



European Network of
Transmission System Operators
for Electricity

STEADY STATE HYPOTHESIS SCHEDULE PROFILE SPECIFICATION

2025-02-13

APPROVED DOCUMENT
VERSION 1.0.2

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**

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

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.
1.0.2-alpha	2025-01-04		For CIM WG review. ICTC approval on 13 February 2025.

34

CONTENTS

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 (NC) EnergyConnectionRegularSchedule.....	26
62	3.16 (abstract) EnergyConnection root class.....	27
63	3.17 (NC) TapRegularSchedule	27
64	3.18 (abstract) TapChanger root class	28
65	3.19 (NC) SwitchRegularSchedule	28
66	3.20 (abstract) Switch root class	29
67	3.21 (NC) InServiceRegularSchedule.....	29
68	3.22 (abstract) Equipment root class.....	30
69	3.23 (NC) ControlAreaRegularSchedule.....	30
70	3.24 (abstract) ControlArea root class.....	31
71	3.25 (NC) SynchronousMachineRegularSchedule	31
72	3.26 (abstract) SynchronousMachine root class	32
73	3.27 (NC) AsynchronousMachineRegularSchedule	32
74	3.28 (abstract) AsynchronousMachine root class	33
75	3.29 (NC) ExternalNetworkInjectionRegularSchedule.....	33
76	3.30 (abstract) ExternalNetworkInjection root class.....	34
77	3.31 (abstract,NC) ACDCCConverterRegularSchedule	34

78	3.32	(NC) VsConverterRegularSchedule	35
79	3.33	(NC) CsConverterRegularSchedule	36
80	3.34	(abstract) VsConverter root class	38
81	3.35	(abstract) CsConverter root class	38
82	3.36	(NC) EquivalentInjectionRegularSchedule	38
83	3.37	(abstract) EquivalentInjection root class	39
84	3.38	(NC) EquivalentInjectionSchedule	39
85	3.39	(NC) EquivalentInjectionTimePoint root class	40
86	3.40	(NC) EnergyConnectionSchedule	40
87	3.41	(NC) EnergyConnectionTimePoint root class	41
88	3.42	(NC) TapSchedule	41
89	3.43	(NC) TapScheduleTimePoint root class	42
90	3.44	(NC) RegulatingControlSchedule	43
91	3.45	(NC) RegulatingControlTimePoint root class	43
92	3.46	(NC) TapChangerControlSchedule	44
93	3.47	(NC) VsConverterSchedule	44
94	3.48	(abstract,NC) ACDCTimePoint root class	45
95	3.49	(NC) VsConverterTimePoint	45
96	3.50	(NC) CsConverterTimePoint	46
97	3.51	(NC) CsConverterSchedule	47
98	3.52	(NC) SwitchSchedule	48
99	3.53	(NC) SwitchTimePoint root class	48
100	3.54	(NC) InServiceSchedule	49
101	3.55	(NC) InServiceTimePoint root class	49
102	3.56	(NC) ControlAreaSchedule	50
103	3.57	(NC) ControlAreaTimePoint root class	50
104	3.58	(NC) SynchronousMachineSchedule	51
105	3.59	(NC) SynchronousMachineTimePoint root class	51
106	3.60	(NC) AsynchronousMachineSchedule	52
107	3.61	(NC) AsynchronousMachineTimePoint root class	52
108	3.62	(NC) ExternalNetworkInjectionSchedule	53
109	3.63	(NC) ExternalNetworkInjectionTimePoint root class	53
110	3.64	(NC) BatteryUnitSchedule	54
111	3.65	(NC) BatteryUnitTimePoint root class	54
112	3.66	(abstract) BatteryUnit root class	55
113	3.67	(NC) VoltageLimitSchedule	55
114	3.68	(abstract) VoltageLimit root class	55
115	3.69	(NC) VoltageLimitTimePoint root class	56
116	3.70	(NC) ActivePowerLimitSchedule	56
117	3.71	(abstract) ActivePowerLimit root class	56
118	3.72	(NC) ActivePowerLimitTimePoint root class	57
119	3.73	(NC) ApparentPowerLimitSchedule	57
120	3.74	(abstract) ApparentPowerLimit root class	57
121	3.75	(NC) ApparentPowerLimitTimePoint root class	58
122	3.76	(NC) CurrentLimitSchedule	58
123	3.77	(abstract) CurrentLimit root class	58

124	3.78	(NC) CurrentLimitTimePoint root class	59
125	3.79	(NC) ShuntCompensatorSchedule.....	59
126	3.80	(NC) ShuntCompensatorTimePoint root class	59
127	3.81	(abstract) ShuntCompensator root class	60
128	3.82	(NC) StaticVarCompensatorSchedule	60
129	3.83	(NC) StaticVarCompensatorTimePoint root class	61
130	3.84	(abstract) StaticVarCompensator root class	61
131	3.85	(NC) GeneratingUnitSchedule	61
132	3.86	(NC) GeneratingUnitTimePoint root class.....	62
133	3.87	(abstract) GeneratingUnit root class	62
134	3.88	MonthDay primitive	63
135	3.89	ActivePower datatype	63
136	3.90	Float primitive	63
137	3.91	UnitMultiplier enumeration	63
138	3.92	UnitSymbol enumeration	63
139	3.93	ReactivePower datatype	64
140	3.94	Voltage datatype.....	65
141	3.95	DateTime primitive	65
142	3.96	ApparentPower datatype.....	65
143	3.97	AsynchronousMachineKind enumeration	65
144	3.98	(NC) DayOfWeekKind enumeration	65
145	3.99	Integer primitive	66
146	3.100	BatteryStateKind enumeration.....	66
147	3.101	RealEnergy datatype.....	66
148	3.102	CsOperatingModeKind enumeration	66
149	3.103	CsPpccControlKind enumeration.....	67
150	3.104	AngleDegrees datatype.....	67
151	3.105	CurrentFlow datatype.....	67
152	3.106	Boolean primitive	67
153	3.107	(NC) PeakKind enumeration.....	67
154	3.108	(NC) EnergyScenarioKind enumeration	68
155	3.109	String primitive.....	68
156	3.110	Time primitive	68
157	3.111	SynchronousMachineOperatingMode enumeration	68
158	3.112	PU datatype.....	68
159	3.113	Resistance datatype	69
160	3.114	VsPpccControlKind enumeration	69
161	3.115	VsQpccControlKind enumeration.....	70
162	3.116	PerCent datatype	70
163		Annex A (informative): Sample data	71
164	A.1	General.....	71
165	A.2	Sample instance data.....	71
166			
167		List of figures	

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	List of tables	
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 – Attributes of	
201	SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule	26
202	Table 17 – Association ends of	
203	SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule with other	
204	classes	26
205	Table 18 – Attributes of SteadyStateHypothesisScheduleProfile::TapRegularSchedule	27
206	Table 19 – Association ends of	
207	SteadyStateHypothesisScheduleProfile::TapRegularSchedule with other classes	28
208	Table 20 – Attributes of	
209	SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule	28
210	Table 21 – Association ends of	
211	SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule with other classes	29

212	Table 22 – Attributes of	
213	SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule	29
214	Table 23 – Association ends of	
215	SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule with other classes	30
216	Table 24 – Attributes of	
217	SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule	30
218	Table 25 – Association ends of	
219	SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule with other	
220	classes	30
221	Table 26 – Attributes of	
222	SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule	31
223	Table 27 – Association ends of	
224	SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule with	
225	other classes	32
226	Table 28 – Attributes of	
227	SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule	32
228	Table 29 – Association ends of	
229	SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule with	
230	other classes	33
231	Table 30 – Attributes of	
232	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule	33
233	Table 31 – Association ends of	
234	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule	
235	with other classes	34
236	Table 32 – Attributes of	
237	SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule	34
238	Table 33 – Association ends of	
239	SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule with other	
240	classes	35
241	Table 34 – Attributes of	
242	SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule	35
243	Table 35 – Association ends of	
244	SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule with other	
245	classes	36
246	Table 36 – Attributes of	
247	SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule	37
248	Table 37 – Association ends of	
249	SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule with other	
250	classes	38
251	Table 38 – Attributes of	
252	SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule	38
253	Table 39 – Association ends of	
254	SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule with	
255	other classes	39
256	Table 40 – Attributes of	
257	SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule	39
258	Table 41 – Association ends of	
259	SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule with other	
260	classes	40

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

308	Table 65 – Attributes of SteadyStateHypothesisScheduleProfile::CsConverterSchedule	47
309	Table 66 – Association ends of	
310	SteadyStateHypothesisScheduleProfile::CsConverterSchedule with other classes	47
311	Table 67 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchSchedule	48
312	Table 68 – Association ends of	
313	SteadyStateHypothesisScheduleProfile::SwitchSchedule with other classes	48
314	Table 69 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchTimePoint	48
315	Table 70 – Association ends of	
316	SteadyStateHypothesisScheduleProfile::SwitchTimePoint with other classes	49
317	Table 71 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceSchedule	49
318	Table 72 – Association ends of	
319	SteadyStateHypothesisScheduleProfile::InServiceSchedule with other classes	49
320	Table 73 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceTimePoint	49
321	Table 74 – Association ends of	
322	SteadyStateHypothesisScheduleProfile::InServiceTimePoint with other classes	50
323	Table 75 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaSchedule	50
324	Table 76 – Association ends of	
325	SteadyStateHypothesisScheduleProfile::ControlAreaSchedule with other classes	50
326	Table 77 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint	50
327	Table 78 – Association ends of	
328	SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint with other classes	51
329	Table 79 – Attributes of	
330	SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule	51
331	Table 80 – Association ends of	
332	SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule with other	
333	classes	51
334	Table 81 – Attributes of	
335	SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint	51
336	Table 82 – Association ends of	
337	SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint with other	
338	classes	52
339	Table 83 – Attributes of	
340	SteadyStateHypothesisScheduleProfile::AsynchronousMachineSchedule	52
341	Table 84 – Association ends of	
342	SteadyStateHypothesisScheduleProfile::AsynchronousMachineSchedule with other	
343	classes	52
344	Table 85 – Attributes of	
345	SteadyStateHypothesisScheduleProfile::AsynchronousMachineTimePoint	52
346	Table 86 – Association ends of	
347	SteadyStateHypothesisScheduleProfile::AsynchronousMachineTimePoint with other	
348	classes	53
349	Table 87 – Attributes of	
350	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionSchedule	53
351	Table 88 – Association ends of	
352	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionSchedule with other	
353	classes	53

354	Table 89 – Attributes of	
355	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint	53
356	Table 90 – Association ends of	
357	SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint with other	
358	classes	54
359	Table 91 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule	54
360	Table 92 – Association ends of	
361	SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule with other classes	54
362	Table 93 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint	55
363	Table 94 – Association ends of	
364	SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint with other classes	55
365	Table 95 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule	55
366	Table 96 – Association ends of	
367	SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule with other classes	55
368	Table 97 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint	56
369	Table 98 – Association ends of	
370	SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint with other classes	56
371	Table 99 – Attributes of	
372	SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule	56
373	Table 100 – Association ends of	
374	SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule with other classes	56
375	Table 101 – Attributes of	
376	SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint	57
377	Table 102 – Association ends of	
378	SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint with other classes	57
379	Table 103 – Attributes of	
380	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule	57
381	Table 104 – Association ends of	
382	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule with other	
383	classes	57
384	Table 105 – Attributes of	
385	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint	58
386	Table 106 – Association ends of	
387	SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint with other	
388	classes	58
389	Table 107 – Attributes of	
390	SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule	58
391	Table 108 – Association ends of	
392	SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule with other classes	58
393	Table 109 – Attributes of	
394	SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint	59
395	Table 110 – Association ends of	
396	SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint with other classes	59
397	Table 111 – Attributes of	
398	SteadyStateHypothesisScheduleProfile::ShuntCompensatorSchedule	59

399	Table 112 – Association ends of	
400	SteadyStateHypothesisScheduleProfile::ShuntCompensatorSchedule with other	
401	classes	59
402	Table 113 – Attributes of	
403	SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint	60
404	Table 114 – Association ends of	
405	SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint with other	
406	classes	60
407	Table 115 – Attributes of	
408	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule	60
409	Table 116 – Association ends of	
410	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule with other	
411	classes	61
412	Table 117 – Attributes of	
413	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint	61
414	Table 118 – Association ends of	
415	SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint with other	
416	classes	61
417	Table 119 – Attributes of	
418	SteadyStateHypothesisScheduleProfile::GeneratingUnitSchedule	62
419	Table 120 – Association ends of	
420	SteadyStateHypothesisScheduleProfile::GeneratingUnitSchedule with other classes	62
421	Table 121 – Attributes of	
422	SteadyStateHypothesisScheduleProfile::GeneratingUnitTimePoint	62
423	Table 122 – Association ends of	
424	SteadyStateHypothesisScheduleProfile::GeneratingUnitTimePoint with other classes	62
425	Table 123 – Attributes of SteadyStateHypothesisScheduleProfile::ActivePower	63
426	Table 124 – Literals of SteadyStateHypothesisScheduleProfile::UnitMultiplier	63
427	Table 125 – Literals of SteadyStateHypothesisScheduleProfile::UnitSymbol	64
428	Table 126 – Attributes of SteadyStateHypothesisScheduleProfile::ReactivePower	64
429	Table 127 – Attributes of SteadyStateHypothesisScheduleProfile::Voltage	65
430	Table 128 – Attributes of SteadyStateHypothesisScheduleProfile::ApparentPower	65
431	Table 129 – Literals of	
432	SteadyStateHypothesisScheduleProfile::AsynchronousMachineKind	65
433	Table 130 – Literals of SteadyStateHypothesisScheduleProfile::DayOfWeekKind	65
434	Table 131 – Literals of SteadyStateHypothesisScheduleProfile::BatteryStateKind	66
435	Table 132 – Attributes of SteadyStateHypothesisScheduleProfile::RealEnergy	66
436	Table 133 – Literals of SteadyStateHypothesisScheduleProfile::CsOperatingModeKind	67
437	Table 134 – Literals of SteadyStateHypothesisScheduleProfile::CsPpccControlKind	67
438	Table 135 – Attributes of SteadyStateHypothesisScheduleProfile::AngleDegrees	67
439	Table 136 – Attributes of SteadyStateHypothesisScheduleProfile::CurrentFlow	67
440	Table 137 – Literals of SteadyStateHypothesisScheduleProfile::PeakKind	68
441	Table 138 – Literals of SteadyStateHypothesisScheduleProfile::EnergyScenarioKind	68
442	Table 139 – Literals of	
443	SteadyStateHypothesisScheduleProfile::SynchronousMachineOperatingMode	68

444	Table 140 – Attributes of SteadyStateHypothesisScheduleProfile::PU	68
445	Table 141 – Attributes of SteadyStateHypothesisScheduleProfile::Resistance	69
446	Table 142 – Literals of SteadyStateHypothesisScheduleProfile::VsPpccControlKind	69
447	Table 143 – Literals of SteadyStateHypothesisScheduleProfile::VsQpccControlKind	70
448	Table 144 – Attributes of SteadyStateHypothesisScheduleProfile::PerCent	70
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_NC23v69_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: 1.0.2
- 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

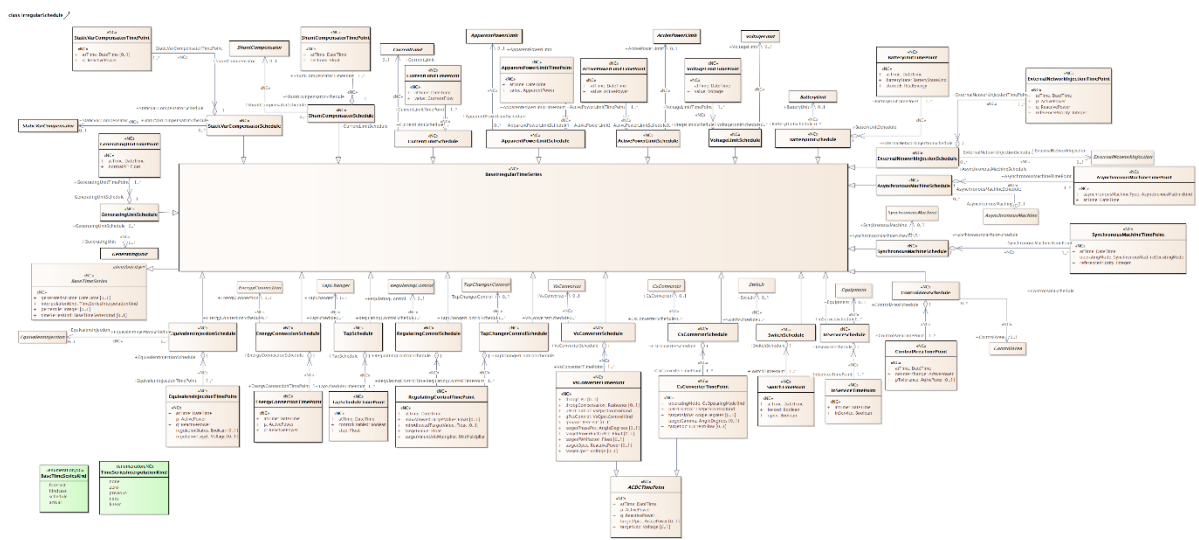
629 **2.4.2 Reference metadata**

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**

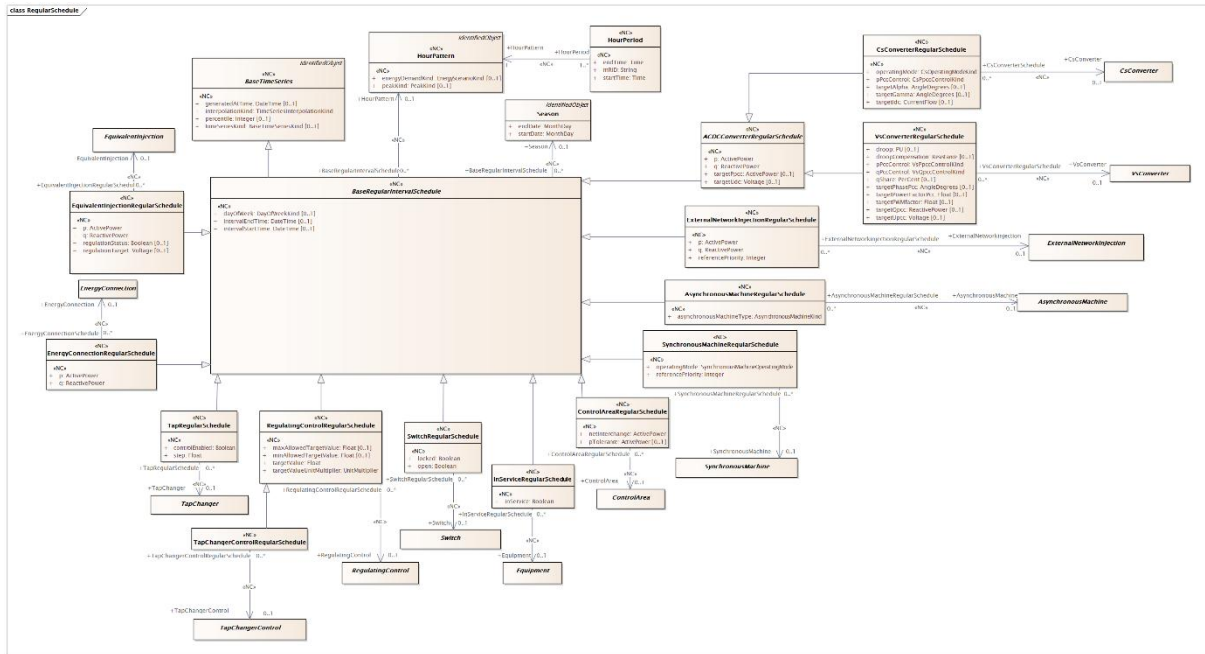
640 This package contains steady state hypothesis schedule profile.



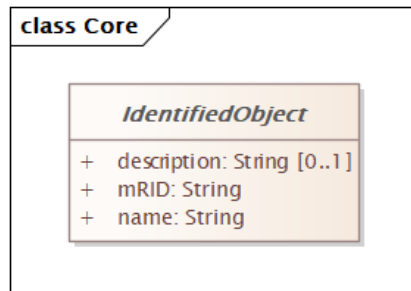
641

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	String	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	String	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	String	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	TimeSeriesInterpolationKind	(NC) Kind of interpolation done between time point.
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) Kind of base time series.
generatedAtTime	0..1	DateTime	(NC) The time this time series (entity) come to existents and available for use.
percentile	0..1	Integer	(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	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

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	MonthDay	Date season ends.
startDate	1..1	MonthDay	Date season starts.
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

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	PeakKind	(NC) Type of peak or intensity that the pattern is valid for.
energyDemandKind	0..1	EnergyScenarioKind	(NC) Type of energy demand that the pattern is valid for.
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

673

674 **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	DayOfWeekKind	(NC) Day of the week for which the schedule is valid for.
intervalStartTime	0..1	DateTime	(NC) Interval start time for which the schedule is valid for.
intervalEndTime	0..1	DateTime	(NC) Interval end time for which the schedule is valid for.
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

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	Season	(NC) Season associated with a base regular interval schedule.
0..*	HourPattern	0..1	HourPattern	(NC) HourPattern that has base regular interval schedule.

685

686 **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	String	(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	Time	(NC) Time the period start and including, e.g. 12:00 which means it include the time of 12:00.
endTime	1..1	Time	(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	HourPattern	(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	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

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	Float	(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	UnitMultiplier	(NC) Specify the multiplier for used for the targetValue.
maxAllowedTargetValue	0..1	Float	(NC) Maximum allowed target value (RegulatingControl.targetValue).

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

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	RegulatingControl	(NC) Regulating control which has RegulatingControlRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

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	Float	(NC) inherited from: RegulatingControlRegularSchedule
targetValueUnitMultiplier	1..1	UnitMultiplier	(NC) inherited from: RegulatingControlRegularSchedule
maxAllowedTargetValue	0..1	Float	(NC) inherited from: RegulatingControlRegularSchedule
minAllowedTargetValue	0..1	Float	(NC) inherited from: RegulatingControlRegularSchedule
dayOfWeek	0..1	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

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	TapChangerControl	(NC) Tap changer control which has TapChangerControlRegularSchedule.
0..*	RegulatingControl	0..1	RegulatingControl	(NC) inherited from: RegulatingControlRegularSchedule

mult from	name	mult to	type	description
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

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 (NC) EnergyConnectionRegularSchedule**

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

769 Regular schedule for energy connection.

770 Table 16 shows all attributes of EnergyConnectionRegularSchedule.

771

772

**Table 16 – Attributes of
SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule**

name	mult	type	description
p	1..1	ActivePower	(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	ReactivePower	(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	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

773

774 Table 17 shows all association ends of EnergyConnectionRegularSchedule with other classes.

775

776

777

**Table 17 – Association ends of
SteadyStateHypothesisScheduleProfile::EnergyConnectionRegularSchedule with other
classes**

mult from	name	mult to	type	description
0..*	EnergyConnection	0..1	EnergyConnection	(NC) EnergyConnection which has EnergyConnectionSchedule.

mult from	name	mult to	type	description
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

778

779 **3.16 (abstract) EnergyConnection root class**

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

781 **3.17 (NC) TapRegularSchedule**782 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

783 Regular schedule for tap.

784 Table 18 shows all attributes of TapRegularSchedule.

785 **Table 18 – Attributes of SteadyStateHypothesisScheduleProfile::TapRegularSchedule**

name	mult	type	description
controlEnabled	1..1	Boolean	(NC) Specifies the regulation status of the equipment. True is regulating, false is not regulating.
step	1..1	Float	(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 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	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

786

787 Table 19 shows all association ends of TapRegularSchedule with other classes.

788
789**Table 19 – Association ends of
SteadyStateHypothesisScheduleProfile::TapRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	TapChanger	0..1	TapChanger	(NC) Tap changer which has TapRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

790

791 **3.18 (abstract) TapChanger root class**

792 Mechanism for changing transformer winding tap positions.

793 **3.19 (NC) SwitchRegularSchedule**794 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

795 Regular schedule for switch.

796 Table 20 shows all attributes of SwitchRegularSchedule.

797

**Table 20 – Attributes of
SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule**

798

name	mult	type	description
open	1..1	Boolean	(NC) The attribute tells if the switch is considered open when used as input to topology processing.
locked	1..1	Boolean	(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; - 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	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

799

800 Table 21 shows all association ends of SwitchRegularSchedule with other classes.

801 **Table 21 – Association ends of**
802 **SteadyStateHypothesisScheduleProfile::SwitchRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	Switch	0..1	Switch	(NC) Switch which has SwitchRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

803

804 **3.20 (abstract) Switch root class**

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

809 **3.21 (NC) InServiceRegularSchedule**

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

811 Regular schedule for elements having in service.

812 Table 22 shows all attributes of InServiceRegularSchedule.

813

814 **Table 22 – Attributes of**
SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule

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

815

816 Table 23 shows all association ends of InServiceRegularSchedule with other classes.

817 **Table 23 – Association ends of**
818 **SteadyStateHypothesisScheduleProfile::InServiceRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	Equipment	0..1	Equipment	(NC) Equipment which has InServiceRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

819

820 **3.22 (abstract) Equipment root class**

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

822 **3.23 (NC) ControlAreaRegularSchedule**823 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

824 Regular schedule for control area.

825 Table 24 shows all attributes of ControlAreaRegularSchedule.

826

827

Table 24 – Attributes of
SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule

name	mult	type	description
netInterchange	1..1	ActivePower	(NC) The specified positive net interchange into the control area, i.e. positive sign means flow into the area.
pTolerance	0..1	ActivePower	(NC) Active power net interchange tolerance. The attribute shall be a positive value or zero.
dayOfWeek	0..1	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

828

829 Table 25 shows all association ends of ControlAreaRegularSchedule with other classes.

830 **Table 25 – Association ends of**
831 **SteadyStateHypothesisScheduleProfile::ControlAreaRegularSchedule with other**
832 **classes**

mult from	name	mult to	type	description
0..*	ControlArea	0..1	ControlArea	(NC) ControlArea which has ControlAreaRegularSchedule.

mult from	name	mult to	type	description
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

833

834 3.24 (abstract) ControlArea root class

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

- 842 1. The control area orientation for net interchange is positive for an import, negative for an
843 export.
- 844 2. The control area net interchange is determined by summing flows in Terminals. The
845 Terminals are identified by creating a set of TieFlow objects associated with a ControlArea
846 object. Each TieFlow object identifies one Terminal.
- 847 3. In a single network model, a tie between two control areas must be modelled in both control
848 area specifications, such that the two representations of the tie flow sum to zero.
- 849 4. The normal orientation of Terminal flow is positive for flow into the conducting equipment
850 that owns the Terminal. (i.e. flow from a bus into a device is positive.) However, the orientation
851 of each flow in the control area specification must align with the control area convention, i.e.
852 import is positive. If the orientation of the Terminal flow referenced by a TieFlow is positive into
853 the control area, then this is confirmed by setting TieFlow.positiveFlowIn flag TRUE. If not, the
854 orientation must be reversed by setting the TieFlow.positiveFlowIn flag FALSE.

855 3.25 (NC) SynchronousMachineRegularSchedule

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

857 Regular schedule for synchronous machine.

858 Table 26 shows all attributes of SynchronousMachineRegularSchedule.

859

Table 26 – Attributes of

860

SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule

name	mult	type	description
operatingMode	1..1	SynchronousMachineOperatingMode	(NC) Current mode of operation.
referencePriority	1..1	Integer	(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	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries

name	mult	type	description
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

861
862 Table 27 shows all association ends of SynchronousMachineRegularSchedule with other
863 classes.

864 **Table 27 – Association ends of**
865 **SteadyStateHypothesisScheduleProfile::SynchronousMachineRegularSchedule with**
866 **other classes**

mult from	name	mult to	type	description
0..*	SynchronousMachine	0..1	SynchronousMachine	(NC) SynchronousMachine which has SynchronousMachineRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

867
868 **3.26 (abstract) SynchronousMachine root class**

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

871 **3.27 (NC) AsynchronousMachineRegularSchedule**

872 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)
873 Regular schedule for asynchronous machine.

874 Table 28 shows all attributes of AsynchronousMachineRegularSchedule.

875 **Table 28 – Attributes of**
876 **SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule**

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

877

878 Table 29 shows all association ends of AsynchronousMachineRegularSchedule with other
879 classes.

880 **Table 29 – Association ends of**
881 **SteadyStateHypothesisScheduleProfile::AsynchronousMachineRegularSchedule with**
882 **other classes**

mult from	name	mult to	type	description
0..*	AsynchronousMachine	0..1	AsynchronousMachine	(NC) AsynchronousMachine which has AsynchronousMachineRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

883

884 3.28 (abstract) AsynchronousMachine root class

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

888 3.29 (NC) ExternalNetworkInjectionRegularSchedule

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

890 Regular schedule for external network injection.

891 Table 30 shows all attributes of ExternalNetworkInjectionRegularSchedule.

892 **Table 30 – Attributes of**
893 **SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule**

name	mult	type	description
referencePriority	1..1	Integer	(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	ActivePower	(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	ReactivePower	(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	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject

name	mult	type	description
name	1..1	String	inherited from: IdentifiedObject

894

895 Table 31 shows all association ends of ExternalNetworkInjectionRegularSchedule with other
896 classes.

897

898 **Table 31 – Association ends of**
899 **SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionRegularSchedule with other classes**

mult from	name	mult to	type	description
0..*	ExternalNetworkInjection	0..1	ExternalNetworkInjection	(NC) External network injection which has ExternalNetworkInjectionRegularSchedule
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

900

901 3.30 (abstract) ExternalNetworkInjection root class

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

903 3.31 (abstract,NC) ACDCConverterRegularSchedule

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

905 Regular schedule for ACDC converter.

906 Table 32 shows all attributes of ACDCConverterRegularSchedule.

907

908 **Table 32 – Attributes of**
SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule

name	mult	type	description
p	1..1	ActivePower	(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	ReactivePower	(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	ActivePower	(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	Voltage	(NC) Target value for DC voltage magnitude. The attribute shall be a positive value.
dayOfWeek	0..1	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

909

910 Table 33 shows all association ends of ACDCConverterRegularSchedule with other classes.

911

912 **Table 33 – Association ends of**
913 **SteadyStateHypothesisScheduleProfile::ACDCConverterRegularSchedule with other**
classes

mult from	name	mult to	type	description
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

914

915 **3.32 (NC) VsConverterRegularSchedule**916 Inheritance path = [ACDCConverterRegularSchedule](#) : [BaseRegularIntervalSchedule](#) :
917 [BaseTimeSeries](#) : [IdentifiedObject](#)

918 Regular schedule for VS converter.

919 Table 34 shows all attributes of VsConverterRegularSchedule.

920

921 **Table 34 – Attributes of**
SteadyStateHypothesisScheduleProfile::VsConverterRegularSchedule

name	mult	type	description
droop	0..1	PU	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	Resistance	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	VsPpccControlKind	Kind of control of real power and/or DC voltage.
qPccControl	1..1	VsQpccControlKind	Kind of reactive power control.
qShare	0..1	PerCent	Reactive power sharing factor among parallel converters on U_{ac} control. The attribute shall be a positive value or zero.
targetQpcc	0..1	ReactivePower	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	Voltage	Voltage target in AC grid, at point of common coupling. The attribute shall be a positive value.
targetPowerFactorPcc	0..1	Float	Power factor target at the AC side, at point of common coupling. The attribute shall be a positive value.

name	mult	type	description
targetPhasePcc	0..1	AngleDegrees	Phase target at AC side, at point of common coupling. The attribute shall be a positive value.
targetPWMfactor	0..1	Float	Magnitude of pulse-modulation factor. The attribute shall be a positive value.
p	1..1	ActivePower	(NC) inherited from: ACDCCConverterRegularSchedule
q	1..1	ReactivePower	(NC) inherited from: ACDCCConverterRegularSchedule
targetPpcc	0..1	ActivePower	(NC) inherited from: ACDCCConverterRegularSchedule
targetUdc	0..1	Voltage	(NC) inherited from: ACDCCConverterRegularSchedule
dayOfWeek	0..1	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

922

923

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

924

925

926

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

mult from	name	mult to	type	description
0..*	VsConverter	0..1	VsConverter	(NC) VsConverter which has VsConverterRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

927

928

3.33 (NC) CsConverterRegularSchedule

929

930

931

932

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

Regular schedule for CS converter.

Table 36 shows all attributes of CsConverterRegularSchedule.

933
934**Table 36 – Attributes of
SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule**

name	mult	type	description
operatingMode	1..1	CsOperatingModeKind	(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	CsPpccControlKind	(NC) Kind of active power control.
targetAlpha	0..1	AngleDegrees	(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	AngleDegrees	(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	CurrentFlow	(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	ActivePower	(NC) inherited from: ACDCConverterRegularSchedule
q	1..1	ReactivePower	(NC) inherited from: ACDCConverterRegularSchedule
targetPpcc	0..1	ActivePower	(NC) inherited from: ACDCConverterRegularSchedule
targetUdc	0..1	Voltage	(NC) inherited from: ACDCConverterRegularSchedule
dayOfWeek	0..1	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

935
936

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

937
938
939

**Table 37 – Association ends of
SteadyStateHypothesisScheduleProfile::CsConverterRegularSchedule with other
classes**

mult from	name	mult to	type	description
0..*	CsConverter	0..1	CsConverter	(NC) CsConverter which has CsConverterRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

940

941 **3.34 (abstract) VsConverter root class**

942 DC side of the voltage source converter (VSC).

943 **3.35 (abstract) CsConverter root class**

944 DC side of the current source converter (CSC).

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

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

958 The attributes minAlpha and maxAlpha define the range of firing angles for rectifier operation
959 between which no discrete tap changer action takes place. The range is typically 10-18 degrees.
960 The attributes minGamma and maxGamma define the range of extinction angles for inverter
961 operation between which no discrete tap changer action takes place. The range is typically 17-
962 20 degrees.

963 **3.36 (NC) EquivalentInjectionRegularSchedule**964 Inheritance path = [BaseRegularIntervalSchedule](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

965 Regular schedule for equivalent injection.

966 Table 38 shows all attributes of EquivalentInjectionRegularSchedule.

967

968

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

name	mult	type	description
regulationStatus	0..1	Boolean	(NC) Specifies the regulation status of the EquivalentInjection. True is regulating. False is not regulating.
regulationTarget	0..1	Voltage	(NC) The target voltage for voltage regulation. The attribute shall be a positive value.
p	1..1	ActivePower	(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.

name	mult	type	description
q	1..1	ReactivePower	(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	DayOfWeekKind	(NC) inherited from: BaseRegularIntervalSchedule
intervalStartTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
intervalEndTime	0..1	DateTime	(NC) inherited from: BaseRegularIntervalSchedule
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

969

970

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

971

972

973

Table 39 – Association ends of SteadyStateHypothesisScheduleProfile::EquivalentInjectionRegularSchedule with other classes

mult from	name	mult to	type	description
0..*	EquivalentInjection	0..1	EquivalentInjection	(NC) EquivalentInjection which has EquivalentInjectionRegularSchedule.
0..*	Season	0..1	Season	(NC) inherited from: BaseRegularIntervalSchedule
0..*	HourPattern	0..1	HourPattern	(NC) inherited from: BaseRegularIntervalSchedule

974

975 3.37 (abstract) EquivalentInjection root class

976

977

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

978 3.38 (NC) EquivalentInjectionSchedule

979

980

981

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

Regular schedule for equivalent injection.

Table 40 shows all attributes of EquivalentInjectionSchedule.

982

983

Table 40 – Attributes of SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries

name	mult	type	description
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

984
985 Table 41 shows all association ends of EquivalentInjectionSchedule with other classes.

986 **Table 41 – Association ends of**
987 **SteadyStateHypothesisScheduleProfile::EquivalentInjectionSchedule with other classes**

mult from	name	mult to	type	description
0..*	EquivalentInjection	0..1	EquivalentInjection	(NC) Equivalent injection which has equivalent injection schedules.

988
989 **3.39 (NC) EquivalentInjectionTimePoint root class**

990 Equivalent injection values for a given point in time.
991 Table 42 shows all attributes of EquivalentInjectionTimePoint.

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

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
regulationStatus	0..1	Boolean	(NC) Specifies the regulation status of the EquivalentInjection. True is regulating. False is not regulating.
regulationTarget	0..1	Voltage	(NC) The target voltage for voltage regulation. The attribute shall be a positive value.
p	1..1	ActivePower	(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	ReactivePower	(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.

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

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

mult from	name	mult to	type	description
1..*	EquivalentInjectionSchedule	1..1	EquivalentInjectionSchedule	(NC) The EquivalentInjection schedule that has this time point.

999
1000 **3.40 (NC) EnergyConnectionSchedule**

1001 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)
1002 Schedule for energy connection.
1003 Table 44 shows all attributes of EnergyConnectionSchedule.

1004
1005

**Table 44 – Attributes of
SteadyStateHypothesisScheduleProfile::EnergyConnectionSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1006
1007

Table 45 shows all association ends of EnergyConnectionSchedule with other classes.

1008
1009

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

mult from	name	mult to	type	description
0..*	EnergyConnection	0..1	EnergyConnection	(NC) Energy connection which has energy connection schedules.

1010

1011 3.41 (NC) EnergyConnectionTimePoint root class

1012 Energy connection values for a given point in time.

1013 Table 46 shows all attributes of EnergyConnectionTimePoint.

1014
1015

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

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
p	1..1	ActivePower	(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	ReactivePower	(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.

1016

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

1018
1019

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

mult from	name	mult to	type	description
1..*	EnergyConnectionSchedule	1..1	EnergyConnectionSchedule	(NC) The energy connection schedule that has this time point.

1020

1021 3.42 (NC) TapSchedule

1022 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1023 Schedule for tap.

1024 Table 48 shows all attributes of TapSchedule.

1025 **Table 48 – Attributes of SteadyStateHypothesisScheduleProfile::TapSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1026

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

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

mult from	name	mult to	type	description
0..*	TapChanger	0..1	TapChanger	(NC) Tap changer which has tap schedules.

1030

1031 3.43 (NC) TapScheduleTimePoint root class

1032 Tap schedule values for a given point in time.

1033 Table 50 shows all attributes of TapScheduleTimePoint.

1034 **Table 50 – Attributes of SteadyStateHypothesisScheduleProfile::TapScheduleTimePoint**

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
controlEnabled	1..1	Boolean	(NC) Specifies the regulation status of the equipment. True is regulating, false is not regulating.
step	1..1	Float	(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 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.

1035

1036 Table 51 shows all association ends of TapScheduleTimePoint with other classes.

1037 **Table 51 – Association ends of**
1038 **SteadyStateHypothesisScheduleProfile::TapScheduleTimePoint with other classes**

mult from	name	mult to	type	description
1..*	TapSchedule	1..1	TapSchedule	(NC) The tap schedule that has this time point.

1039

1040 **3.44 (NC) RegulatingControlSchedule**1041 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1042 Schedule for regulating control.

1043 Table 52 shows all attributes of RegulatingControlSchedule.

1044

1045

**Table 52 – Attributes of
SteadyStateHypothesisScheduleProfile::RegulatingControlSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1046

1047

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

1048

1049

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

mult from	name	mult to	type	description
0..*	RegulatingControl	0..1	RegulatingControl	(NC) Regulating control which has regulating control schedules.

1050

1051 **3.45 (NC) RegulatingControlTimePoint root class**

1052 Regulating control values for a given point in time.

1053 Table 54 shows all attributes of RegulatingControlTimePoint.

1054

1055

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

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
targetValue	1..1	Float	(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	UnitMultiplier	(NC) Specify the multiplier for used for the targetValue.
maxAllowedTargetValue	0..1	Float	(NC) Maximum allowed target value (RegulatingControl.targetValue).
minAllowedTargetValue	0..1	Float	(NC) Minimum allowed target value (RegulatingControl.targetValue).

1056

1057

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

1058 **Table 55 – Association ends of**
1059 **SteadyStateHypothesisScheduleProfile::RegulatingControlTimePoint with other classes**

mult from	name	mult to	type	description
1..*	RegulatingControlSchedule	1..1	RegulatingControlSchedule	(NC) The regulating control schedule that has this time point.
1..*	TapChangerControlSchedule	1..1	TapChangerControlSchedule	(NC) The tap changer control schedule that has this time point.

1060

1061 3.46 (NC) TapChangerControlSchedule

1062 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1063 Schedule for tap changer control.

1064 Table 56 shows all attributes of TapChangerControlSchedule.

1065 **Table 56 – Attributes of**
1066 **SteadyStateHypothesisScheduleProfile::TapChangerControlSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1067

1068 Table 57 shows all association ends of TapChangerControlSchedule with other classes.

1069 **Table 57 – Association ends of**
1070 **SteadyStateHypothesisScheduleProfile::TapChangerControlSchedule with other**
1071 **classes**

mult from	name	mult to	type	description
0..*	TapChangerControl	0..1	TapChangerControl	(NC) Tap changer control which has TapChangerControlSchedule.

1072

1073 3.47 (NC) VsConverterSchedule

1074 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1075 Schedule for VS converter.

1076 Table 58 shows all attributes of VsConverterSchedule.

1077 **Table 58 – Attributes of SteadyStateHypothesisScheduleProfile::VsConverterSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries

name	mult	type	description
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1078

1079

Table 59 shows all association ends of VsConverterSchedule with other classes.

1080

1081

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

mult from	name	mult to	type	description
0..*	VsConverter	0..1	VsConverter	(NC) Vs converter which has Vs converter schedules.

1082

1083

3.48 (abstract,NC) ACDCTimePoint root class

1084

ACDC values for a given point in time.

1085

Table 60 shows all attributes of ACDCTimePoint.

1086

Table 60 – Attributes of SteadyStateHypothesisScheduleProfile::ACDCTimePoint

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
p	1..1	ActivePower	(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	ReactivePower	(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	ActivePower	(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	Voltage	(NC) Target value for DC voltage magnitude. The attribute shall be a positive value.

1087

1088

3.49 (NC) VsConverterTimePoint

1089

Inheritance path = [ACDCTimePoint](#)

1090

VS converter values for a given point in time.

1091

Table 61 shows all attributes of VsConverterTimePoint.

1092

Table 61 – Attributes of SteadyStateHypothesisScheduleProfile::VsConverterTimePoint

name	mult	type	description
droop	0..1	PU	(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	Resistance	(NC) Compensation constant. Used to compensate for voltage drop when controlling voltage at a distant bus. The attribute shall be a positive value.

name	mult	type	description
pPccControl	1..1	VsPpccControlKind	(NC) Kind of control of real power and/or DC voltage.
qPccControl	1..1	VsQpccControlKind	(NC) Kind of reactive power control.
qShare	0..1	PerCent	(NC) Reactive power sharing factor among parallel converters on Uac control. The attribute shall be a positive value or zero.
targetQpcc	0..1	ReactivePower	(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	Voltage	(NC) Voltage target in AC grid, at point of common coupling. The attribute shall be a positive value.
targetPowerFactorPcc	0..1	Float	(NC) Power factor target at the AC side, at point of common coupling. The attribute shall be a positive value.
targetPhasePcc	0..1	AngleDegrees	(NC) Phase target at AC side, at point of common coupling. The attribute shall be a positive value.
targetPWMfactor	0..1	Float	(NC) Magnitude of pulse-modulation factor. The attribute shall be a positive value.
atTime	1..1	DateTime	(NC) inherited from: ACDCTimePoint
p	1..1	ActivePower	(NC) inherited from: ACDCTimePoint
q	1..1	ReactivePower	(NC) inherited from: ACDCTimePoint
targetPpcc	0..1	ActivePower	(NC) inherited from: ACDCTimePoint
targetUdc	0..1	Voltage	(NC) inherited from: ACDCTimePoint

1093

1094

Table 62 shows all association ends of VsConverterTimePoint with other classes.

1095

1096

Table 62 – Association ends of SteadyStateHypothesisScheduleProfile::VsConverterTimePoint with other classes

mult from	name	mult to	type	description
1..*	VsConverterSchedule	1..1	VsConverterSchedule	(NC) The VS converter schedule that has this time point.

1097

1098

3.50 (NC) CsConverterTimePoint

1099

Inheritance path = [ACDCTimePoint](#)

1100

CsConverter values for a given point in time.

1101

Table 63 shows all attributes of CsConverterTimePoint.

1102

Table 63 – Attributes of SteadyStateHypothesisScheduleProfile::CsConverterTimePoint

name	mult	type	description
operatingMode	1..1	CsOperatingModeKind	(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	CsPpccControlKind	(NC) Kind of active power control.
targetAlpha	0..1	AngleDegrees	(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

name	mult	type	description
			minAlpha<=targetAlpha<=maxAlpha. The attribute shall be a positive value.
targetGamma	0..1	AngleDegrees	(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 minGamma<=targetGamma<=maxGamma. The attribute shall be a positive value.
targetIdc	0..1	CurrentFlow	(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	DateTime	(NC) inherited from: ACDCTimePoint
p	1..1	ActivePower	(NC) inherited from: ACDCTimePoint
q	1..1	ReactivePower	(NC) inherited from: ACDCTimePoint
targetPpcc	0..1	ActivePower	(NC) inherited from: ACDCTimePoint
targetUdc	0..1	Voltage	(NC) inherited from: ACDCTimePoint

1103

1104

Table 64 shows all association ends of CsConverterTimePoint with other classes.

1105

1106

Table 64 – Association ends of SteadyStateHypothesisScheduleProfile::CsConverterTimePoint with other classes

mult from	name	mult to	type	description
1..*	CsConverterSchedule	1..1	CsConverterSchedule	(NC) The CS converter schedule that has this time point.

1107

1108

3.51 (NC) CsConverterSchedule

1109

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

1110

Schedule for CS converter.

1111

Table 65 shows all attributes of CsConverterSchedule.

1112

Table 65 – Attributes of SteadyStateHypothesisScheduleProfile::CsConverterSchedule

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1113

1114

Table 66 shows all association ends of CsConverterSchedule with other classes.

1115

1116

Table 66 – Association ends of SteadyStateHypothesisScheduleProfile::CsConverterSchedule with other classes

mult from	name	mult to	type	description
0..*	CsConverter	0..1	CsConverter	(NC) Cs converter which has Cs converter schedules.

1117

1118 **3.52 (NC) SwitchSchedule**1119 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1120 Schedule for switch.

1121 Table 67 shows all attributes of SwitchSchedule.

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

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1123

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

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

1126

mult from	name	mult to	type	description
0..*	Switch	0..1	Switch	(NC) Switch which has switch schedules.

1127

1128 **3.53 (NC) SwitchTimePoint root class**

1129 Switch values for a given point in time.

1130 Table 69 shows all attributes of SwitchTimePoint.

1131 **Table 69 – Attributes of SteadyStateHypothesisScheduleProfile::SwitchTimePoint**

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
open	1..1	Boolean	(NC) The attribute tells if the switch is considered open when used as input to topology processing.
locked	1..1	Boolean	(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; - locked=false and Switch.open=true. The resulting state is open; - locked=false and Switch.open=false. The resulting state is closed.

1132

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

1134
1135**Table 70 – Association ends of
SteadyStateHypothesisScheduleProfile::SwitchTimePoint with other classes**

mult from	name	mult to	type	description
1..*	SwitchSchedule	1..1	SwitchSchedule	The switch schedule that has this time point.

1136

3.54 (NC) InServiceSchedule1138 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1139 Schedule for elements having in service.

1140 Table 71 shows all attributes of InServiceSchedule.

Table 71 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceSchedule

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1142

1143 Table 72 shows all association ends of InServiceSchedule with other classes.

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

mult from	name	mult to	type	description
0..*	Equipment	0..1	Equipment	(NC) Equipment which has equipment schedules.

1146

3.55 (NC) InServiceTimePoint root class

1148 In service values for a given point in time.

1149 Table 73 shows all attributes of InServiceTimePoint.

Table 73 – Attributes of SteadyStateHypothesisScheduleProfile::InServiceTimePoint

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
inService	1..1	Boolean	(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 the equipment is treated by network applications as if it is not in the model.

1151

1152 Table 74 shows all association ends of InServiceTimePoint with other classes.

1153
1154**Table 74 – Association ends of
SteadyStateHypothesisScheduleProfile::InServiceTimePoint with other classes**

mult from	name	mult to	type	description
1..*	InServiceSchedule	1..1	InServiceSchedule	(NC) The in service schedule that has this time point.

1155

3.56 (NC) ControlAreaSchedule1157 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1158 Schedule for control area.

1159 Table 75 shows all attributes of ControlAreaSchedule.

Table 75 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaSchedule

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1161

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

**Table 76 – Association ends of
SteadyStateHypothesisScheduleProfile::ControlAreaSchedule with other classes**

mult from	name	mult to	type	description
0..*	ControlArea	0..1	ControlArea	(NC) Control area which has control area schedules.

1165

3.57 (NC) ControlAreaTimePoint root class

1167 Participation factor for a given point in time.

1168 Table 77 shows all attributes of ControlAreaTimePoint.

Table 77 – Attributes of SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
netInterchange	1..1	ActivePower	(NC) The specified positive net interchange into the control area, i.e. positive sign means flow into the area.
pTolerance	0..1	ActivePower	(NC) Active power net interchange tolerance. The attribute shall be a positive value or zero.

1170

1171 Table 78 shows all association ends of ControlAreaTimePoint with other classes.

1172
1173**Table 78 – Association ends of
SteadyStateHypothesisScheduleProfile::ControlAreaTimePoint with other classes**

mult from	name	mult to	type	description
1..*	ControlAreaSchedule	1..1	ControlAreaSchedule	(NC) The control area schedule that has this time point.

1174

3.58 (NC) SynchronousMachineSchedule1176 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1177 Schedule for synchronous machine.

1178 Table 79 shows all attributes of SynchronousMachineSchedule.

1179
1180**Table 79 – Attributes of
SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1181

1182 Table 80 shows all association ends of SynchronousMachineSchedule with other classes.

1183
1184
1185**Table 80 – Association ends of
SteadyStateHypothesisScheduleProfile::SynchronousMachineSchedule with other
classes**

mult from	name	mult to	type	description
0..*	SynchronousMachine	0..1	SynchronousMachine	(NC) Synchronous machine which has synchronous machine schedules.

1186

3.59 (NC) SynchronousMachineTimePoint root class

1188 Synchronous machine values for a given point in time.

1189 Table 81 shows all attributes of SynchronousMachineTimePoint.

1190
1191**Table 81 – Attributes of
SteadyStateHypothesisScheduleProfile::SynchronousMachineTimePoint**

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
operatingMode	1..1	SynchronousMachineOperatingMode	(NC) Current mode of operation.
referencePriority	1..1	Integer	(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.

1192

1193 Table 82 shows all association ends of SynchronousMachineTimePoint with other classes.

1194
1195
1196

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

mult from	name	mult to	type	description
1..*	SynchronousMachineSchedule	1..1	SynchronousMachineSchedule	(NC) The synchronous machine schedule that has this time point.

1197

1198 3.60 (NC) AsynchronousMachineSchedule

1199 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1200 Schedule for asynchronous machine.

1201 Table 83 shows all attributes of AsynchronousMachineSchedule.

1202

1203

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

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1204

1205 Table 84 shows all association ends of AsynchronousMachineSchedule with other classes.

1206

1207

1208

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

mult from	name	mult to	type	description
0..*	AsynchronousMachine	0..1	AsynchronousMachine	(NC) Asynchronous machine which has asynchronous machine schedules.

1209

1210 3.61 (NC) AsynchronousMachineTimePoint root class

1211 Asynchronous machine values for a given point in time.

1212 Table 85 shows all attributes of AsynchronousMachineTimePoint.

1213

1214

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

name	mult	type	description
asynchronousMachineType	1..1	AsynchronousMachineKind	(NC) Indicates the type of Asynchronous Machine (motor or generator).
atTime	1..1	DateTime	(NC) The time the data is valid for.

1215

1216

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

1217
1218
1219

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

mult from	name	mult to	type	description
1..*	AsynchronousMachineSchedule	1..1	AsynchronousMachineSchedule	(NC) The asynchronous machine schedule that has this time point.

1220

1221 3.62 (NC) ExternalNetworkInjectionSchedule

1222 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1223 Schedule for external network injection.

1224 Table 87 shows all attributes of ExternalNetworkInjectionSchedule.

1225

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

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1227

1228 Table 88 shows all association ends of ExternalNetworkInjectionSchedule with other classes.

1229

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

mult from	name	mult to	type	description
0..*	ExternalNetworkInjection	0..1	ExternalNetworkInjection	(NC) External Network Injection which has External Network Injection schedules.

1232

1233 3.63 (NC) ExternalNetworkInjectionTimePoint root class

1234 External network injection values for a given point in time.

1235 Table 89 shows all attributes of ExternalNetworkInjectionTimePoint.

1236

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

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
referencePriority	1..1	Integer	(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.

1237

name	mult	type	description
p	1..1	ActivePower	(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	ReactivePower	(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.

1238

1239

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

1240

Table 90 – Association ends of

1241

SteadyStateHypothesisScheduleProfile::ExternalNetworkInjectionTimePoint with other classes

1242

mult from	name	mult to	type	description
1..*	ExternalNetworkInjectionSchedule	1..1	ExternalNetworkInjectionSchedule	(NC) The external network injection schedule that has this time point.

1243

3.64 (NC) BatteryUnitSchedule

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

Schedule for battery unit.

Table 91 shows all attributes of BatteryUnitSchedule.

1248

Table 91 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1249

1250

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

1251

Table 92 – Association ends of

1252

SteadyStateHypothesisScheduleProfile::BatteryUnitSchedule with other classes

mult from	name	mult to	type	description
0..*	BatteryUnit	0..1	BatteryUnit	(NC) Battery unit which has battery unit schedules.

1253

3.65 (NC) BatteryUnitTimePoint root class

Battery unit values for a given point in time.

Table 93 shows all attributes of BatteryUnitTimePoint.

1256

1257 **Table 93 – Attributes of SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint**

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

1258

1259 Table 94 shows all association ends of BatteryUnitTimePoint with other classes.

1260

1261 **Table 94 – Association ends of SteadyStateHypothesisScheduleProfile::BatteryUnitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	BatteryUnitSchedule	1..1	BatteryUnitSchedule	(NC) The battery unit schedule that has this time point.

1262

1263 **3.66 (abstract) BatteryUnit root class**

1264 An electrochemical energy storage device.

1265 **3.67 (NC) VoltageLimitSchedule**1266 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1267 Schedule for voltage limit.

1268 Table 95 shows all attributes of VoltageLimitSchedule.

1269 **Table 95 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1270

1271 Table 96 shows all association ends of VoltageLimitSchedule with other classes.

1272

1273 **Table 96 – Association ends of SteadyStateHypothesisScheduleProfile::VoltageLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	VoltageLimit	0..1	VoltageLimit	(NC) Voltage limit which has voltage limit schedules.

1274

1275 **3.68 (abstract) VoltageLimit root class**

1276 Operational limit applied to voltage.

1277 The use of operational VoltageLimit is preferred instead of limits defined at VoltageLevel. The operational VoltageLimits are used, if present.

1278

1279 **3.69 (NC) VoltageLimitTimePoint root class**

1280 Voltage limit values for a given point in time.

1281 Table 97 shows all attributes of VoltageLimitTimePoint.

1282 **Table 97 – Attributes of SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint**

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
value	1..1	Voltage	(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.

1283

1284 Table 98 shows all association ends of VoltageLimitTimePoint with other classes.

1285 **Table 98 – Association ends of**
1286 **SteadyStateHypothesisScheduleProfile::VoltageLimitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	VoltageLimitSchedule	1..1	VoltageLimitSchedule	(NC) The voltage limit schedule that has this time point.

1287

1288 **3.70 (NC) ActivePowerLimitSchedule**1289 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1290 Schedule for active power limit.

1291 Table 99 shows all attributes of ActivePowerLimitSchedule.

1292 **Table 99 – Attributes of**
1293 **SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1294

1295 Table 100 shows all association ends of ActivePowerLimitSchedule with other classes.

1296 **Table 100 – Association ends of**
1297 **SteadyStateHypothesisScheduleProfile::ActivePowerLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	ActivePowerLimit	0..1	ActivePowerLimit	(NC) Active power limit which has active power limit schedules.

1298

1299 **3.71 (abstract) ActivePowerLimit root class**

1300 Limit on active power flow.

1301 **3.72 (NC) ActivePowerLimitTimePoint root class**

1302 Active power limit for a given point in time.

1303 Table 101 shows all attributes of ActivePowerLimitTimePoint.

1304

1305

**Table 101 – Attributes of
SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint**

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

1306

1307

Table 102 shows all association ends of ActivePowerLimitTimePoint with other classes.

1308

1309

**Table 102 – Association ends of
SteadyStateHypothesisScheduleProfile::ActivePowerLimitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	ActivePowerLimitSchedule	1..1	ActivePowerLimitSchedule	(NC) The active power limit schedule that has this time point.

1310

1311 **3.73 (NC) ApparentPowerLimitSchedule**1312 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1313 Schedule for apparent power limit.

1314 Table 103 shows all attributes of ApparentPowerLimitSchedule.

1315

1316

**Table 103 – Attributes of
SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1317

1318

Table 104 shows all association ends of ApparentPowerLimitSchedule with other classes.

1319

1320

1321

**Table 104 – Association ends of
SteadyStateHypothesisScheduleProfile::ApparentPowerLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	ApparentPowerLimit	0..1	ApparentPowerLimit	(NC) Apparent power limit which has apparent power limit schedules.

1322

1323 **3.74 (abstract) ApparentPowerLimit root class**

1324 Apparent power limit.

1325 **3.75 (NC) ApparentPowerLimitTimePoint root class**

1326 Apparent power limit for a given point in time.

1327 Table 105 shows all attributes of ApparentPowerLimitTimePoint.

1328 **Table 105 – Attributes of**
1329 **SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint**

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

1330

1331 Table 106 shows all association ends of ApparentPowerLimitTimePoint with other classes.

1332 **Table 106 – Association ends of**
1333 **SteadyStateHypothesisScheduleProfile::ApparentPowerLimitTimePoint with other**
1334 **classes**

mult from	name	mult to	type	description
1..*	ApparentPowerLimitSchedule	1..1	ApparentPowerLimitSchedule	(NC) The apparent power limit schedule that has this time point.

1335

1336 **3.76 (NC) CurrentLimitSchedule**1337 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1338 Schedule for current limit.

1339 Table 107 shows all attributes of CurrentLimitSchedule.

1340 **Table 107 – Attributes of SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1341

1342 Table 108 shows all association ends of CurrentLimitSchedule with other classes.

1343 **Table 108 – Association ends of**
1344 **SteadyStateHypothesisScheduleProfile::CurrentLimitSchedule with other classes**

mult from	name	mult to	type	description
0..*	CurrentLimit	0..1	CurrentLimit	(NC) Current limit which has current limit schedules.

1345

1346 **3.77 (abstract) CurrentLimit root class**

1347 Operational limit on current.

1348 **3.78 (NC) CurrentLimitTimePoint root class**

1349 Current limit values for a given point in time.

1350 Table 109 shows all attributes of CurrentLimitTimePoint.

1351

1352

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

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

1353

1354

Table 110 shows all association ends of CurrentLimitTimePoint with other classes.

1355

1356

**Table 110 – Association ends of
SteadyStateHypothesisScheduleProfile::CurrentLimitTimePoint with other classes**

mult from	name	mult to	type	description
1..*	CurrentLimitSchedule	1..1	CurrentLimitSchedule	(NC) The current limit schedule that has this time point.

1357

1358 **3.79 (NC) ShuntCompensatorSchedule**1359 Inheritance path = [BaselrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1360 Schedule for shunt compensator.

1361 Table 111 shows all attributes of ShuntCompensatorSchedule.

1362

1363

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

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1364

1365

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

1366

1367

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

mult from	name	mult to	type	description
0..*	ShuntCompensator	0..1	ShuntCompensator	(NC) Shunt compensator which has shunt compensator schedules.

1368

1369 **3.80 (NC) ShuntCompensatorTimePoint root class**

1370 Shunt compensator values for a given point in time.

1371 Table 113 shows all attributes of ShuntCompensatorTimePoint.

1372
1373**Table 113 – Attributes of
SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint**

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
sections	1..1	Float	(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.

1374
1375
1376
1377
1378

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

**Table 114 – Association ends of
SteadyStateHypothesisScheduleProfile::ShuntCompensatorTimePoint with other
classes**

mult from	name	mult to	type	description
1..*	ShuntCompensatorSchedule	1..1	ShuntCompensatorSchedule	(NC) The shunt compensator schedule that has this time point.

1379

3.81 (abstract) ShuntCompensator root class

1381 A shunt capacitor or reactor or switchable bank of shunt capacitors or reactors. A section of a
1382 shunt compensator is an individual capacitor or reactor. A negative value for bPerSection
1383 indicates that the compensator is a reactor. ShuntCompensator is a single terminal device.
1384 Ground is implied.

3.82 (NC) StaticVarCompensatorSchedule

1386 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)

1387 Schedule for static var compensator.

1388 Table 115 shows all attributes of StaticVarCompensatorSchedule.

1389
1390**Table 115 – Attributes of
SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule**

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries

name	mult	type	description
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1391
1392 Table 116 shows all association ends of StaticVarCompensatorSchedule with other classes.

1393 **Table 116 – Association ends of**
1394 **SteadyStateHypothesisScheduleProfile::StaticVarCompensatorSchedule with other**
1395 **classes**

mult from	name	mult to	type	description
0..*	StaticVarCompensator	0..1	StaticVarCompensator	(NC) Static var compensator which has static var compensator schedules.

1396
1397 **3.83 (NC) StaticVarCompensatorTimePoint root class**

1398 Static var compensator values for a given point in time.
1399 Table 117 shows all attributes of StaticVarCompensatorTimePoint.

1400 **Table 117 – Attributes of**
1401 **SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint**

name	mult	type	description
atTime	0..1	DateTime	(NC) The time the data is valid for.
q	1..1	ReactivePower	(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.

1402
1403 Table 118 shows all association ends of StaticVarCompensatorTimePoint with other classes.

1404 **Table 118 – Association ends of**
1405 **SteadyStateHypothesisScheduleProfile::StaticVarCompensatorTimePoint with other**
1406 **classes**

mult from	name	mult to	type	description
1..*	StaticVarCompensatorSchedule	1..1	StaticVarCompensatorSchedule	(NC) The StaticVarCompensator schedule that has this time point.

1407
1408 **3.84 (abstract) StaticVarCompensator root class**

1409 A facility for providing variable and controllable shunt reactive power. The SVC typically
1410 consists of a stepdown transformer, filter, thyristor-controlled reactor, and thyristor-switched
1411 capacitor arms.

1412 The SVC may operate in fixed MVar output mode or in voltage control mode. When in voltage
1413 control mode, the output of the SVC will be proportional to the deviation of voltage at the
1414 controlled bus from the voltage setpoint. The SVC characteristic slope defines the proportion.
1415 If the voltage at the controlled bus is equal to the voltage setpoint, the SVC MVar output is zero.

1416 **3.85 (NC) GeneratingUnitSchedule**

1417 Inheritance path = [BaseIrregularTimeSeries](#) : [BaseTimeSeries](#) : [IdentifiedObject](#)
1418 Schedule for generating unit.

1419 Table 119 shows all attributes of GeneratingUnitSchedule.

1420

1421

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

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) inherited from: BaseTimeSeries
timeSeriesKind	0..1	BaseTimeSeriesKind	(NC) inherited from: BaseTimeSeries
generatedAtTime	0..1	DateTime	(NC) inherited from: BaseTimeSeries
percentile	0..1	Integer	(NC) inherited from: BaseTimeSeries
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	1..1	String	inherited from: IdentifiedObject

1422

1423

Table 120 shows all association ends of GeneratingUnitSchedule with other classes.

1424

1425

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

mult from	name	mult to	type	description
0..*	GeneratingUnit	0..1	GeneratingUnit	(NC) Generating unit which has generating unit schedules.

1426

1427 3.86 (NC) GeneratingUnitTimePoint root class

1428 Generating unit values for a given point in time.

1429 Table 121 shows all attributes of GeneratingUnitTimePoint.

1430

1431

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

name	mult	type	description
atTime	1..1	DateTime	(NC) The time the data is valid for.
normalPF	1..1	Float	(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.

1432

1433

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

1434

1435

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

mult from	name	mult to	type	description
1..*	GeneratingUnitSchedule	1..1	GeneratingUnitSchedule	The generating unit schedule that has this time point.

1436

1437 3.87 (abstract) GeneratingUnit root class

1438 A single or set of synchronous machines for converting mechanical power into alternating-
1439 current power. For example, individual machines within a set may be defined for scheduling
1440 purposes while a single control signal is derived for the set. In this case there would be a

1441 GeneratingUnit for each member of the set and an additional GeneratingUnit corresponding to
1442 the set.

1443 3.88 MonthDay primitive

1444 MonthDay format as "--mm-dd", which conforms with XSD data type gMonthDay.

1445 3.89 ActivePower datatype

1446 Product of RMS value of the voltage and the RMS value of the in-phase component of the
1447 current.

1448 Table 123 shows all attributes of ActivePower.

1449 **Table 123 – Attributes of SteadyStateHypothesisScheduleProfile::ActivePower**

name	mult	type	description
value	0..1	Float	
multiplier	0..1	UnitMultiplier	(const=M)
unit	0..1	UnitSymbol	(const=W)

1450

1451 3.90 Float primitive

1452 A floating point number. The range is unspecified and not limited.

1453 3.91 UnitMultiplier enumeration

1454 The unit multipliers defined for the CIM. When applied to unit symbols, the unit symbol is
1455 treated as a derived unit. Regardless of the contents of the unit symbol text, the unit symbol
1456 shall be treated as if it were a single-character unit symbol. Unit symbols should not contain
1457 multipliers, and it should be left to the multiplier to define the multiple for an entire data type.

1458 For example, if a unit symbol is "m2Pers" and the multiplier is "k", then the value is $k(m^{**2}/s)$,
1459 and the multiplier applies to the entire final value, not to any individual part of the value. This
1460 can be conceptualized by substituting a derived unit symbol for the unit type. If one imagines
1461 that the symbol "P" represents the derived unit "m2Pers", then applying the multiplier "k" can
1462 be conceptualized simply as "kP".

1463 For example, the SI unit for mass is "kg" and not "g". If the unit symbol is defined as "kg", then
1464 the multiplier is applied to "kg" as a whole and does not replace the "k" in front of the "g". In
1465 this case, the multiplier of "m" would be used with the unit symbol of "kg" to represent one gram.
1466 As a text string, this violates the instructions in IEC 80000-1. However, because the unit symbol
1467 in CIM is treated as a derived unit instead of as an SI unit, it makes more sense to conceptualize
1468 the "kg" as if it were replaced by one of the proposed replacements for the SI mass symbol. If
1469 one imagines that the "kg" were replaced by a symbol "P", then it is easier to conceptualize the
1470 multiplier "m" as creating the proper unit "mP", and not the forbidden unit "mkg".

1471 Table 124 shows all literals of UnitMultiplier.

1472 **Table 124 – 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} .

1473

1474 3.92 UnitSymbol enumeration

1475 The derived units defined for usage in the CIM. In some cases, the derived unit is equal to an
1476 SI unit. Whenever possible, the standard derived symbol is used instead of the formula for the
1477 derived unit. For example, the unit symbol Farad is defined as "F" instead of "CPerV". In cases
1478 where a standard symbol does not exist for a derived unit, the formula for the unit is used as
1479 the unit symbol. For example, density does not have a standard symbol and so it is represented

1480 as "kgPerm3". With the exception of the "kg", which is an SI unit, the unit symbols do not contain
1481 multipliers and therefore represent the base derived unit to which a multiplier can be applied as
1482 a whole.

1483 Every unit symbol is treated as an unparseable text as if it were a single-letter symbol. The
1484 meaning of each unit symbol is defined by the accompanying descriptive text and not by the
1485 text contents of the unit symbol.

1486 To allow the widest possible range of serializations without requiring special character handling,
1487 several substitutions are made which deviate from the format described in IEC 80000-1. The
1488 division symbol "/" is replaced by the letters "Per". Exponents are written in plain text after the
1489 unit as "m3" instead of being formatted as "m" with a superscript of 3 or introducing a symbol
1490 as in "m^3". The degree symbol "°" is replaced with the letters "deg". Any clarification of the
1491 meaning for a substitution is included in the description for the unit symbol.

1492 Non-SI units are included in list of unit symbols to allow sources of data to be correctly labelled
1493 with their non-SI units (for example, a GPS sensor that is reporting numbers that represent feet
1494 instead of meters). This allows software to use the unit symbol information correctly convert
1495 and scale the raw data of those sources into SI-based units.

1496 The integer values are used for harmonization with IEC 61850.

1497 Table 125 shows all literals of UnitSymbol.

1498 **Table 125 – 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 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.

1499

1500 3.93 ReactivePower datatype

1501 Product of RMS value of the voltage and the RMS value of the quadrature component of the
1502 current.

1503 Table 126 shows all attributes of ReactivePower.

1504 **Table 126 – Attributes of SteadyStateHypothesisScheduleProfile::ReactivePower**

name	mult	type	description
value	0..1	Float	

name	mult	type	description
unit	0..1	UnitSymbol	(const=VAr)
multiplier	0..1	UnitMultiplier	(const=M)

1505

1506 **3.94 Voltage datatype**

1507 Electrical voltage, can be both AC and DC.

1508 Table 127 shows all attributes of Voltage.

1509 **Table 127 – Attributes of SteadyStateHypothesisScheduleProfile::Voltage**

name	mult	type	description
value	0..1	Float	
multiplier	0..1	UnitMultiplier	(const=k)
unit	0..1	UnitSymbol	(const=V)

1510

1511 **3.95 DateTime primitive**

1512 Date and time as "yyyy-mm-ddThh:mm:ss.sss", which conforms with ISO 8601. UTC time zone
1513 is specified as "yyyy-mm-ddThh:mm:ss.sssZ". A local timezone relative UTC is specified as
1514 "yyyy-mm-ddThh:mm:ss.sss-hh:mm". The second component (shown here as "ss.sss") could
1515 have any number of digits in its fractional part to allow any kind of precision beyond seconds.

1516 **3.96 ApparentPower datatype**

1517 Product of the RMS value of the voltage and the RMS value of the current.

1518 Table 128 shows all attributes of ApparentPower.

1519 **Table 128 – Attributes of SteadyStateHypothesisScheduleProfile::ApparentPower**

name	mult	type	description
value	0..1	Float	
multiplier	0..1	UnitMultiplier	(const=M)
unit	0..1	UnitSymbol	(const=VA)

1520

1521 **3.97 AsynchronousMachineKind enumeration**

1522 Kind of Asynchronous Machine.

1523 Table 129 shows all literals of AsynchronousMachineKind.

1524 **Table 129 – Literals of**1525 **SteadyStateHypothesisScheduleProfile::AsynchronousMachineKind**

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

1526

1527 **3.98 (NC) DayOfWeekKind enumeration**

1528 The kind of day to be included in a regular schedule.

1529 Table 130 shows all literals of DayOfWeekKind.

1530 **Table 130 – Literals of SteadyStateHypothesisScheduleProfile::DayOfWeekKind**

literal	value	description
monday		Monday as the day of the week.

literal	value	description
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		Day of the week other than Sunday or Saturday.
weekend		Day of the week which is Sunday or Saturday.
all		All days of the week.
publicHoliday		Public holiday is an officially designated day when most workplaces and schools are closed.
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.

1531

1532 **3.99 Integer primitive**

1533 An integer number. The range is unspecified and not limited.

1534 **3.100 BatteryStateKind enumeration**

1535 The state of the battery unit.

1536 Table 131 shows all literals of BatteryStateKind.

1537 **Table 131 – 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.

1538

1539 **3.101 RealEnergy datatype**

1540 Real electrical energy.

1541 Table 132 shows all attributes of RealEnergy.

1542 **Table 132 – Attributes of SteadyStateHypothesisScheduleProfile::RealEnergy**

name	mult	type	description
multiplier	0..1	UnitMultiplier	(const=M)
unit	0..1	UnitSymbol	(const=Wh)
value	0..1	Float	

1543

1544 **3.102 CsOperatingModeKind enumeration**

1545 Operating mode for HVDC line operating as Current Source Converter.

1546 Table 133 shows all literals of CsOperatingModeKind.

1547 **Table 133 – 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.

1548

1549 **3.103 CsPpccControlKind enumeration**

1550 Active power control modes for HVDC line operating as Current Source Converter.

1551 Table 134 shows all literals of CsPpccControlKind.

1552 **Table 134 – 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 ACDCCConverter.targetPpcc.
dcVoltage		Control is DC voltage with target value provided by ACDCCConverter.targetUdc.
dcCurrent		Control is DC current with target value provided by CsConverter.targetIdc.

1553

1554 **3.104 AngleDegrees datatype**

1555 Measurement of angle in degrees.

1556 Table 135 shows all attributes of AngleDegrees.

1557 **Table 135 – Attributes of SteadyStateHypothesisScheduleProfile::AngleDegrees**

name	mult	type	description
value	0..1	Float	
unit	0..1	UnitSymbol	(const=deg)
multiplier	0..1	UnitMultiplier	(const=none)

1558

1559 **3.105 CurrentFlow datatype**

1560 Electrical current with sign convention: positive flow is out of the conducting equipment into the connectivity node. Can be both AC and DC.

1562 Table 136 shows all attributes of CurrentFlow.

1563 **Table 136 – Attributes of SteadyStateHypothesisScheduleProfile::CurrentFlow**

name	mult	type	description
value	0..1	Float	
multiplier	0..1	UnitMultiplier	(const=none)
unit	0..1	UnitSymbol	(const=A)

1564

1565 **3.106 Boolean primitive**

1566 A type with the value space "true" and "false".

1567 **3.107 (NC) PeakKind enumeration**

1568 Kind of time period with similar intensity.

1569 Table 137 shows all literals of PeakKind.

1570 **Table 137 – 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.

1571

1572 **3.108 (NC) EnergyScenarioKind enumeration**

1573 Kind of energy scenario.

1574 Table 138 shows all literals of EnergyScenarioKind.

1575 **Table 138 – Literals of SteadyStateHypothesisScheduleProfile::EnergyScenarioKind**

literal	value	description
consumption		Scenario focus on consumption of energy.
production		Scenario focus on production of energy.
storage		Scenario focus on storage of energy.
export		Scenario focus on export of energy.
import		Scenario focus on import of energy.

1576

1577 **3.109 String primitive**1578 A string consisting of a sequence of characters. The character encoding is UTF-8. The string
1579 length is unspecified and unlimited.1580 **3.110 Time primitive**1581 Time as "hh:mm:ss.sss", which conforms with ISO 8601. UTC time zone is specified as
1582 "hh:mm:ss.sssZ". A local timezone relative UTC is specified as "hh:mm:ss.sss±hh:mm". The
1583 second component (shown here as "ss.sss") could have any number of digits in its fractional
1584 part to allow any kind of precision beyond seconds.1585 **3.111 SynchronousMachineOperatingMode enumeration**

1586 Synchronous machine operating mode.

1587 Table 139 shows all literals of SynchronousMachineOperatingMode.

1588 **Table 139 – Literals of**
1589 **SteadyStateHypothesisScheduleProfile::SynchronousMachineOperatingMode**

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

1590

1591 **3.112 PU datatype**1592 Per Unit - a positive or negative value referred to a defined base. Values typically range from -
1593 10 to +10.

1594 Table 140 shows all attributes of PU.

1595 **Table 140 – Attributes of SteadyStateHypothesisScheduleProfile::PU**

name	mult	type	description
value	0..1	Float	

name	mult	type	description
unit	0..1	UnitSymbol	(const=none)
multiplier	0..1	UnitMultiplier	(const=none)

1596

1597 **3.113 Resistance datatype**

1598 Resistance (real part of impedance).

1599 Table 141 shows all attributes of Resistance.

1600 **Table 141 – Attributes of SteadyStateHypothesisScheduleProfile::Resistance**

name	mult	type	description
value	0..1	Float	
unit	0..1	UnitSymbol	(const=ohm)
multiplier	0..1	UnitMultiplier	(const=none)

1601

1602 **3.114 VsPpccControlKind enumeration**

1603 Types applicable to the control of real power and/or DC voltage by voltage source converter.

1604 Table 142 shows all literals of VsPpccControlKind.

1605 **Table 142 – Literals of SteadyStateHypothesisScheduleProfile::VsPpccControlKind**

literal	value	description
pPcc		Control is real power at point of common coupling. The target value is provided by <code>ACDCCConverter.targetPpcc</code> .
udc		Control is DC voltage with target value provided by <code>ACDCCConverter.targetUdc</code> .
pPccAndUdcDroop		Control is active power at point of common coupling and local DC voltage, with the droop. Target values are provided by <code>ACDCCConverter.targetPpcc</code> , <code>ACDCCConverter.targetUdc</code> and <code>VsConverter.droop</code> .
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 (real or virtual) node in the DC network. Targets are provided by <code>ACDCCConverter.targetPpcc</code> , <code>ACDCCConverter.targetUdc</code> , <code>VsConverter.droop</code> and <code>VsConverter.droopCompensation</code> .
pPccAndUdcDroopPilot		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>ACDCCConverter.targetPpcc</code> , <code>ACDCCConverter.targetUdc</code> and <code>VsConverter.droop</code> .
phasePcc		Control is phase at point of common coupling. Target is provided by <code>VsConverter.targetPhasePcc</code> .

1606

1607 **3.115 VsQpccControlKind enumeration**

1608 Kind of reactive power control at point of common coupling for a voltage source converter.

1609 Table 143 shows all literals of VsQpccControlKind.

1610 **Table 143 – 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).

1611

1612 **3.116 PerCent datatype**

1613 Percentage on a defined base. For example, specify as 100 to indicate at the defined base.

1614 Table 144 shows all attributes of PerCent.

1615 **Table 144 – Attributes of SteadyStateHypothesisScheduleProfile::PerCent**

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

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