratingUnitSchedule**

name	mult	type	description
interpolationKind	1..1	<a href="#">TimeSeriesInterpolationKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
timeSeriesKind	0..1	<a href="#">BaseTimeSeriesKind</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
generatedAtTime	0..1	<a href="#">DateTime</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
percentile	0..1	<a href="#">Integer</a>	(NC) inherited from: <a href="#">BaseTimeSeries</a>
description	0..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
mRID	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>
name	1..1	<a href="#">String</a>	inherited from: <a href="#">IdentifiedObject</a>

1432

1433

Table 122 shows all association ends of GeneratingUnitSchedule with other classes.

1434

1435

**Table 122 – Association ends of  
SteadyStateHypothesisScheduleProfile::GeneratingUnitSchedule with other classes**

mult from	name	mult to	type	description
0..*	GeneratingUnit	0..1	<a href="#">GeneratingUnit</a>	(NC) Generating unit which has generating unit schedules.

1436

**1437 3.88 (NC) GeneratingUnitTimePoint root class**

1438 Generating unit values for a given point in time.

1439 Table 123 shows all attributes of GeneratingUnitTimePoint.

1440

1441

**Table 123 – Attributes of  
SteadyStateHypothesisScheduleProfile::GeneratingUnitTimePoint**

name	mult	type	description
atTime	1..1	<a href="#">DateTime</a>	(NC) The time the data is valid for.
normalPF	1..1	<a href="#">Float</a>	(NC) Generating unit economic participation factor. The sum of the participation factors across generating units does not have to sum to one. It is used for representing distributed slack participation factor. The attribute shall be a positive value or zero.

1442

1443

Table 124 shows all association ends of GeneratingUnitTimePoint with other classes.

1444  
1445

**Table 124 – Association ends of SteadyStateHypothesisScheduleProfile::GeneratingUnitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	GeneratingUnitSchedule	1..1	<a href="#">GeneratingUnitSchedule</a>	The generating unit schedule that has this time point.

1446

1447 **3.89 (abstract) GeneratingUnit root class**

1448 A single or set of synchronous machines for converting mechanical power into alternating-  
1449 current power. For example, individual machines within a set may be defined for scheduling  
1450 purposes while a single control signal is derived for the set. In this case there would be a  
1451 GeneratingUnit for each member of the set and an additional GeneratingUnit corresponding to  
1452 the set.

1453 **3.90 MonthDay primitive**

1454 MonthDay format as "--mm-dd", which conforms with XSD data type gMonthDay.

1455 **3.91 ActivePower datatype**

1456 Product of RMS value of the voltage and the RMS value of the in-phase component of the  
1457 current.

1458 Table 125 shows all attributes of ActivePower.

1459

**Table 125 – Attributes of SteadyStateHypothesisScheduleProfile::ActivePower**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=M)
unit	0..1	<a href="#">UnitSymbol</a>	(const=W)

1460

1461 **3.92 Float primitive**

1462 A floating point number. The range is unspecified and not limited.

1463 **3.93 UnitMultiplier enumeration**

1464 The unit multipliers defined for the CIM. When applied to unit symbols, the unit symbol is  
1465 treated as a derived unit. Regardless of the contents of the unit symbol text, the unit symbol  
1466 shall be treated as if it were a single-character unit symbol. Unit symbols should not contain  
1467 multipliers, and it should be left to the multiplier to define the multiple for an entire data type.

1468 For example, if a unit symbol is "m2Pers" and the multiplier is "k", then the value is  $k(m^{**2}/s)$ ,  
1469 and the multiplier applies to the entire final value, not to any individual part of the value. This  
1470 can be conceptualized by substituting a derived unit symbol for the unit type. If one imagines  
1471 that the symbol "P" represents the derived unit "m2Pers", then applying the multiplier "k" can  
1472 be conceptualized simply as "kP".

1473 For example, the SI unit for mass is "kg" and not "g". If the unit symbol is defined as "kg", then  
1474 the multiplier is applied to "kg" as a whole and does not replace the "k" in front of the "g". In  
1475 this case, the multiplier of "m" would be used with the unit symbol of "kg" to represent one gram.  
1476 As a text string, this violates the instructions in IEC 80000-1. However, because the unit symbol  
1477 in CIM is treated as a derived unit instead of as an SI unit, it makes more sense to conceptualize  
1478 the "kg" as if it were replaced by one of the proposed replacements for the SI mass symbol. If  
1479 one imagines that the "kg" were replaced by a symbol "P", then it is easier to conceptualize the  
1480 multiplier "m" as creating the proper unit "mP", and not the forbidden unit "mkg".

1481 Table 126 shows all literals of UnitMultiplier.

1482

**Table 126 – Literals of SteadyStateHypothesisScheduleProfile::UnitMultiplier**

literal	value	description
none	0	No multiplier or equivalently multiply by 1.
k	3	Kilo 10**3.
M	6	Mega 10**6.

1483

**1484 3.94 UnitSymbol enumeration**

1485 The derived units defined for usage in the CIM. In some cases, the derived unit is equal to an  
1486 SI unit. Whenever possible, the standard derived symbol is used instead of the formula for the  
1487 derived unit. For example, the unit symbol Farad is defined as "F" instead of "CPerV". In cases  
1488 where a standard symbol does not exist for a derived unit, the formula for the unit is used as  
1489 the unit symbol. For example, density does not have a standard symbol and so it is represented  
1490 as "kgPerm3". With the exception of the "kg", which is an SI unit, the unit symbols do not contain  
1491 multipliers and therefore represent the base derived unit to which a multiplier can be applied as  
1492 a whole.

1493 Every unit symbol is treated as an unparseable text as if it were a single-letter symbol. The  
1494 meaning of each unit symbol is defined by the accompanying descriptive text and not by the  
1495 text contents of the unit symbol.

1496 To allow the widest possible range of serializations without requiring special character handling,  
1497 several substitutions are made which deviate from the format described in IEC 80000-1. The  
1498 division symbol "/" is replaced by the letters "Per". Exponents are written in plain text after the  
1499 unit as "m3" instead of being formatted as "m" with a superscript of 3 or introducing a symbol  
1500 as in "m^3". The degree symbol "°" is replaced with the letters "deg". Any clarification of the  
1501 meaning for a substitution is included in the description for the unit symbol.

1502 Non-SI units are included in list of unit symbols to allow sources of data to be correctly labelled  
1503 with their non-SI units (for example, a GPS sensor that is reporting numbers that represent feet  
1504 instead of meters). This allows software to use the unit symbol information correctly convert  
1505 and scale the raw data of those sources into SI-based units.

1506 The integer values are used for harmonization with IEC 61850.

1507 Table 127 shows all literals of UnitSymbol.

1508

**Table 127 – Literals of SteadyStateHypothesisScheduleProfile::UnitSymbol**

literal	value	description
none	0	Dimension less quantity, e.g. count, per unit, etc.
A	5	Current in amperes.
deg	9	Plane angle in degrees.
V	29	Electric potential in volts (W/A).
ohm	30	Electric resistance in ohms (V/A).
W	38	Real power in watts (J/s). Electrical power may have real and reactive components. The real portion of electrical power ( $I^2R$ or $VI\cos(\phi)$ ), is expressed in Watts. See also apparent power and reactive power.
VA	61	Apparent power in volt amperes. See also real power and reactive power.
VAr	63	Reactive power in volt amperes reactive. The "reactive" or "imaginary" component of electrical power ( $VI\sin(\phi)$ ). (See also real power and apparent power).  Note: Different meter designs use different methods to arrive at their results. Some meters may compute reactive power as an arithmetic value, while others compute the value

literal	value	description
		vectorially. The data consumer should determine the method in use and the suitability of the measurement for the intended purpose.
Wh	72	Real energy in watt hours.

1509

1510 **3.95 ReactivePower datatype**1511 Product of RMS value of the voltage and the RMS value of the quadrature component of the  
1512 current.

1513 Table 128 shows all attributes of ReactivePower.

1514 **Table 128 – Attributes of SteadyStateHypothesisScheduleProfile::ReactivePower**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
unit	0..1	<a href="#">UnitSymbol</a>	(const=VAr)
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=M)

1515

1516 **3.96 Voltage datatype**

1517 Electrical voltage, can be both AC and DC.

1518 Table 129 shows all attributes of Voltage.

1519 **Table 129 – Attributes of SteadyStateHypothesisScheduleProfile::Voltage**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=k)
unit	0..1	<a href="#">UnitSymbol</a>	(const=V)

1520

1521 **3.97 DateTime primitive**1522 Date and time as "yyyy-mm-ddThh:mm:ss.sss", which conforms with ISO 8601. UTC time zone  
1523 is specified as "yyyy-mm-ddThh:mm:ss.sssZ". A local timezone relative UTC is specified as  
1524 "yyyy-mm-ddThh:mm:ss.sss-hh:mm". The second component (shown here as "ss.sss") could  
1525 have any number of digits in its fractional part to allow any kind of precision beyond seconds.1526 **3.98 ApparentPower datatype**

1527 Product of the RMS value of the voltage and the RMS value of the current.

1528 Table 130 shows all attributes of ApparentPower.

1529 **Table 130 – Attributes of SteadyStateHypothesisScheduleProfile::ApparentPower**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=M)
unit	0..1	<a href="#">UnitSymbol</a>	(const=VA)

1530

1531 **3.99 AsynchronousMachineKind enumeration**

1532 Kind of Asynchronous Machine.

1533 Table 131 shows all literals of AsynchronousMachineKind.

1534  
1535**Table 131 – Literals of  
SteadyStateHypothesisScheduleProfile::AsynchronousMachineKind**

literal	value	description
generator		The Asynchronous Machine is a generator.
motor		The Asynchronous Machine is a motor.

1536

**3.100 (NC) DayOfWeekKind enumeration**

1538 The kind of day to be included in a regular schedule.

1539 Table 132 shows all literals of DayOfWeekKind.

**Table 132 – Literals of SteadyStateHypothesisScheduleProfile::DayOfWeekKind**

literal	value	description
monday		Monday as the day of the week.
tuesday		Tuesday as the day of the week.
wednesday		Wednesday as the day of the week.
thursday		Thursday as the day of the week.
friday		Friday as the day of the week.
saturday		Saturday as the day of the week.
sunday		Sunday as the day of the week.
weekday		A day of the week other than Sunday or Saturday.
weekend		A day of the week which is Sunday or Saturday.
all		All days of the week.
holiday		
bridgeDay		A day that is a gap between two distinguished days e.g holiday and weekend that leads to an abnormal scheduling behavior. e.g. if Ascension day falls on a Thursday, then Friday would be a bridge day due to the schedule will not have a normal Friday consumption and production.

1541

**3.101 Integer primitive**

1543 An integer number. The range is unspecified and not limited.

**3.102 BatteryStateKind enumeration**

1545 The state of the battery unit.

1546 Table 133 shows all literals of BatteryStateKind.

**Table 133 – Literals of SteadyStateHypothesisScheduleProfile::BatteryStateKind**

literal	value	description
discharging		Stored energy is decreasing.
full		Unable to charge, and not discharging.
waiting		Neither charging nor discharging, but able to do so.
charging		Stored energy is increasing.
empty		Unable to discharge, and not charging.

1548

1549 **3.103 RealEnergy datatype**

1550 Real electrical energy.

1551 Table 134 shows all attributes of RealEnergy.

1552 **Table 134 – Attributes of SteadyStateHypothesisScheduleProfile::RealEnergy**

name	mult	type	description
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=M)
unit	0..1	<a href="#">UnitSymbol</a>	(const=Wh)
value	0..1	<a href="#">Float</a>	

1553

1554 **3.104 CsOperatingModeKind enumeration**

1555 Operating mode for HVDC line operating as Current Source Converter.

1556 Table 135 shows all literals of CsOperatingModeKind.

1557 **Table 135 – Literals of SteadyStateHypothesisScheduleProfile::CsOperatingModeKind**

literal	value	description
inverter		Operating as inverter, which is the power receiving end.
rectifier		Operating as rectifier, which is the power sending end.

1558

1559 **3.105 CsPpccControlKind enumeration**

1560 Active power control modes for HVDC line operating as Current Source Converter.

1561 Table 136 shows all literals of CsPpccControlKind.

1562 **Table 136 – Literals of SteadyStateHypothesisScheduleProfile::CsPpccControlKind**

literal	value	description
activePower		Control is active power control at AC side, at point of common coupling. Target is provided by ACDCConverter.targetPpcc.
dcVoltage		Control is DC voltage with target value provided by ACDCConverter.targetUdc.
dcCurrent		Control is DC current with target value provided by CsConverter.targetIdc.

1563

1564 **3.106 AngleDegrees datatype**

1565 Measurement of angle in degrees.

1566 Table 137 shows all attributes of AngleDegrees.

1567 **Table 137 – Attributes of SteadyStateHypothesisScheduleProfile::AngleDegrees**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
unit	0..1	<a href="#">UnitSymbol</a>	(const=deg)
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=none)

1568

1569 **3.107 CurrentFlow datatype**1570 Electrical current with sign convention: positive flow is out of the conducting equipment into the connectivity node. Can be both AC and DC.  
1571

1572 Table 138 shows all attributes of CurrentFlow.

1573 **Table 138 – Attributes of SteadyStateHypothesisScheduleProfile::CurrentFlow**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=none)
unit	0..1	<a href="#">UnitSymbol</a>	(const=A)

1574

### 1575 **3.108 Boolean primitive**

1576 A type with the value space "true" and "false".

### 1577 **3.109 (NC) PeakKind enumeration**

1578 Kind of time period with similar intensity.

1579 Table 139 shows all literals of PeakKind.

1580 **Table 139 – Literals of SteadyStateHypothesisScheduleProfile::PeakKind**

literal	value	description
offPeak		Off-peak refer to periods of lower demand for a particular service or commodity.
onPeak		Off-peak refer to periods of higher demand for a particular service or commodity.

1581

### 1582 **3.110 (NC) EnergyDemandKind enumeration**

1583 Kind of energy demand.

1584 Table 140 shows all literals of EnergyDemandKind.

1585 **Table 140 – Literals of SteadyStateHypothesisScheduleProfile::EnergyDemandKind**

literal	value	description
consumption		
production		
storage		
export		
import		

1586

### 1587 **3.111 String primitive**

1588 A string consisting of a sequence of characters. The character encoding is UTF-8. The string  
1589 length is unspecified and unlimited.

### 1590 **3.112 Time primitive**

1591 Time as "hh:mm:ss.sss", which conforms with ISO 8601. UTC time zone is specified as  
1592 "hh:mm:ss.sssZ". A local timezone relative UTC is specified as "hh:mm:ss.sss±hh:mm". The  
1593 second component (shown here as "ss.sss") could have any number of digits in its fractional  
1594 part to allow any kind of precision beyond seconds.

### 1595 **3.113 SynchronousMachineOperatingMode enumeration**

1596 Synchronous machine operating mode.

1597 Table 141 shows all literals of SynchronousMachineOperatingMode.



1598  
1599**Table 141 – Literals of  
SteadyStateHypothesisScheduleProfile::SynchronousMachineOperatingMode**

literal	value	description
generator		Operating as generator.
condenser		Operating as condenser.
motor		Operating as motor.

1600

**3.114 PU datatype**

1602 Per Unit - a positive or negative value referred to a defined base. Values typically range from -  
1603 10 to +10.

1604 Table 142 shows all attributes of PU.

1605

**Table 142 – Attributes of SteadyStateHypothesisScheduleProfile::PU**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
unit	0..1	<a href="#">UnitSymbol</a>	(const=none)
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=none)

1606

**3.115 Resistance datatype**

1608 Resistance (real part of impedance).

1609 Table 143 shows all attributes of Resistance.

1610

**Table 143 – Attributes of SteadyStateHypothesisScheduleProfile::Resistance**

name	mult	type	description
value	0..1	<a href="#">Float</a>	
unit	0..1	<a href="#">UnitSymbol</a>	(const=ohm)
multiplier	0..1	<a href="#">UnitMultiplier</a>	(const=none)

1611

**3.116 VsPpccControlKind enumeration**

1613 Types applicable to the control of real power and/or DC voltage by voltage source converter.

1614 Table 144 shows all literals of VsPpccControlKind.

1615

**Table 144 – Literals of SteadyStateHypothesisScheduleProfile::VsPpccControlKind**

literal	value	description
pPcc		Control is real power at point of common coupling. The target value is provided by ACDCCConverter.targetPpcc.
udc		Control is DC voltage with target value provided by ACDCCConverter.targetUdc.
pPccAndUdcDroop		Control is active power at point of common coupling and local DC voltage, with the droop. Target values are provided by ACDCCConverter.targetPpcc, ACDCCConverter.targetUdc and VsConverter.droop.
pPccAndUdcDroopWithCompensation		Control is active power at point of common coupling and compensated DC voltage, with the droop. Compensation factor is the resistance, as an approximation of the DC voltage of a common

literal	value	description
		(real or virtual) node in the DC network. Targets are provided by <code>ACDCConverter.targetPpcc</code> , <code>ACDCConverter.targetUdc</code> , <code>VsConverter.droop</code> and <code>VsConverter.droopCompensation</code> .
<code>pPccAndUdcDroopPilot</code>		Control is active power at point of common coupling and the pilot DC voltage, with the droop. The mode is used for Multi Terminal High Voltage DC (MTDC) systems where multiple HVDC Substations are connected to the HVDC transmission lines. The pilot voltage is then used to coordinate the control the DC voltage across the HVDC substations. Targets are provided by <code>ACDCConverter.targetPpcc</code> , <code>ACDCConverter.targetUdc</code> and <code>VsConverter.droop</code> .
<code>phasePcc</code>		Control is phase at point of common coupling. Target is provided by <code>VsConverter.targetPhasePcc</code> .

1616

1617

1618

1619

## **Annex A (informative): Sample data**

### **A.1 General**

1621 This Annex is designed to illustrate the profile by using fragments of sample data. It is not meant  
1622 to be a complete set of examples covering all possibilities of using the profile. Defining a  
1623 complete set of test data is considered a separate activity to be performed for the purpose of  
1624 setting up interoperability testing and conformity related to this profile.

### **A.2 Sample instance data**

1626 Test data files are available in the CIM EG SharePoint.

1627

1628