



European Network of
Transmission System Operators
for Electricity

AVAILABILITY SCHEDULE PROFILE SPECIFICATION

2026-09-10

ICTC APPROVED
VERSION 2.4.0

Copyright notice:

Copyright © ENTSO-E. All Rights Reserved.

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

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

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

This document is maintained by the ENTSO-E CIM WG. Comments or remarks are to be provided at cim@entsoe.eu

NOTE CONCERNING WORDING USED IN THIS DOCUMENT

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

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

Revision History

Version	Date	Paragraph	Comments
0.1.0	2021-10-12		For CIM EG review
1.0.0	2022-02-16		For SOC approval
2.1.0	2022-07-21		For SOC approval
2.2.0	2023-03-24		For review
2.2.0	2023-04-20		For ICTC approval.
2.3.0-alpha	2024-02-17		For internal review.
2.3.0-beta	2024-03-20		For CIM WG review.
2.3.1-alpha	2024-09-07		For CIM WG review.
2.4.0	2026-08-05		For CIM WG review.
2.4.0	2026-09-10		ICTC approval.

34 CONTENTS

35	Copyright notice:.....	2
36	Revision History.....	3
37	CONTENTS	4
38	1 Introduction	8
39	2 Application profile specification	8
40	2.1 Version information	8
41	2.2 Constraints naming convention	8
42	2.3 Profile constraints	9
43	2.4 Metadata.....	12
44	2.4.1 Constraints	12
45	2.4.2 Reference metadata	13
46	3 Detailed Profile Specification	13
47	3.1 General.....	13
48	3.2 (NC) EventSchedule	14
49	3.3 (NC) EventTimePoint root class	14
50	3.4 (abstract,NC) GridStateAlterationCollection root class	15
51	3.5 (NC) AvailabilityEquipment	15
52	3.6 (NC) AvailabilityExceptionalLimit.....	15
53	3.7 (NC) AvailabilityGroup	16
54	3.8 (abstract,NC) AvailabilityPowerSystemFunction	16
55	3.9 (NC) AvailabilityRemedialActionScheme	17
56	3.10 (NC) OutageRequestVersion	18
57	3.11 (NC) OutageResponse root class	18
58	3.12 (NC) AvailabilityEnabled	19
59	3.13 (abstract,NC) AssessedElement root class.....	19
60	3.14 (abstract,NC) AvailabilityRemedialAction root class	20
61	3.15 (NC) AvailabilityPlan	20
62	3.16 (abstract,NC) SystemOperator root class	20
63	3.17 (abstract,NC) OutageCoordinator root class.....	20
64	3.18 (abstract,NC) RecordVersion root class	21
65	3.19 (NC) AvailabilityPlanVersion	21
66	3.20 (NC) AvailabilityPlanResponse root class	22
67	3.21 (NC) RecordStatusKind enumeration	23
68	3.22 (NC) RequestResponseKind enumeration	23
69	3.23 (NC) OutageRequest	24
70	3.24 (NC) AvailabilitySchedule.....	25
71	3.25 (NC) AvailabilityContainer	26
72	3.26 (abstract,NC) BaseIrregularTimeSeries	27
73	3.27 (abstract,NC) BaseTimeSeries	27
74	3.28 (abstract) Equipment root class.....	27
75	3.29 (abstract) EquipmentContainer root class.....	27
76	3.30 (abstract,NC) GridStateAlteration root class.....	27
77	3.31 (abstract) IdentifiedObject root class	28

78	3.32	(abstract) OperationalLimit root class	28
79	3.33	(abstract,NC) RemedialActionScheme root class	28
80	3.34	(NC) TimeSeriesInterpolationKind enumeration	28
81	3.35	(NC) AvailabilityFunctionKind enumeration	28
82	3.36	(NC) AvailabilityScheduleCauseKind enumeration	29
83	3.37	(NC) BaseTimeSeriesKind enumeration	29
84	3.38	UnitMultiplier enumeration	29
85	3.39	UnitSymbol enumeration	30
86	3.40	Seconds datatype	30
87	3.41	Boolean primitive	31
88	3.42	DateTime primitive	31
89	3.43	Duration primitive	31
90	3.44	Integer primitive	31
91	3.45	Float primitive	31
92	3.46	String primitive	31
93	3.47	(abstract,NC) OutagePlanningAgent root class	31
94	3.48	(NC) OutageRequestResource root class	31
95	3.49	(abstract) PowerSystemResource root class	32
96	3.50	(NC) OutageAvailability root class	32
97	3.51	(NC) OutageDependency	32
98	3.52	(NC) ReferenceValueKind enumeration	33
99	3.53	Time primitive	34
100	3.54	(abstract,NC) PeerTemporalDependency root class	34
101	3.55	(NC) DependencyConstraintKind enumeration	36
102		Annex A (informative): Sample data	37
103	A.1	General	37
104	A.2	Sample instance data	37
105			
106		List of figures	
107		Figure 1 – Class diagram AvailabilityScheduleProfile::AvailabilityPlan	13
108		Figure 2 – Class diagram AvailabilityScheduleProfile::OutageRequest	13
109		Figure 3 – Class diagram AvailabilityScheduleProfile::AvailabilitySchedule	14
110			
111		List of tables	
112		Table 1 – Attributes of AvailabilityScheduleProfile::EventSchedule	14
113		Table 2 – Attributes of AvailabilityScheduleProfile::EventTimePoint	14
114		Table 3 – Association ends of AvailabilityScheduleProfile::EventTimePoint with other	
115		classes	15
116		Table 4 – Attributes of AvailabilityScheduleProfile::AvailabilityEquipment	15
117		Table 5 – Association ends of AvailabilityScheduleProfile::AvailabilityEquipment with	
118		other classes	15
119		Table 6 – Attributes of AvailabilityScheduleProfile::AvailabilityExceptionalLimit	16

120	Table 7 – Association ends of AvailabilityScheduleProfile::AvailabilityExceptionalLimit	
121	with other classes	16
122	Table 8 – Attributes of AvailabilityScheduleProfile::AvailabilityGroup	16
123	Table 9 – Attributes of AvailabilityScheduleProfile::AvailabilityPowerSystemFunction	17
124	Table 10 – Association ends of	
125	AvailabilityScheduleProfile::AvailabilityPowerSystemFunction with other classes	17
126	Table 11 – Attributes of AvailabilityScheduleProfile::AvailabilityRemedialActionScheme	17
127	Table 12 – Association ends of	
128	AvailabilityScheduleProfile::AvailabilityRemedialActionScheme with other classes	17
129	Table 13 – Attributes of AvailabilityScheduleProfile::OutageRequestVersion	18
130	Table 14 – Association ends of AvailabilityScheduleProfile::OutageRequestVersion	
131	with other classes	18
132	Table 15 – Attributes of AvailabilityScheduleProfile::OutageResponse	18
133	Table 16 – Association ends of AvailabilityScheduleProfile::OutageResponse with	
134	other classes	19
135	Table 17 – Attributes of AvailabilityScheduleProfile::AvailabilityEnabled	19
136	Table 18 – Association ends of AvailabilityScheduleProfile::AvailabilityEnabled with	
137	other classes	19
138	Table 19 – Attributes of AvailabilityScheduleProfile::AvailabilityPlan	20
139	Table 20 – Association ends of AvailabilityScheduleProfile::AvailabilityPlan with other	
140	classes	20
141	Table 21 – Attributes of AvailabilityScheduleProfile::RecordVersion	21
142	Table 22 – Association ends of AvailabilityScheduleProfile::RecordVersion with other	
143	classes	21
144	Table 23 – Attributes of AvailabilityScheduleProfile::AvailabilityPlanVersion	21
145	Table 24 – Association ends of AvailabilityScheduleProfile::AvailabilityPlanVersion with	
146	other classes	22
147	Table 25 – Attributes of AvailabilityScheduleProfile::AvailabilityPlanResponse	22
148	Table 26 – Association ends of AvailabilityScheduleProfile::AvailabilityPlanResponse	
149	with other classes	22
150	Table 27 – Literals of AvailabilityScheduleProfile::RecordStatusKind	23
151	Table 28 – Literals of AvailabilityScheduleProfile::RequestResponseKind	23
152	Table 29 – Attributes of AvailabilityScheduleProfile::OutageRequest	24
153	Table 30 – Association ends of AvailabilityScheduleProfile::OutageRequest with other	
154	classes	25
155	Table 31 – Attributes of AvailabilityScheduleProfile::AvailabilitySchedule	25
156	Table 32 – Association ends of AvailabilityScheduleProfile::AvailabilitySchedule with	
157	other classes	26
158	Table 33 – Attributes of AvailabilityScheduleProfile::AvailabilityContainer	26
159	Table 34 – Association ends of AvailabilityScheduleProfile::AvailabilityContainer with	
160	other classes	27
161	Table 35 – Attributes of AvailabilityScheduleProfile::BaseIrregularTimeSeries	27
162	Table 36 – Attributes of AvailabilityScheduleProfile::BaseTimeSeries	27
163	Table 37 – Attributes of AvailabilityScheduleProfile::IdentifiedObject	28

164	Table 38 – Literals of AvailabilityScheduleProfile::TimeSeriesInterpolationKind	28
165	Table 39 – Literals of AvailabilityScheduleProfile::AvailabilityFunctionKind	29
166	Table 40 – Literals of AvailabilityScheduleProfile::AvailabilityScheduleCauseKind	29
167	Table 41 – Literals of AvailabilityScheduleProfile::BaseTimeSeriesKind	29
168	Table 42 – Literals of AvailabilityScheduleProfile::UnitMultiplier	30
169	Table 43 – Literals of AvailabilityScheduleProfile::UnitSymbol	30
170	Table 44 – Attributes of AvailabilityScheduleProfile::Seconds	30
171	Table 45 – Attributes of AvailabilityScheduleProfile::OutageRequestResource	31
172	Table 46 – Association ends of AvailabilityScheduleProfile::OutageRequestResource	
173	with other classes	32
174	Table 47 – Attributes of AvailabilityScheduleProfile::OutageAvailability	32
175	Table 48 – Association ends of AvailabilityScheduleProfile::OutageAvailability with	
176	other classes	32
177	Table 49 – Attributes of AvailabilityScheduleProfile::OutageDependency	33
178	Table 50 – Association ends of AvailabilityScheduleProfile::OutageDependency with	
179	other classes	33
180	Table 51 – Literals of AvailabilityScheduleProfile::ReferenceValueKind	33
181	Table 52 – Attributes of AvailabilityScheduleProfile::PeerTemporalDependency	34
182	Table 53 – Literals of AvailabilityScheduleProfile::DependencyConstraintKind	36
183		

184 1 Introduction

185 The availability schedule profile is a profile to exchange information on availability related to
 186 not only equipment, but also equipment containers, remedial action schemes, individual grid
 187 state alterations and collections of them, and operational limits. Availability schedules and
 188 functions are exchanged. A given (un)availability schedule provides information on status,
 189 cause and can include multiple equipment that is simultaneously scheduled for unavailability.
 190 The availability power system function gives the state change (e.g., inService, outOfService) of
 191 the relevant function (e.g., Line) for a single availability schedule. Only power system functions
 192 that are directly impacted are explicitly included. For example, the unavailability of a switch
 193 might cause a line to be unavailable. Only the switch is included in the schedule and not the
 194 line that becomes de-energized as a cause of the availability schedule for switch.

195 2 Application profile specification

196 2.1 Version information

197 The content is generated from UML model file CIM17-2_CGMES31v01_PROF-
 198 20v03_NC25v25_MS10v12_DES10v01.qea.

199 This edition is based on the IEC 61970 UML version “”, dated “”.

- 200 - Title:
- 201 - Keyword:
- 202 - Description:
- 203 - Version IRI:
- 204 - Version info:
- 205 - Prior version:
- 206 - Conforms to:
- 207 - Identifier:

208

209 2.2 Constraints naming convention

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

212 “{rule.Type}:{rule.Standard}:{rule.Profile}:{rule.Property}:{rule.Name}”

213 where

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

215 rule.Standard: the number of the standard e.g. 301 for 61970-301, 456 for 61970-456, 13 for
 216 61968-13. 61970-600 specific constraints refer to 600 although they are related to one or
 217 combination of the 61970-450 series profiles. For NC profiles, NC is used.

218 rule.Profile: the abbreviation of the profile, e.g. TP for Topology profile. If set to “ALL” the
 219 constraint is applicable to all IEC 61970-600 profiles.

220 rule.Property: for UML classes, the name of the class, for attributes and associations, the name
 221 of the class and attribute or association end, e.g. EnergyConsumer, IdentifiedObject.name, etc.
 222 If set to “NA” the property is not applicable to a specific UML element.

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

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

225 2.3 Profile constraints

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

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

- 231 • C:452:ALL:NA:datatypes

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

- 234 • R:452:ALL:NA:exchange

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

- 237 • R:452:ALL:NA:exchange1

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

- 243 • R:452:ALL:NA:exchange2

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

- 252 • R:452:ALL:NA:exchange3

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

- 256 • R:452:ALL:NA:exchange4

257 From the standpoint of the model import used by a data recipient, the document
258 describes a subset of the CIM that importing software shall be able to interpret in order
259 to import exported models. Data providers are free to exceed the minimum requirements
260 described herein as long as their resulting data files are compliant with the CIM Schema
261 and the conventions established in Clause 5. The document, therefore, describes
262 additional classes and class data that, although not required, exporters will, in all
263 likelihood, choose to include in their data files. The additional classes and data are
264 labelled as required (cardinality 1..1) or as optional (cardinality 0..1) to distinguish them
265 from their required counterparts. Please note, however, that data importers could

- 266 potentially receive data containing instances of any and all classes described by the
267 CIM Schema.
- 268 • R:452:ALL:NA:cardinality
- 269 The cardinality defined in the CIM model shall be followed, unless a more restrictive
270 cardinality is explicitly defined in this document. For instance, the cardinality on the
271 association between VoltageLevel and BaseVoltage indicates that a VoltageLevel shall
272 be associated with one and only one BaseVoltage, but a BaseVoltage can be associated
273 with zero to many VoltageLevels.
- 274 • R:452:ALL:NA:associations
- 275 Associations between classes referenced in this document and classes not referenced
276 here are not required regardless of cardinality.
- 277 • R:452:ALL:IdentifiedObject.name:rule
- 278 The attribute “name” inherited by many classes from the abstract class IdentifiedObject
279 is not required to be unique. It must be a human readable identifier without additional
280 embedded information that would need to be parsed. The attribute is used for purposes
281 such as User Interface and data exchange debugging. The MRID defined in the data
282 exchange format is the only unique and persistent identifier used for this data exchange.
283 The attribute IdentifiedObject.name is, however, always required for CoreEquipment
284 profile and Short Circuit profile.
- 285 • R:452:ALL:IdentifiedObject.description:rule
- 286 The attribute “description” inherited by many classes from the abstract class
287 IdentifiedObject must contain human readable text without additional embedded
288 information that would need to be parsed.
- 289 • R:452:ALL:NA:uniqueIdentifier
- 290 All IdentifiedObject-s shall have a persistent and globally unique identifier (Master
291 Resource Identifier - mRID).
- 292 • R:452:ALL:NA:unitMultiplier
- 293 For exchange of attributes defined using CIM Data Types (ActivePower, Susceptance,
294 etc.) a unit multiplier of 1 is used if the UnitMultiplier specified in this document is “none”.
- 295 • C:452:ALL:IdentifiedObject.name:stringLength
- 296 The string IdentifiedObject.name has a maximum of 128 characters.
- 297 • C:452:ALL:IdentifiedObject.description:stringLength
- 298 The string IdentifiedObject.description is maximum 256 characters.
- 299 • C:452:ALL:NA:float
- 300 An attribute that is defined as float (e.g. has a type Float or a type which is a Datatype
301 with .value attribute of type Float) shall support ISO/IEC 60559:2020 for floating-point
302 arithmetic using single precision floating point. A single precision float supports 7
303 significant digits where the significant digits are described as an integer, or a decimal
304 number with 6 decimal digits. Two float values are equal when the significant with 7
305 digits are identical, e.g. 1234567 is equal 1.234567E6 and so are 1.2345678 and
306 1.234567E0.

- R:NC:ALL:NA:serialization

The profiles are defined in the EnterpriseArchitect application and have multiple artifacts that describe them. The main artifacts are:

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

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

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

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

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

- for association

```
<cim:PowerTransformer rdf:about="_c328f787-bc17-47ad-a59f-6ba7133340d0">
```

```
  <nc:Equipment.Circuit rdf:resource="#_9ced16ac-d076-4ef9-a241-a998a579e77b"/>
```

```
</cim:PowerTransformer>
```

- for attribute

```
<cim:ACLineSegment rdf:about="_04f681aa-6999-4fb3-9775-acaa5eb7ceff">
```

```
  <cim:Equipment.inService>true</cim:Equipment.inService>
```

```
</cim:ACLineSegment>
```

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

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

```
<cim:Equipment rdf:about="_c328f787-bc17-47ad-a59f-6ba7133340d0">
```

```
  <nc:Equipment.Circuit rdf:resource="#_9ced16ac-d076-4ef9-a241-a998a579e77b"/>
```

```
</cim:Equipment>
```

- 350 • C:NC:AS:AvailabilityEnabled:associations
- 351 AvailabilityEnabled shall have an association with either GridStateAlteration or
- 352 AssessedElement.
- 353 • C:NC:AS:PeerTemporalDependency.finishToFinishLagKind:required
- 354 If PeerTemporalDependency.finishToFinishLag is present
- 355 PeerTemporalDependency.finishToFinishLagKind is required.
- 356 • C:NC:AS:PeerTemporalDependency.finishToStartLagKind:required
- 357 If PeerTemporalDependency.finishToStartLag is present
- 358 PeerTemporalDependency.finishToStartLagKind is required.
- 359 • C:NC:AS:PeerTemporalDependency.overlapKind:required
- 360 If PeerTemporalDependency.overlap is present PeerTemporalDependency.overlapKind
- 361 is required.
- 362 • C:NC:AS:PeerTemporalDependency.startToStartLagKind:required
- 363 If PeerTemporalDependency.startToStartLag is present
- 364 PeerTemporalDependency.startToStartLagKind is required.
- 365 • C:NC:AS:PeerTemporalDependency.constraintKind:exclusive
- 366 If PeerTemporalDependency.constraintKind is set to
- 367 DependencyConstraintKind.exclusive, the following attributes shall not be defined:
- 368 PeerTemporalDependency.overlap, PeerTemporalDependency.startToStartLag,
- 369 PeerTemporalDependency.finishToStartLag,
- 370 PeerTemporalDependency.finishToFinishLag.

371

372

373 2.4 Metadata

374 ENTSO-E agreed to extend the header and metadata definitions by IEC 61970-552 Ed2. This
 375 new header definitions rely on W3C recommendations which are used worldwide and are
 376 positively recognised by the European Commission. The new definitions of the header mainly
 377 use Provenance ontology (PROV-O), Time Ontology and Data Catalog Vocabulary (DCAT). The
 378 global new header applicable for this profile is included in the metadata and document header
 379 specification document.

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

384 2.4.1 Constraints

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

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

388 The prov:wasAttributedTo should normally be the "X" EIC code of the actor or their URI
 389 (prov:Agent).

390

2.4.2 Reference metadata

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

3 Detailed Profile Specification

3.1 General

This package contains the availability plan profile.

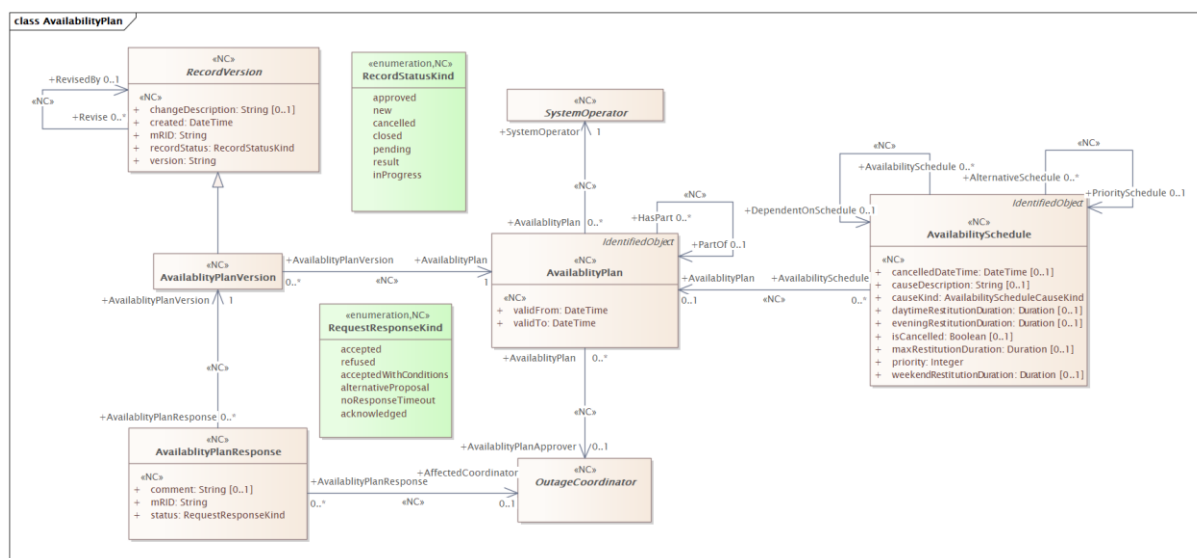


Figure 1 – Class diagram AvailabilityScheduleProfile::AvailabilityPlan

Figure 1: The diagram contains the main classes used for the availability plan.

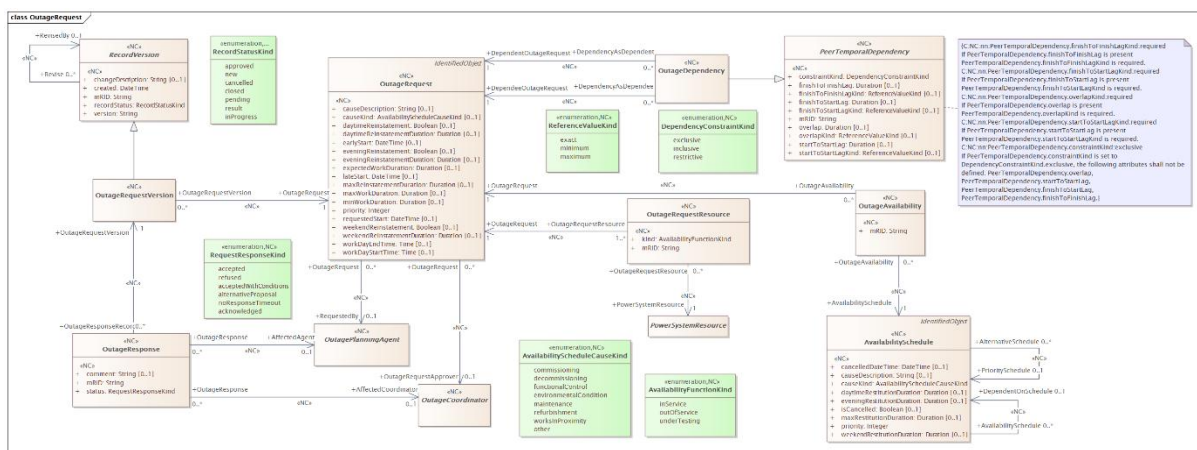


Figure 2 – Class diagram AvailabilityScheduleProfile::OutageRequest

Figure 2: The diagram contains the main classes used for the outage request.

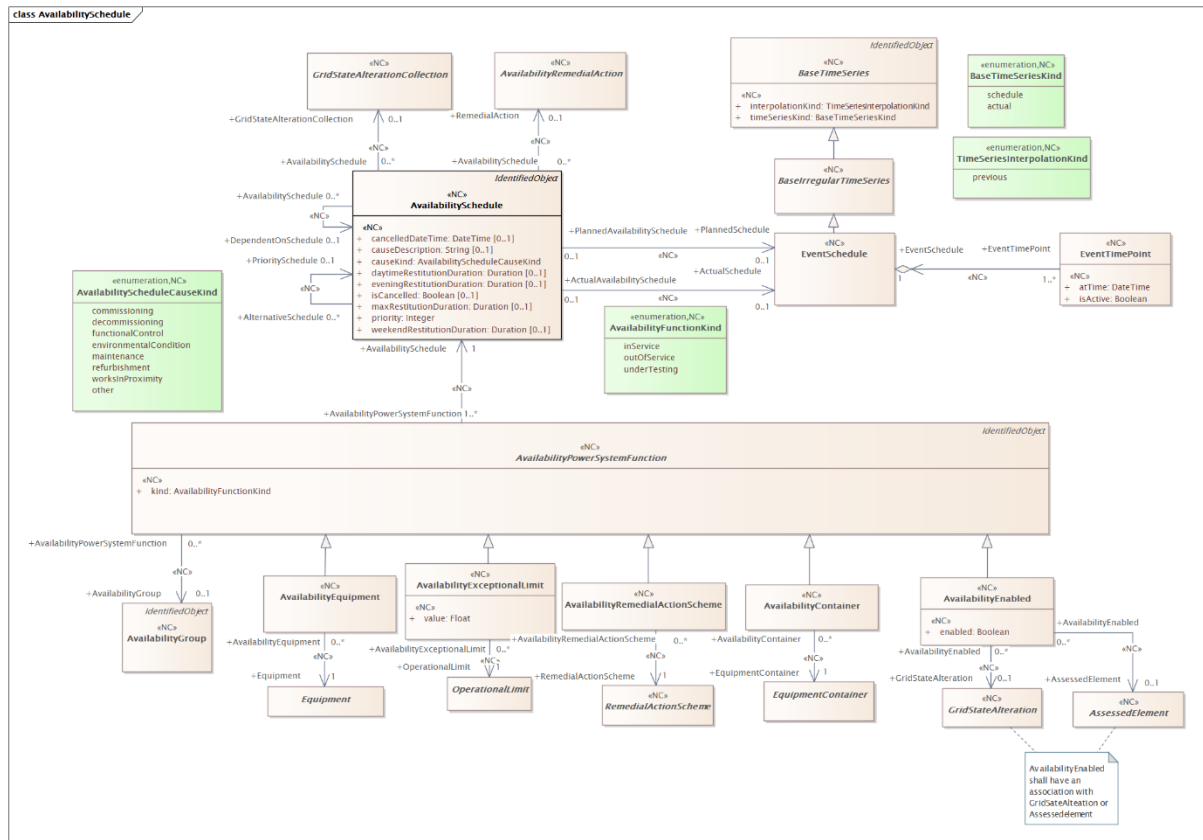


Figure 3 – Class diagram AvailabilityScheduleProfile::AvailabilitySchedule

Figure 3: The diagram contains the main classes used for availability schedule.

3.2 (NC) EventSchedule

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

Time series represent irregular event described by event points in time.

Table 1 shows all attributes of EventSchedule.

Table 1 – Attributes of AvailabilityScheduleProfile::EventSchedule

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

3.3 (NC) EventTimePoint root class

Event valid for a given point in time.

Table 2 shows all attributes of EventTimePoint.

Table 2 – Attributes of AvailabilityScheduleProfile::EventTimePoint

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

name	mult	type	description
isActive	1..1	Boolean	(NC) True, if the event is occurring (Active) at this time point. Otherwise false.

Table 3 shows all association ends of EventTimePoint with other classes.

Table 3 – Association ends of AvailabilityScheduleProfile::EventTimePoint with other classes

mult from	name	mult to	type	description
1..*	EventSchedule	1..1	EventSchedule	(NC) Time series the time point values belongs to.

3.4 (abstract,NC) GridStateAlterationCollection root class

A collection of grid state alterations.

3.5 (NC) AvailabilityEquipment

Inheritance path = [AvailabilityPowerSystemFunction](#) : [IdentifiedObject](#)

Availability equipment serves for associating an equipment with an availability schedule. For instance, putting in or out of service an ACLineSegment in combination with other availability functions with the same availability schedule.

Table 4 shows all attributes of AvailabilityEquipment.

Table 4 – Attributes of AvailabilityScheduleProfile::AvailabilityEquipment

name	mult	type	description
kind	1..1	AvailabilityFunctionKind	(NC) inherited from: AvailabilityPowerSystemFunction
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 5 shows all association ends of AvailabilityEquipment with other classes.

Table 5 – Association ends of AvailabilityScheduleProfile::AvailabilityEquipment with other classes

mult from	name	mult to	type	description
0..*	Equipment	1..1	Equipment	(NC) Equipment that is affected by the availability given by this availability equipment.
0..*	AvailabilityGroup	0..1	AvailabilityGroup	(NC) inherited from: AvailabilityPowerSystemFunction
1..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) inherited from: AvailabilityPowerSystemFunction

3.6 (NC) AvailabilityExceptionallimit

Inheritance path = [AvailabilityPowerSystemFunction](#) : [IdentifiedObject](#)

Availability exceptional limit serves for associating an operational limit restriction with an availability schedule. For instance, enabling or disabling the current limit on ACLineSegment terminal in combination with other availability functions with the same availability schedule or de-rating due to fault.

Table 6 shows all attributes of AvailabilityExceptionallimit.

Table 6 – Attributes of AvailabilityScheduleProfile::AvailabilityExceptionalLimit

name	mult	type	description
value	1..1	Float	(NC) Value for the referred operational limit.
kind	1..1	AvailabilityFunctionKind	(NC) inherited from: AvailabilityPowerSystemFunction
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 7 shows all association ends of AvailabilityExceptionalLimit with other classes.

Table 7 – Association ends of AvailabilityScheduleProfile::AvailabilityExceptionalLimit with other classes

mult from	name	mult to	type	description
0..*	OperationalLimit	1..1	OperationalLimit	(NC) Operational limit that is constrained by this availability exceptional limit.
0..*	AvailabilityGroup	0..1	AvailabilityGroup	(NC) inherited from: AvailabilityPowerSystemFunction
1..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) inherited from: AvailabilityPowerSystemFunction

3.7 (NC) AvailabilityGroup

Inheritance path = [IdentifiedObject](#)

Container to link relevant equipment that is affected by (un)availability schedule across availability coordinator (e.g. TSO-TSO, TSO-DSO or DSO-DSO).

Table 8 shows all attributes of AvailabilityGroup.

Table 8 – Attributes of AvailabilityScheduleProfile::AvailabilityGroup

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

3.8 (abstract,NC) AvailabilityPowerSystemFunction

Inheritance path = [IdentifiedObject](#)

Availability power system function describes the power system function that has a non-normal availability in the associated availability schedule. The availability of the function is needed as part of a power flow solution. This function is the cause and not the effect of the availability, if the effect can be calculated through power flow. For instance if only the step-up transformer for a generator is not available, the power flow will calculate that the generator is de-energized (outage). If both are tagged as not available it will not be possible to investigate remedial action for connecting the generator. It is expected that the power flow function is able to perform simple topology changes affected by a function taken out of service, e.g. open switches on both end of a ACLineSegment when the ACLineSegment is taken out of service. More complex changes, like change regulation set point, must be described in the linked GridStateAlterationCollection. Table 9 shows all attributes of AvailabilityPowerSystemFunction.

Table 9 – Attributes of AvailabilityScheduleProfile::AvailabilityPowerSystemFunction

name	mult	type	description
kind	1..1	AvailabilityFunctionKind	(NC) Kind of availability that affect the power system function.
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 10 shows all association ends of AvailabilityPowerSystemFunction with other classes.

**Table 10 – Association ends of
AvailabilityScheduleProfile::AvailabilityPowerSystemFunction with other classes**

mult from	name	mult to	type	description
0..*	AvailabilityGroup	0..1	AvailabilityGroup	(NC) Grouping for all availability power system functions (controlled by all relevant system operators) that have the same availability schedule.
1..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) Availability schedule for this availability power system function.

3.9 (NC) AvailabilityRemedialActionScheme

Inheritance path = [AvailabilityPowerSystemFunction](#) : [IdentifiedObject](#)

Availability remedial action scheme serves for associating a remedial action scheme with an availability schedule. For instance, taking in or out of service a SIPS / SPS due to communication issue, in combination with other availability functions with the same availability schedule.

Table 11 shows all attributes of AvailabilityRemedialActionScheme.

Table 11 – Attributes of AvailabilityScheduleProfile::AvailabilityRemedialActionScheme

name	mult	type	description
kind	1..1	AvailabilityFunctionKind	(NC) inherited from: AvailabilityPowerSystemFunction
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 12 shows all association ends of AvailabilityRemedialActionScheme with other classes.

**Table 12 – Association ends of
AvailabilityScheduleProfile::AvailabilityRemedialActionScheme with other classes**

mult from	name	mult to	type	description
0..*	RemedialActionScheme	1..1	RemedialActionScheme	(NC) Remedial action scheme that is affected by the availability given by this availability remedial action scheme.
0..*	AvailabilityGroup	0..1	AvailabilityGroup	(NC) inherited from: AvailabilityPowerSystemFunction
1..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) inherited from: AvailabilityPowerSystemFunction

3.10 (NC) OutageRequestVersion

Inheritance path = [RecordVersion](#)

Specialisation of RecordVersion that represents a specific version of an OutageRequest used in the outage coordination process.

An OutageRequestVersion contains the current proposed timing, scope and conditions of the outage and is the object against which outage dependencies, outage responses and resulting availability are evaluated.

Table 13 shows all attributes of OutageRequestVersion.

Table 13 – Attributes of AvailabilityScheduleProfile::OutageRequestVersion

name	mult	type	description
changeDescription	0..1	String	(NC) inherited from: RecordVersion
created	1..1	DateTime	(NC) inherited from: RecordVersion
mRID	1..1	String	(NC) inherited from: RecordVersion
recordStatus	1..1	RecordStatusKind	(NC) inherited from: RecordVersion
version	1..1	String	(NC) inherited from: RecordVersion

Table 14 shows all association ends of OutageRequestVersion with other classes.

Table 14 – Association ends of AvailabilityScheduleProfile::OutageRequestVersion with other classes

mult from	name	mult to	type	description
0..*	OutageRequest	1..1	OutageRequest	(NC) The OutageRequest for which this object represents a specific version used during coordination.
0..*	RevisedBy	0..1	RecordVersion	(NC) inherited from: RecordVersion

3.11 (NC) OutageResponse root class

Represents the formal response provided by an affected agent or the outage coordinator to a specific OutageRequestVersion.

Table 15 shows all attributes of OutageResponse.

Table 15 – Attributes of AvailabilityScheduleProfile::OutageResponse

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.
status	1..1	RequestResponseKind	(NC) Outcome of the response to the outage request version, indicating the decision or position of the responding party.
comment	0..1	String	(NC) Free-text explanation, justification or additional information provided with the response to the outage request version.

Table 16 shows all association ends of OutageResponse with other classes.

Table 16 – Association ends of AvailabilityScheduleProfile::OutageResponse with other classes

mult from	name	mult to	type	description
0..*	OutageRequestVersion	1..1	OutageRequestVersion	(NC) The version of the outage request to which this response applies.
0..*	AffectedCoordinator	0..1	OutageCoordinator	(NC) The system operator acting as outage coordinator who issued the response.
0..*	AffectedAgent	0..1	OutagePlanningAgent	(NC) The agent that issued the response.

3.12 (NC) AvailabilityEnabled

Inheritance path = [AvailabilityPowerSystemFunction](#) : [IdentifiedObject](#)

Availability enabled is enabling or disabling grid state alteration (e.g. tap position action) or assessed element that is related to the availability schedule. For instance, the cancellation of availability schedule can lead to changes in the assessed element. This is done by enabling one assessment and disabling another.

Table 17 shows all attributes of AvailabilityEnabled.

Table 17 – Attributes of AvailabilityScheduleProfile::AvailabilityEnabled

name	mult	type	description
enabled	1..1	Boolean	(NC) Instruction to enable or disable alteration and assessment.
kind	1..1	AvailabilityFunctionKind	(NC) inherited from: AvailabilityPowerSystemFunction
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 18 shows all association ends of AvailabilityEnabled with other classes.

Table 18 – Association ends of AvailabilityScheduleProfile::AvailabilityEnabled with other classes

mult from	name	mult to	type	description
0..*	AssessedElement	0..1	AssessedElement	(NC) Assessed element that is affected by the availability given by this availability enabling.
0..*	GridStateAlteration	0..1	GridStateAlteration	(NC) Grid state alteration that is affected by the availability given by this availability enabling.
0..*	AvailabilityGroup	0..1	AvailabilityGroup	(NC) inherited from: AvailabilityPowerSystemFunction
1..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) inherited from: AvailabilityPowerSystemFunction

3.13 (abstract,NC) AssessedElement root class

Assessed element is a network element for which the electrical state is evaluated in the regional or cross-regional process and which value is expected to fulfil regional rules function of the operational security limits.

The measurements and limits are as defined in the steady state hypothesis.

3.14 (abstract,NC) AvailabilityRemedialAction root class

Availability remedial action is a remedial action that cancels or reschedules an availability schedule.

3.15 (NC) AvailabilityPlan

Inheritance path = [IdentifiedObject](#)

A consolidated plan established by a System Operator that groups the AvailabilitySchedules applicable to a defined period and operational scope.

Note: An AvailabilityPlan provides the system-level view of resource availability used for outage coordination and operational planning. It may be divided into subordinate AvailabilityPlans and may be subject to approval by an Outage Coordinator.

Table 19 shows all attributes of AvailabilityPlan.

Table 19 – Attributes of AvailabilityScheduleProfile::AvailabilityPlan

name	mult	type	description
validTo	1..1	DateTime	(NC) The date and time the end of the availability plan.
validFrom	1..1	DateTime	(NC) The date and time the start of the availability plan.
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 20 shows all association ends of AvailabilityPlan with other classes.

Table 20 – Association ends of AvailabilityScheduleProfile::AvailabilityPlan with other classes

mult from	name	mult to	type	description
0..*	SystemOperator	1..1	SystemOperator	(NC) The System Operator responsible for establishing and maintaining the AvailabilityPlan.
0..*	AvailabilityPlanApprover	0..1	OutageCoordinator	(NC) The Outage Coordinator responsible for approving the AvailabilityPlan.
0..*	PartOf	0..1	AvailabilityPlan	(NC) The parent AvailabilityPlan of which this AvailabilityPlan forms a part.

3.16 (abstract,NC) SystemOperator root class

A power system operation post responsible for performing and coordinating the real-time and planning functions required to maintain the secure, reliable, and efficient operation of the power system within a defined operational scope.

Note: SystemOperator represents the execution of power system operational responsibilities in practice. While the operational post exists independently at the ecosystem level, SystemOperator identifies the specific agent fulfilling that responsibility at a given time. The operational scope may correspond to a control area, network, voltage domain, or other defined boundary.

3.17 (abstract,NC) OutageCoordinator root class

A post that coordinates the planned availability status of relevant power system equipment to meet the need by the asset owner or operator and the security of the power system.

3.18 (abstract,NC) RecordVersion root class

A RecordVersion captures the state of a record at a given point in time, including version identifiers, status, timestamps, and revision links to previous or subsequent versions, enabling controlled evolution and full traceability.

Table 21 shows all attributes of RecordVersion.

Table 21 – Attributes of AvailabilityScheduleProfile::RecordVersion

name	mult	type	description
changeDescription	0..1	String	(NC) Free-text description of the changes or rationale that distinguish this version from the previous one.
created	1..1	DateTime	(NC) Date and time when this record version was created.
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.
recordStatus	1..1	RecordStatusKind	(NC) Lifecycle status of this record version, indicating its current state in the business record.
version	1..1	String	(NC) Human-readable identifier of the version (for example “1”, “1.1”, “2025-01”), used for reference in business processes and documentation.

Table 22 shows all association ends of RecordVersion with other classes.

Table 22 – Association ends of AvailabilityScheduleProfile::RecordVersion with other classes

mult from	name	mult to	type	description
0..*	RevisedBy	0..1	RecordVersion	(NC) Reference to the next version that supersedes this record version.

3.19 (NC) AvailabilityPlanVersion

Inheritance path = [RecordVersion](#)

A specific version of an AvailabilityPlan used in the coordination and approval process.

Note: An AvailabilityPlanVersion preserves the state of the plan at a given point in time and is the subject of responses issued by affected Outage Coordinators. Changes to the AvailabilityPlan are represented by successive AvailabilityPlanVersions, thereby preserving the history and traceability of the coordination process.

Table 23 shows all attributes of AvailabilityPlanVersion.

Table 23 – Attributes of AvailabilityScheduleProfile::AvailabilityPlanVersion

name	mult	type	description
changeDescription	0..1	String	(NC) inherited from: RecordVersion
created	1..1	DateTime	(NC) inherited from: RecordVersion
mRID	1..1	String	(NC) inherited from: RecordVersion

name	mult	type	description
recordStatus	1..1	RecordStatusKind	(NC) inherited from: RecordVersion
version	1..1	String	(NC) inherited from: RecordVersion

Table 24 shows all association ends of AvailabilityPlanVersion with other classes.

Table 24 – Association ends of AvailabilityScheduleProfile::AvailabilityPlanVersion with other classes

mult from	name	mult to	type	description
0..*	AvailabilityPlan	1..1	AvailabilityPlan	(NC) The AvailabilityPlan represented by this version.
0..*	RevisedBy	0..1	RecordVersion	(NC) inherited from: RecordVersion

3.20 (NC) AvailabilityPlanResponse root class

A formal response provided by an affected Outage Coordinator to a specific AvailabilityPlanVersion.

Note: The response records the coordinator's position on the proposed plan, such as acceptance, refusal, acceptance with conditions, an alternative proposal, acknowledgement, or absence of a response within the applicable time limit.

Table 25 shows all attributes of AvailabilityPlanResponse.

Table 25 – Attributes of AvailabilityScheduleProfile::AvailabilityPlanResponse

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.
status	1..1	RequestResponseKind	(NC) Outcome of the response to the outage request version, indicating the decision or position of the responding party.
comment	0..1	String	(NC) Free-text explanation, justification or additional information provided with the response to the outage request version.

Table 26 shows all association ends of AvailabilityPlanResponse with other classes.

Table 26 – Association ends of AvailabilityScheduleProfile::AvailabilityPlanResponse with other classes

mult from	name	mult to	type	description
0..*	AffectedCoordinator	0..1	OutageCoordinator	(NC) The affected Outage Coordinator that provides the response to the AvailabilityPlanVersion.
0..*	AvailabilityPlanVersion	1..1	AvailabilityPlanVersion	(NC) The version of the AvailabilityPlan to which this response applies.

3.21 (NC) RecordStatusKind enumeration

Enumeration indicating the lifecycle status of a business record.

Table 27 shows all literals of RecordStatusKind.

Table 27 – Literals of AvailabilityScheduleProfile::RecordStatusKind

literal	value	description
approved		Indicates that the record version has been reviewed and formally accepted by the responsible entity.
new		Indicates that the record version has been created but not yet submitted for review or coordination. It represents an initial draft state.
cancelled		Indicates that the record version has been withdrawn and is no longer valid for coordination. A cancelled record is retained only for audit and traceability.
closed		Indicates that the record version has completed its lifecycle and no further action is required. The record remains available for reference but is not active in any process.
pending		Indicates that the record version has been submitted and is awaiting responses or decisions from assigned agents.
result		Indicates that the record version represents a final outcome or result of any coordination process, typically summarising the consolidated decision or approved plan.
inProgress		Indicates that the record version is actively being processed, updated, or evaluated as part of any coordination workflow.

3.22 (NC) RequestResponseKind enumeration

Classification of the response to a request.

Table 28 shows all literals of RequestResponseKind.

Table 28 – Literals of AvailabilityScheduleProfile::RequestResponseKind

literal	value	description
accepted		Indicates that the responding agent agrees and accept the request version as submitted.
refused		Indicates that the responding agent does not accept the request version due to operational, technical, or regulatory constraints.
acceptedWithConditions		Indicates that the responding agent can accept the request version only if specific conditions, modifications, or mitigations are met.
alternativeProposal		Indicates that the responding agent proposes alternative timing, duration, scope, or other adjustments to the request version.
noResponseTimeout		Indicates that the responding agent did not provide a response within the required timeframe, and the coordination process proceeded using a timeout rule.
acknowledged		Indicates that the responding agent has received and reviewed the request version, but has not yet provided an approval or objection.

612 **3.23 (NC) OutageRequest**613 Inheritance path = [IdentifiedObject](#)

614 Represents the initial business intent to take one or more resources out of service during a specified period, as submitted by an Outage Planning Agent.

616 Table 29 shows all attributes of OutageRequest.

617 **Table 29 – Attributes of AvailabilityScheduleProfile::OutageRequest**

name	mult	type	description
causeDescription	0..1	String	(NC) A cause description for a cause kind. In case of CauseKind equals other, description or a reference of the cause of the (un)availability schedule.
causeKind	0..1	AvailabilityScheduleCauseKind	(NC) Kind of cause for the availability schedule.
daytimeReinstatement	0..1	Boolean	(NC) True, when no work is planned for the weekend or public holidays and the equipment can be reinstated.
daytimeReinstatementDuration	0..1	Duration	(NC) The time required to take the out-of-service equipment back into service during daytime. This includes the start-up time for generating units.
earlyStart	0..1	DateTime	(NC) Earliest possible start time/date of the work.
workDayEndTime	0..1	Time	(NC) The time in a normal work day that equipment can be reinstated.
workDayStartTime	0..1	Time	(NC) The time in a normal work day that equipment would be taken out of service in case of reinstatement later in the working day.
eveningReinstatement	0..1	Boolean	(NC) True, when no work is planned for the weekend or public holidays and the equipment can be reinstated.
eveningReinstatementDuration	0..1	Duration	(NC) The time required to take the out-of-service equipment back into service after office hours. This includes the start-up time for generating units.
expectedWorkDuration	0..1	Duration	(NC) The expected planned work duration.
lateStart	0..1	DateTime	(NC) The latest time/date the work can start.
maxReinstatementDuration	0..1	Duration	(NC) The maximum time required to take the out-of-service equipment back into service after office hours. This includes the start-up time for generating units.
maxWorkDuration	0..1	Duration	(NC) The maximum work duration taken into account risk mitigation with less resources available. If like that assessment is done on this duration to avoid re-planning on a later stage.
minWorkDuration	0..1	Duration	(NC) The minimum work duration that can be achieved when extra resources are allocated.
priority	1..1	Integer	(NC) 0 means ignore priority. 1 means the highest priority, 2 is the second highest priority.
requestedStart	0..1	DateTime	(NC) Requested start time/date.
weekendReinstatement	0..1	Boolean	(NC) True, when no work is planned for the weekend or public holidays and the equipment can be reinstated.
weekendReinstatementDuration	0..1	Duration	(NC) The time required to take the out-of-service equipment back into service in the weekend or during bank holidays. This includes the start-up time for generating units.

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

Table 30 shows all association ends of OutageRequest with other classes.

Table 30 – Association ends of AvailabilityScheduleProfile::OutageRequest with other classes

mult from	name	mult to	type	description
0..*	OutageRequestApprover	0..1	OutageCoordinator	(NC) The coordinator responsible for approving the final version of the outage request.
0..*	RequestedBy	0..1	OutagePlanningAgent	(NC) The agent responsible for creating and submitting the outage request.

3.24 (NC) AvailabilitySchedule

Inheritance path = [IdentifiedObject](#)

A given (un)availability schedule with a given status and cause that include multiple equipment that need to follow the same scheduling periods.

Table 31 shows all attributes of AvailabilitySchedule.

Table 31 – Attributes of AvailabilityScheduleProfile::AvailabilitySchedule

name	mult	type	description
cancelledDateTime	0..1	DateTime	(NC) The date and time the (un)availability schedule were cancelled .
causeDescription	0..1	String	(NC) A cause description for a cause kind. In case of CauseKind equals other, description or a reference of the cause of the (un)availability schedule.
causeKind	1..1	AvailabilityScheduleCauseKind	(NC) Kind of cause for the availability schedule.
maxRestitutionDuration	0..1	Duration	(NC) The maximum time required to take the out-of-service equipment back into service. This includes the start-up time for generating units.
priority	1..1	Integer	(NC) Value 0 means ignore priority. 1 means the highest priority, 2 is the second highest priority.
daytimeRestitutionDuration	0..1	Duration	(NC) The time required to take the out-of-service equipment back into service during daytime. This includes the start-up time for generating units.
eveningRestitutionDuration	0..1	Duration	(NC) The time required to take the out-of-service equipment back into service after office hours. This includes the start-up time for generating units.
weekendRestitutionDuration	0..1	Duration	(NC) The time required to take the out-of-service equipment back into service in the weekend or during bank holidays. This includes the start-up time for generating units.
isCancelled	0..1	Boolean	(NC) Defines the cancelling of the availability schedule. True means that is cancelling, False means that it is not cancelling.
description	0..1	String	inherited from: IdentifiedObject

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

Table 32 shows all association ends of AvailabilitySchedule with other classes.

Table 32 – Association ends of AvailabilityScheduleProfile::AvailabilitySchedule with other classes

mult from	name	mult to	type	description
0..*	GridStateAlterationCollection	0..1	GridStateAlterationCollection	(NC) The grid state alteration collection that has this availability schedule.
0..*	RemedialAction	0..1	AvailabilityRemedialAction	(NC) Remedial action that is cancelling this availability schedule.
0..*	DependentOnSchedule	0..1	AvailabilitySchedule	(NC) (Un)availability schedule requested by one operator may require another operator to request their (un)availability schedule. This association is linking the schedules so that the dependency is clear.
0..1	ActualSchedule	0..1	EventSchedule	(NC) Actual schedule that relates to this availability schedule; used for ex-post reporting and analysis (e.g., to compare planned vs. actual).
0..1	PlannedSchedule	0..1	EventSchedule	(NC) Planned schedule that relates to this availability schedule used for planning availability (e.g., to compare planned vs. actual).
0..*	AvailabilityPlan	0..1	AvailabilityPlan	(NC) The AvailabilityPlan of which this AvailabilitySchedule is a component.
0..*	PrioritySchedule	0..1	AvailabilitySchedule	(NC) Priority schedule. This is the schedule that has the highest priority and the only valid if not cancelled.

3.25 (NC) AvailabilityContainer

Inheritance path = [AvailabilityPowerSystemFunction](#) : [IdentifiedObject](#)

Availability container serves for associating an equipment container with an availability schedule. For instance, putting in or out of service all the equipment inside a Line or a Bay in combination with other availability functions with the same availability schedule.

Table 33 shows all attributes of AvailabilityContainer.

Table 33 – Attributes of AvailabilityScheduleProfile::AvailabilityContainer

name	mult	type	description
kind	1..1	AvailabilityFunctionKind	(NC) inherited from: AvailabilityPowerSystemFunction
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

Table 34 shows all association ends of AvailabilityContainer with other classes.

Table 34 – Association ends of AvailabilityScheduleProfile::AvailabilityContainer with other classes

mult from	name	mult to	type	description
0..*	EquipmentContainer	1..1	EquipmentContainer	(NC) Equipment container that is affected by the availability given by this availability container.
0..*	AvailabilityGroup	0..1	AvailabilityGroup	(NC) inherited from: AvailabilityPowerSystemFunction
1..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) inherited from: AvailabilityPowerSystemFunction

3.26 (abstract,NC) BaselrregularTimeSeriesInheritance path = [BaseTimeSeries](#) : [IdentifiedObject](#)

Time series that has irregular points in time.

Table 35 shows all attributes of BaselrregularTimeSeries.

Table 35 – Attributes of AvailabilityScheduleProfile::BaselrregularTimeSeries

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

3.27 (abstract,NC) BaseTimeSeriesInheritance path = [IdentifiedObject](#)

Time series of values at points in time.

Table 36 shows all attributes of BaseTimeSeries.

Table 36 – Attributes of AvailabilityScheduleProfile::BaseTimeSeries

name	mult	type	description
interpolationKind	1..1	TimeSeriesInterpolationKind	(NC) Kind of interpolation done between time point.
timeSeriesKind	1..1	BaseTimeSeriesKind	(NC) Kind of base time series.
description	0..1	String	inherited from: IdentifiedObject
mRID	1..1	String	inherited from: IdentifiedObject
name	0..1	String	inherited from: IdentifiedObject

3.28 (abstract) Equipment root class

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

3.29 (abstract) EquipmentContainer root class

A modelling construct to provide a root class for containing equipment.

3.30 (abstract,NC) GridStateAlteration root class

Grid state alteration is a change of values describing state (operating point) of one element in the grid model compared to the base case.

3.31 (abstract) IdentifiedObject root class

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

Table 37 shows all attributes of IdentifiedObject.

Table 37 – Attributes of AvailabilityScheduleProfile::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. 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	0..1	String	The name is any free human readable and possibly non unique text naming the object.

3.32 (abstract) OperationalLimit root class

A value and normal value associated with a specific kind of limit.

The sub class value and normalValue attributes vary inversely to the associated OperationalLimitType.acceptableDuration (acceptableDuration for short).

If a particular piece of equipment has multiple operational limits of the same kind (apparent power, current, etc.), the limit with the greatest acceptableDuration shall have the smallest limit value and the limit with the smallest acceptableDuration shall have the largest limit value. Note: A large current can only be allowed to flow through a piece of equipment for a short duration without causing damage, but a lesser current can be allowed to flow for a longer duration.

3.33 (abstract,NC) RemedialActionScheme root class

Remedial Action Scheme (RAS), Special Protection Schemes (SPS), System Protection Schemes (SPS) or System Integrity Protection Schemes (SIPS).

A Remedial Action Scheme consists of one or more stages that can trigger and execute a protection action.

3.34 (NC) TimeSeriesInterpolationKind enumeration

Kinds of interpolation of values between two time point.

Table 38 shows all literals of TimeSeriesInterpolationKind.

Table 38 – Literals of AvailabilityScheduleProfile::TimeSeriesInterpolationKind

literal	value	description
previous		The value between two time points is set to previous value.

3.35 (NC) AvailabilityFunctionKind enumeration

Kind of availability that is affecting the function.

Table 39 shows all literals of AvailabilityFunctionKind.

Table 39 – Literals of AvailabilityScheduleProfile::AvailabilityFunctionKind

literal	value	description
inService		Function is in service.
outOfService		Function is out-of-service.
underTesting		Function is under testing and need to expect unscheduled availability.

3.36 (NC) AvailabilityScheduleCauseKind enumeration

The kinds of cause of the (un)availability schedule.

Table 40 shows all literals of AvailabilityScheduleCauseKind.

Table 40 – Literals of AvailabilityScheduleProfile::AvailabilityScheduleCauseKind

literal	value	description
commissioning		The cause is due to a commissioning.
decommissioning		The cause is due to a decommissioning.
functionalControl		The cause is due to a functional control (in & out).
environmentalCondition		The cause is due to an environmental condition. This can lead to exceptional margin and limits.
maintenance		The cause is due to a maintenance.
refurbishment		The cause is due to a refurbishment, either upgrade or downgrade.
worksInProximity		The cause is due to a works in proximity.
other		The cause is of other kind.

3.37 (NC) BaseTimeSeriesKind enumeration

Kind of time series.

Table 41 shows all literals of BaseTimeSeriesKind.

Table 41 – Literals of AvailabilityScheduleProfile::BaseTimeSeriesKind

literal	value	description
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.

3.38 UnitMultiplier enumeration

The unit multipliers defined for the CIM. When applied to unit symbols, the unit symbol is treated as a derived unit. Regardless of the contents of the unit symbol text, the unit symbol shall be treated as if it were a single-character unit symbol. Unit symbols should not contain multipliers, and it should be left to the multiplier to define the multiple for an entire data type. For example, if a unit symbol is "m2Pers" and the multiplier is "k", then the value is k(m**2/s), and the multiplier applies to the entire final value, not to any individual part of the value. This can be conceptualized by substituting a derived unit symbol for the unit type. If one imagines

that the symbol "P" represents the derived unit "m2Pers", then applying the multiplier "k" can be conceptualized simply as "kP".

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

Table 42 shows all literals of UnitMultiplier.

Table 42 – Literals of AvailabilityScheduleProfile::UnitMultiplier

literal	value	description
none		No multiplier or equivalently multiply by 1.

3.39 UnitSymbol enumeration

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

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

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

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

The integer values are used for harmonization with IEC 61850.

Table 43 shows all literals of UnitSymbol.

Table 43 – Literals of AvailabilityScheduleProfile::UnitSymbol

literal	value	description
s		Time in seconds.

3.40 Seconds datatype

Time, in seconds.

Table 44 shows all attributes of Seconds.

Table 44 – Attributes of AvailabilityScheduleProfile::Seconds

name	mult	type	description
value	0..1	Float	Time, in seconds

name	mult	type	description
unit	0..1	UnitSymbol	
multiplier	0..1	UnitMultiplier	

3.41 Boolean primitive

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

3.42 DateTime primitive

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

3.43 Duration primitive

Duration as "PnYnMnDTnHnMnS" which conforms to ISO 8601, where nY expresses a number of years, nM a number of months, nD a number of days. The letter T separates the date expression from the time expression and, after it, nH identifies a number of hours, nM a number of minutes and nS a number of seconds. The number of seconds could be expressed as a decimal number, but all other numbers are integers.

3.44 Integer primitive

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

3.45 Float primitive

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

3.46 String primitive

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

3.47 (abstract,NC) OutagePlanningAgent root class

An entity with the task of planning the availability status of a relevant power generating module, a relevant demand facility or a relevant grid element.

3.48 (NC) OutageRequestResource root class

Associates an OutageRequest with a PowerSystemResource affected by the requested outage. Table 45 shows all attributes of OutageRequestResource.

Table 45 – Attributes of AvailabilityScheduleProfile::OutageRequestResource

name	mult	type	description
kind	1..1	AvailabilityFunctionKind	(NC) The requested operational availability function of the associated PowerSystemResource. This value is used as input when determining the resulting AvailabilitySchedule.
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.

Table 46 shows all association ends of OutageRequestResource with other classes.

Table 46 – Association ends of AvailabilityScheduleProfile::OutageRequestResource with other classes

mult from	name	mult to	type	description
0..*	PowerSystemResource	1..1	PowerSystemResource	(NC) The specific power system resource that is affected by the outage request.
1..*	OutageRequest	1..1	OutageRequest	(NC) The outage request that includes this specific affected resource.

3.49 (abstract) PowerSystemResource root class

A power system resource (PSR) can be an item of equipment such as a switch, an equipment container containing many individual items of equipment such as a substation, or an organisational entity such as sub-control area. Power system resources can have measurements associated.

3.50 (NC) OutageAvailability root class

Relationship between an OutageRequest and an AvailabilitySchedule, indicating that the outage request is one of the inputs or constraints used to establish that availability schedule. Multiple outage requests may contribute to a single availability schedule, and a single outage request may be reflected in multiple availability schedules issued by the system operator. Table 47 shows all attributes of OutageAvailability.

Table 47 – Attributes of AvailabilityScheduleProfile::OutageAvailability

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.

Table 48 shows all association ends of OutageAvailability with other classes.

Table 48 – Association ends of AvailabilityScheduleProfile::OutageAvailability with other classes

mult from	name	mult to	type	description
0..*	AvailabilitySchedule	1..1	AvailabilitySchedule	(NC) The availability schedule derived in part from this outage request.
0..*	OutageRequest	1..1	OutageRequest	(NC) The outage request that contributes to or influences the resulting availability schedule.

3.51 (NC) OutageDependency

Inheritance path = [PeerTemporalDependency](#)

A peer temporal dependency that defines temporal or operational constraints between two outage requests.

Note: The dependency constrains the start or end of the dependent outage relative to the start or end of the dependee outage. Lag constraints are evaluated using the latest applicable

transition of the dependee. This supports sequencing and coordination of related outages without implying a hierarchical relationship between the outage requests.

Example:

- startToStartLag = 1H, startToStartLagKind = minimum: the dependent outage may start no earlier than one hour after the dependee outage starts.

- finishToStartLag = 20M, finishToStartLagKind = exact: the dependent outage starts exactly 20 minutes after the dependee outage ends, for example to allow a defined switching or restoration sequence between the outages.

- finishToFinishLag = 30M, finishToFinishLagKind = maximum: the dependent outage shall end no later than 30 minutes after the dependee outage ends.

Table 49 shows all attributes of OutageDependency.

Table 49 – Attributes of AvailabilityScheduleProfile::OutageDependency

name	mult	type	description
constraintKind	1..1	DependencyConstraintKind	(NC) inherited from: PeerTemporalDependency
finishToFinishLag	0..1	Duration	(NC) inherited from: PeerTemporalDependency
finishToFinishLagKind	0..1	ReferenceValueKind	(NC) inherited from: PeerTemporalDependency
finishToStartLag	0..1	Duration	(NC) inherited from: PeerTemporalDependency
finishToStartLagKind	0..1	ReferenceValueKind	(NC) inherited from: PeerTemporalDependency
mRID	1..1	String	(NC) inherited from: PeerTemporalDependency
overlap	0..1	Duration	(NC) inherited from: PeerTemporalDependency
overlapKind	0..1	ReferenceValueKind	(NC) inherited from: PeerTemporalDependency
startToStartLag	0..1	Duration	(NC) inherited from: PeerTemporalDependency
startToStartLagKind	0..1	ReferenceValueKind	(NC) inherited from: PeerTemporalDependency

Table 50 shows all association ends of OutageDependency with other classes.

Table 50 – Association ends of AvailabilityScheduleProfile::OutageDependency with other classes

mult from	name	mult to	type	description
0..*	DependentOutageRequest	1..1	OutageRequest	(NC) The outage request whose timing or execution is constrained by the dependee outage request.
0..*	DependeeOutageRequest	1..1	OutageRequest	(NC) The outage request upon which the dependent outage request depends.

3.52 (NC) ReferenceValueKind enumeration

Kind of reference value.

Table 51 shows all literals of ReferenceValueKind.

Table 51 – Literals of AvailabilityScheduleProfile::ReferenceValueKind

literal	value	description
exact		The referenced value is to be met exactly.
minimum		The referenced value establishes a lower bound.
maximum		The referenced value establishes an upper bound.

3.53 Time primitive

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

3.54 (abstract,NC) PeerTemporalDependency root class

An abstract directed dependency between two peer instances of the same domain class that defines temporal and operational constraints governing their relative execution or operation.

Note: Temporal lag constraints relate the next applicable transition of the dependent to the latest applicable transition of the dependee. A dependee transition remains applicable while the dependee remains in the state resulting from that transition. If the dependee subsequently changes state before the corresponding transition of the dependent occurs, the previous transition is superseded and the constraint is evaluated from the next applicable transition. Once the corresponding transition of the dependent has occurred, the lag constraint is satisfied for that operating cycle.

Each lag is qualified by its corresponding ReferenceValueKind, which specifies whether the lag represents an exact, minimum, or maximum temporal constraint.

Table 52 shows all attributes of PeerTemporalDependency.

Table 52 – Attributes of AvailabilityScheduleProfile::PeerTemporalDependency

name	mult	type	description
constraintKind	1..1	DependencyConstraintKind	(NC) The kind of temporal or operational constraint defined between the dependee and the dependent. Note: Certain dependency constraint kinds may require or prohibit the use of temporal constraint attributes, depending on the semantics of the specialization.
finishToFinishLag	0..1	Duration	(NC) The duration constraining the next finish or deactivation of the dependent relative to the latest applicable finish or deactivation of the dependee.
finishToFinishLagKind	0..1	ReferenceValueKind	(NC) Specifies whether finishToFinishLag is an exact, minimum, or maximum temporal constraint. Note: This attribute shall be specified when finishToFinishLag is provided. Example: finishToFinishLag = 2D1H30M is interpreted together with finishToFinishLagKind: - exact: the dependent finishes exactly 2 days, 1 hour and 30 minutes after the latest applicable finish of the dependee. - minimum: the dependent may finish no earlier than 2 days, 1 hour and 30 minutes after the latest applicable finish of the dependee. - maximum: the dependent shall finish no later than 2 days, 1 hour and 30 minutes after the latest applicable finish of the dependee. In all cases, the dependee finish remains applicable only while the dependee remains in the state resulting from that finish. If the dependee subsequently changes state before the dependent finishes, the applicability of the previous finish is re-evaluated.
finishToStartLag	0..1	Duration	(NC) The duration constraining the next start or activation of the dependent relative to the latest applicable finish or deactivation of the dependee.

name	mult	type	description
finishToStartLagKind	0..1	ReferenceValueKind	(NC) Specifies whether finishToStartLag is an exact, minimum, or maximum temporal constraint. Note: This attribute shall be specified when finishToStartLag is provided. Example: finishToStartLag = 20M is interpreted together with finishToStartLagKind: - exact: the dependent starts exactly 20 minutes after the latest applicable finish of the dependee. - minimum: the dependent may start no earlier than 20 minutes after the latest applicable finish of the dependee. - maximum: the dependent shall start no later than 20 minutes after the latest applicable finish of the dependee. In all cases, the dependee finish remains applicable only while the dependee remains in the state resulting from that finish. If the dependee subsequently changes state before the dependent starts, the constraint is evaluated from the next applicable finish of the dependee.
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.
overlap	0..1	Duration	(NC) The duration constraining the period for which the dependee and dependent are simultaneously active or effective.
overlapKind	0..1	ReferenceValueKind	(NC) Specifies whether overlap is an exact, minimum, or maximum duration of simultaneous activity or effectiveness. Note: This attribute shall be specified when overlap is provided. Example: overlap = 1H28M45S is interpreted together with overlapKind: - exact: the dependee and dependent are simultaneously active or effective for exactly 1 hour, 28 minutes, and 45 seconds. - minimum: the dependee and dependent shall be simultaneously active or effective for at least 1 hour, 28 minutes, and 45 seconds. - maximum: the dependee and dependent shall be simultaneously active or effective for no more than 1 hour, 28 minutes, and 45 seconds. Unlike the lag attributes, overlap constrains the duration of a simultaneous state rather than the temporal separation between two state transitions.
startToStartLag	0..1	Duration	(NC) The duration constraining the next start or activation of the dependent relative to the latest applicable start or activation of the dependee.
startToStartLagKind	0..1	ReferenceValueKind	(NC) Specifies whether startToStartLag is an exact, minimum, or maximum temporal constraint.

name	mult	type	description
			Note: This attribute shall be specified when startToStartLag is provided. Example: startToStartLag = 2H is interpreted together with startToStartLagKind: - exact: the dependent starts exactly 2 hours after the latest applicable start of the dependee. - minimum: the dependent may start no earlier than 2 hours after the latest applicable start of the dependee. - maximum: the dependent shall start no later than 2 hours after the latest applicable start of the dependee. In all cases, the dependee start remains applicable only while the dependee remains in the state resulting from that start. If the dependee subsequently changes state before the dependent starts, the constraint is evaluated from the next applicable start of the dependee.

3.55 (NC) DependencyConstraintKind enumeration

The kind of dependency constraint defined between the dependee and dependent peer objects. Note: The dependee is the peer object upon which another peer object depends. The dependent is the peer object whose temporal or operational behaviour is constrained by the dependee. Although dependee is not a commonly used English term, it is used here to distinguish this relationship from a parent-child hierarchy, as both objects are peer instances of the same class. Table 53 shows all literals of DependencyConstraintKind.

Table 53 – Literals of AvailabilityScheduleProfile::DependencyConstraintKind

literal	value	description
exclusive		The dependee and dependent shall not both be active or effective at the same time.
inclusive		The dependent shall only be active or effective while the dependee is active or effective.
restrictive		The dependee constrains the dependent without requiring the dependent to become active or effective. Note: A restrictive dependency defines temporal or operational constraints on the behaviour of the dependent while leaving its activation or operation optional.

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

866 **A.1 General**

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

871 **A.2 Sample instance data**

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

873

874