Table of contents

Note: all examples are written in python 2.7 and assume that python suds module is installed. Before connecting to WebAPI soap client needs to be created:

```
>>from suds.client import Client
>>soapclient = Client("http://localhost:43080/eds.wsdl")
```

Substitute eds.wsdl address with the one from your configuration

Factory functions

Factory functions allow to easily prepare structures used later in a number of functions

Timestamp

Creates timestamp attribute e.g:

```
>>> timestamp = soapclient.factory.create('Timestamp')
>>> print (timestamp)
(Timestamp){
   second = None
 }
>>> timestamp.second = time.time()
```

TimePeriod

Prepares a structure storing timespan used to download events and trends. Used in various filters

```
>>> timeperiod = soapclient.factory.create('TimePeriod')
>>> print (timeperiod)
(TimePeriod){
   from =
      (Timestamp){
         second = None
      }
   till =
      (Timestamp){
         second = None
      }
 }
```

To access filter attribites use <timeperiod>.<filter_attribute> notation. E.g:

```
>>> timeperiod.till.second = time.time()
```

Notes:

- from and till attributes can be created with soap client factory function
- 'second' attribute has to be UNIX timestamp.
- in Python 'from' is a reserved keyword and this attribute has to be accessed with getattr built-in function e.g:

```
>>> getattr(timeperiod, 'from').second = time.time()
```

EventFilter

Structure that holds criteria for event filtering, it's used in requestEvents funtion.

```
>>> eventfilter = soapclient.factory.create('EventFilter')
>>> print (eventfilter)
(EventFilter){
   period =
      (TimePeriod){
         from =
            (Timestamp){
               second = None
            }
         till =
            (Timestamp){
               second = None
            }
      }
   category[] = <empty>
   rt[] = <empty>
   priority[] = <empty>
   tg[] = <empty>
   zd[] = <empty>
   iessRe = None
   messageRe = None
   pointId[] = <empty>
 }
```

To access filter attribites use <filter>.<filter_attribute> notation. E.g:

```
eventfilter.iessRe = 'some_point_name'
```

Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
period	Period for which events are to be downloaded	TimePeriod	Created with factory function
category	Event category	enum	Accepted values: EVENT-CATEGORY-SYSTEM EVENT-CATEGORY-CUSTOM EVENT-CATEGORY-EXTERNAL EVENT-CATEGORY-ALARM EVENT-CATEGORY-SET-POINT
rt	Point type	enum	Accepted values: POINT-TYPE-UNKNOWN POINT-TYPE-ANALOG POINT-TYPE-ANALOG POINT-TYPE-BINARY POINT-TYPE-PACKED

			POINT-TYPE-PACKED
priority	Event priority	unsignedByte	Priority values change from 1 to 8
tg	Technological group	unsignedByte	Technological group numbers can change from 0 to 255
zd	Point source	string	Accepts regular expressions
iessRe	IESS point name	string	Accepts regular expressions
idcsRe	IDCS point name	string	Accepts regular expressions
pointID	pointId structure	PointID	Created with factory function

AlterableObjectAttributes

Used in alterObject function. Prepares a structure for new attributes of an object.

```
>>> newAttributes =
soapclient.factory.create('AlterableObjectAttributes')
>>> print(newAttributes)
(AlterableObjectAttributes){
   name = None
   sg[] = <empty>
   tg[] = <empty>
}
```

If some attribute in newAttributes is not set then it is not changed. Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
name	New name for an object	string	optional
sg	New security groups for an object	unsignedByte	optional
tg	New technological group for an object	unsignedByte	optional

AlterableObjectSourceAttributes

Used in alterObjectSource function. Prepares a structure for new attributes of a source.

```
>>> newAttributes =
soapclient.factory.create('AlterableObjectSourceAttributes')
>>> print(newAttributes)
(AlterableObjectSourceAttributes){
   desc = None
   sg[] = <empty>
   tg[] = <empty>
}
```

If some attribute in newAttributes is not set then it is not changed. Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**

desc	New description for a source	string	optional
sg	New security groups for a source	unsignedByte	optional
tg	New technological group for a source	unsignedByte	optional

ObjectFilter

Used in getObjectsMetadata, getReportsConfigs functions. Allows querying EDS92 for objects matching selected criteria.

```
>>> filter = soapclient.factory.create('ObjectFilter')
>>> print(filter)
(ObjectFilter){
   fileRe = None
   nameRe = None
   sourceNameRe = None
   sourceId[] = <empty>
   modified = None
   sg[] = <empty>
   tg[] = <empty>
   md5[] = <empty>
   withoutPermission = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
fileRe	Filename of objects	string	optional
nameRe	Name of objects	string	optional
sourceNameRe	Source name of objects	string	optional
sourceId	Source ID of objects	unsignedInt	optional
modified	Objects modified in range between two points in time	TimePeriod	optional • from [Timestamp] Represents a point in time, defined as the number of seconds elapsed since midnight proleptic Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds. ○ second [long] • till [Timestamp] Represents a point in time, defined as the number of seconds elapsed

			since midnight proleptic Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds. ○ second [long]
sg	Security group of objects	unsignedByte	optional
tg	Technological group of objects	unsignedByte	optional
md5	MD5 value of objects	string	optional
withoutPermission	Objects with param hasPermission = False	boolean	optional

ObjectSourceFilter

Used in getObjectsSources. Prepare structure to filter object source list.

```
>>> filter = soapclient.factory.create('ObjectSourceFilter')
>>> print(filter)
(ObjectSourceFilter){
   id[] = <empty>
   nameRe = None
   descRe = None
   sg[] = <empty>
   tg[] = <empty>
   kind = None
   hostRe = None
   prefixRe = None
   suffixRe = None
   optionsSet = None
   optionsUnset = None
   withoutPermission = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
id	ID of sources	unsignedInt	optional
nameRe	Name of sources	string	optional
descRe	Description of sources	string	optional
sg	Security groups of sources	unsignedByte	optional
tg	Technological groups of sources	unsignedByte	optional

kind	Object source kind list	ObjectSourceKind	Optional [string] - enum { 'OBJECT-SOURCE-KIND-UNKNOWN', 'OBJECT-SOURCE-KIND-DISK', 'OBJECT-SOURCE-KIND-FTP', 'OBJECT-SOURCE-KIND-DB' }
hostRe	Host of sources	string	optional
prefixRe	Prefix of sources	string	optional
suffixRe	Suffix of sources	string	optional
optionsSet	Options set up of sources	unsignedInt	optional
optionsUnset	Options unset of sources	unsignedInt	optional
withoutPermission	hasPermission = False	boolean	optional

DiagramResolution

Used in resizeDiagram function. Prepares a structure for new resolution.

```
>>> newResolution = soapclient.factory.create('DiagramResolution')
>>> print(newResolution)
(DiagramResolution){
   width = None
   height = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
width	New width of the diagram	unsignedShort	Value in pixels
height	New height of the diagram	unsignedShort	Value in pixels

DiagramArea

Used in setDiagramViewport function. Prepares a structure for new diagram viewport. It contains topLeft and bottomRight parametres which are another factory functions.

```
>>> viewport = soapclient.factory.create('DiagramArea')
>>> print(viewport)
(DiagramArea){
   topLeft =
      (DiagramPosition){
```

```
         x = None
         y = None
      }
   bottomRight =
      (DiagramPosition){
         x = None
         y = None
      }
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
topLeft	Represents relative position on diagram where [0.0, 0.0] is a top left diagram coordinate and [1.0, 1.0] is a bottom right diagram corner.	Factory functio DiagramPosition	
bottomRight	Represents relative position on diagram where [0.0, 0.0] is a top left diagram coordinate and [1.0, 1.0] is a bottom right diagram corner.	Factory functio DiagramPosition	

DiagramPosition

Used in DiagramArea factory function. Prepares a structure for relative position on diagram.

```
>>> dp = soapclient.factory.create('DiagramPosition')
>>> print(dp)
(DiagramPosition){
   x = None
   y = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes

x	Location pointer on the x-axis	Factory functio DiagramPosition	On diagram there [0.0, 0.0] is a top left diagram coordinate and [1.0, 1.0] is a bottom right diagram corner.
y	Location pointer on the y-axis	Factory functio DiagramPosition	On diagram there [0.0, 0.0] is a top left diagram coordinate and [1.0, 1.0] is a bottom right diagram corner.

ReportConfigDefinition

Used in createReportConfig and alterReportConfig. Prepares a structure for report configuration definition. At least one of the outputMask field must be filled.

```
>>> config = soapclient.factory.create('ReportConfigDefinition')
>>> print(config)
(ReportConfigDefinition){
   referenceTimeShift = None
   runDelay =
      (TimeDuration){
         seconds = None
      }
   timeMaskExpression = None
   eventsExpression = None
   inputValues = None
   outputMaskFileTxt = None
   outputMaskFileRdf = None
   outputMaskFileEdf = None
   outputMaskFileHtml = None
   outputMaskFilePdf = None
   outputMaskFileCsv = None
   outputMaskDatabaseRdf = None
   outputMaskDatabaseEdf = None
   outputMaskDatabaseHtml = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
referenceTimeShift	Reference time shift	string	
runDelay	Represents a length of time	string	
timeMaskExpression	Time mask expression	string	
eventsExpression	Events expression	string	
inputValues		string	

outputMaskFileTxt	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskFileRdf	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskFileEdf	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskFileHtml	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskFilePdf	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskFileCsv	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskDatabaseRdf	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskDatabaseEdf	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled
outputMaskDatabaseHtml	Absolute path to the file in which the data is to be written	string	Optional. At least one of the outputMask field must be filled

ReportConfigFilter

Used in getReportsConfig function. Function allows querying for reports configs matching selected criteria. It contains another factory function ObjectFilter.

```
>>> filter = soapclient.factory.create('ReportConfigFilter')
>>> print(filter)
(ReportConfigFilter){
   objectFilter = None
   outputType = None
```

```
   executionCondition = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
objectFilter	Filter allows selecting objects using values of most of their fields	Factory function ObjectFilter	optional
outputType	Output type	string	enum { 'REPORT-OUTPUT-TYPE-UNKNOWN', 'REPORT-OUTPUT-TYPE-FILE-TXT', 'REPORT-OUTPUT-TYPE-FILE-RDF', 'REPORT-OUTPUT-TYPE-FILE-EDF', 'REPORT-OUTPUT-TYPE-FILE-HTML', 'REPORT-OUTPUT-TYPE-FILE-PDF', 'REPORT-OUTPUT-TYPE-FILE-CSV', 'REPORT-OUTPUT-TYPE-DATABASE-RDF', 'REPORT-OUTPUT-TYPE-DATABASE-EDF', 'REPORT-OUTPUT-TYPE-DATABASE-HTML' }
executionCondition	Execution condition	string	enum { 'REPORT-EXECUTION-CONDITION-UNKNOWN', 'REPORT-EXECUTION-CONDITION-CYCLIC', 'REPORT-EXECUTION-CONDITION-ON-EVENT' }

ReportDefinition

Used in createGlobalReport function. It prepares a structure for rdf file. It contains other factory functions: ReportDefinitionRow and ReportDefinitionCell.

```
>>> rdf = soapclient.factory.create('ReportDefinition')
>>> print(rdf)
(ReportDefinition){
   localTime = None
   showDstTransition = None
   showQualities = None
   precisions = None
   timeMode = None
   addressingType = None
   shadesPriority = None
   rows[] = <empty>
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
localTime	Sets up if local time is used	boolean	Optional
showDstTransition	Sets up if dst. Transition will be shown	boolean	Optional
showQualities	Sets up if quality of the points will be shown	boolean	Optional
precisions	Precision parameter of the rdf file	unsignedByte	Optional
timeMode	Report time mode	string	Optional. enum { 'REPORT-TIME-MODE-RELATIVE', 'REPORT-TIME-MODE-ABSOLUTE' }
addressingType	Report addressing type	string	Optional. enum { 'REPORT-ADDRESSING-TYPE-R1C1', 'REPORT-ADDRESSING-TYPE-A1' }
shadesPriority	Shades priority	string	Optional. enum { 'REGULAR-OVER-SHADE', 'SHADE-OVER-REGULAR', 'REGULAR-ONLY', 'SHADE-ONLY', 'DEFAULT-SHADE-PRIORITY' }
rows	Definition of rows in the rdf file	Factory function ReportDefinitionRow	Optional. It contains ReportDefinitionCell factory function

ReportDefinitionRow

Used in ReportDefinition factory function. It contains ReportDefinitionCell factory function.

```
>>> rows = soapclient.factory.create('ReportDefinitionRow')
>>> print(rows)
(ReportDefinitionRow){
   cells[] = <empty>
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
cells	Single report definition cell	ReportDefinitionCell factory function	Optional

ReportDefinitionCell

Used in ReportDefinitionRow factory function. It is single report definition cell.

```
>>> cells = soapclient.factory.create('ReportDefinitionCell')
>>> print(cells)
(ReportDefinitionCell){
   content = None
   showQuality = None
   precision = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
content	Content of the cell	string	Content field should always contain at least one character. Otherwise cell may be disregarded in SOAP serialization. Empty cells can be represented by a single space character. All time expressions must be in seconds relative to @DT_REF value.
showQuality	Sets up if quality of the point will be displayed in a cell	boolean	Optional
precision	Sets up precision of a cell	unsignedByte	Optional

CustomReportRequest

Used in requestCustomReport function. It contains ReportDefinition, ReportDefinitionRow, ReportDefinitionCell, Timestamp, Report Argument and report Value factory functions

```
>>> request = soapclient.factory.create('CustomReportRequest')
>>> print(request)
(CustomReportRequest){
   rdf =
      (ReportDefinition){
         localTime = None
         showDstTransition = None
         showQualities = None
         precisions = None
         timeMode = None
```

```
         addressingType = None
         shadesPriority = None
         rows[] = <empty>
      }
   dtRef =
      (Timestamp){
         second = None
      }
   args[] = <empty>
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
rdf	Report definition	ReportDefinition factory function	
dtRef	Date reference	Timestamp factory function	
args	Arguments	ReportArgument factory function	Optional

ReportArgument

Used in CustomReportRequest factory function. Prepares structure of report arguments. It contains ReportValue factory function.

```
>>> args = soapclient.factory.create('ReportArgument')
>>> print(args)
(ReportArgument){
   name = None
   value =
      (ReportValue){
         boolean = None
         number = None
         packed = None
         string = None
         timestamp = None
         point = None
         quality = None
      }
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
name	Name of the argument	string	
value	Value of the argument	ReportValue factory function	

ReportValue

Used in ReportArgument factory function. It prepares a structure for report argument value.

```
>>> value = soapclient.factory.create('ReportValue')
>>> print(value)
(ReportValue){
   boolean = None
   number = None
   packed = None
   string = None
   timestamp = None
   point = None
   quality = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
boolean	Boolean value of the report argument	boolean	Optional
number	Number value of the report argument	double	Optional
packed	Packed value of the report argument	unsignedInt	Optional
string	String value of the report argument	string	Optional
timestamp	Represents a point in time, defined as the number of seconds elapsed since midnight proleptic Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds.	Timestamp factory function	Optional
point	Point value of the report argument	string	Optional

quality	Quality value of the report argument	string	Optional. enum { 'QUALITY-NONE', 'QUALITY-GOOD', 'QUALITY-FAIR', 'QUALITY-POOR', 'QUALITY-BAD' }

GlobalReportRequest

Used in requestGlobalReport. Prepares structure for global report execution. It contains facotry functions: Timestamp, ReportArgument, ReportValue

```
>>> request = soapclient.factory.create('GlobalReportRequest')
>>> print(request)
(GlobalReportRequest){
   reportConfigId = None
   dtRef =
      (Timestamp){
         second = None
      }
   args[] = <empty>
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
reportConfigId	Id of report config to execute	unsignedInt	
dtRef	Date reference	Timestamp factory function	
args	Arguments	ReportArgument factory function	Optional

TimeDuration

Used in ReportConfigDefinition factory function. Stores the length of time in seconds.

```
>>> td = soapclient.factory.create('TimeDuration')
>>> print(td)
(TimeDuration){
   seconds = None
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes

seconds	Length of time in seconds	long	

PointFilter

Used in getPoints function. Prepares a structure storing point filtering criteria

```
>>> filter = soapclient.factory.create('PointFilter')
>>> print (filter)
(PointFilter){
   sid[] = <empty>
   iessRe = None
   idcsRe = None
   zdRe = None
   descRe = None
   auxRe = None
   rt[] = <empty>
   ts =
      (TimePeriod){
         from =
            (Timestamp){
               second = None
            }
         till =
            (Timestamp){
               second = None
            }
      }
   quality[] = <empty>
   stSet = None
   stUnset = None
   dfSet = None
   dfUnset = None
   ar[] = <empty>
   artd[] = <empty>
   sg[] = <empty>
   tg[] = <empty>
   ap[] = <empty>
   withoutPermission = None
 }
```

To access filter attribites use <filter>.<filter_attribute> notation. E.g:

```
filter.iessRe = 'some_point_name'
```

String filter attributes can be set to regular expressions to find several points meeting various criteria. Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**

sid	Point index in database	unsigned int	
iessRe	Point IESS name	string	accepts regular experssions
idcsRe	Point IDCS (i.e source system) name	string	accepts regular experssions
zdRe	Point source	string	accepts regular experssions
descRe	Point description	string	accepts regular experssions
auxRe	Auxiliary field content	string	accepts regular experssions
rt	Point type	enum	Accepted values: POINT-TYPE-UNKNOWN POINT-TYPE-ANALOG POINT-TYPE-ANALOG POINT-TYPE-BINARY POINT-TYPE-PACKED POINT-TYPE-PACKED
ts	Point timestamp	TimePeriod	Can be generated with factory function
quality	Point quality	enum	Accepted values: QUALITY-NONE QUALITY-GOOD QUALITY-POOR QUALITY-FAIR QUALITY-BAD
ar	Archiving type	enum	Accepted values: ARCHIVING-UNKNOWN ARCHIVING-NONE ARCHIVING-LOCAL ARCHIVING-EXTERNAL ARCHIVING-FILLIN
artd	Archiving deadband type	enum	Accepted values: ARCHIVING-DEADBAND-UNKNOWN ARCHIVING-DEADBAND-NONE ARCHIVING-DEADBAND-FLOW ARCHIVING-DEADBAND-LOG ARCHIVING-DEADBAND-PCT-RANGE ARCHIVING-DEADBAND-POWER ARCHIVING-DEADBAND-RADIATION ARCHIVING-DEADBAND-RATIO ARCHIVING-DEADBAND-STANDARD ARCHIVING-DEADBAND-TEST ARCHIVING-DEADBAND-GEOMETRIC1 ARCHIVING-DEADBAND-GEOMETRIC3 ARCHIVING-DEADBAND-TRAPEZOIDAL
sg	Security group	unsigned byte	
tg	Technological group	unsigned byte	
ap	Alarm priority	unsigned byte	

PointId

Creates PointID structure used to identify EDS point.

Example:

```
>>> pointid = soapclient.factory.create('PointId')
>>> print (pointid)
(PointId){
   sid = None
   iess = None
   idcs = None
   zd = None
 }
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
sid	Point identifier	unsigned int	
iess	Point IESS name	string	
idcs	Point IDCS name	string	
zd	Point source	string	

PointValue

Structure storing point value. Only one of attributes need to be set depending on point type:

Example:

```
>>> val = soapclient.factory.create('PointValue')
>>> print (val)
(PointValue){
   av = None
   dav = None
   bv = None
   pv = None
   ipv = None
 }
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
av	Analog value	float	
dav	Double value	double	
bv	Binary value	boolean	
pv	Packed point value	unsigned int	
ipv	Int64 point value	long	

PointsNewValue

Structure storing point id, value and quality to be sent to EDS. Attributes can be accesed directly or created with other factory functions

Example:

```
>>> pointval = soapclient.factory.create('PointNewValue')
>>> pointid = soapclient.factory.create('PointId')
>>> pointval.id = pointid
>>> pointval.value.av = 567.567
```

```
>>> pointval.quality = 'QUALITY-POOR'
>>> print (pointval)
(PointNewValue){
   id =
      (PointId){
         sid = None
         iess = "811445_IESS"
         idcs = None
         zd = None
      }
   value =
      (PointValue){
         av = 567.567
         dav = None
         bv = None
         pv = None
         ipv = None
      }
   quality = "QUALITY-POOR"
   duration =
      (TimeDuration){
         seconds = None
      }
 }
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
id	Point identifier	PointId	Can be created with factory function
value	Point value	PointValue	Can be created with factory function
quality	Point quality	Enum	Accepted values: QUALITY-NONE QUALITY-GOOD QUALITY-POOR QUALITY-FAIR QUALITY-BAD
duration	Time for which value is valid	TimeDuration	Can be created with factory function

ShadeValue

Structure storing a single substitution value. Attributes can be accesed directly or created with other factory functions

Example:

```
>>> shade_value = soapclient.factory.create('ShadeValue')
>>> print (shade_value)
(ShadeValue){
   value =
      (PointValue){
         av = None
         dav = None
         bv = None
```

```
         pv = None
         ipv = None
      }
   quality =
      (Quality){
         value = None
      }
   period =
      (TimePeriod){
         from =
            (Timestamp){
               second = None
            }
         till =
            (Timestamp){
               second = None
            }
      }
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
value	Point value	PointValue	Can be created with factory function
quality	Point quality	Enum	Accepted values: QUALITY-NONE QUALITY-GOOD QUALITY-POOR QUALITY-FAIR QUALITY-BAD
period	Time for which substitution value is valid	TimePeriod	Can be created with factory function

Shade

Structure storing point substitution values. Attributes can be accesed directly or created with other factory functions

```
>>> shade = soapclient.factory.create('Shade')
>>> print (shade)
(Shade){
   pointId =
      (PointId){
         sid = None
         iess = None
         idcs = None
         zd = None
      }
   values[] = <empty>
}
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
pointId	Point identifier	PointId	Can be created with factory function
values[]	Substitution values	Array of ShadeValue	Each ShadeValue can be created with factory function

ShadeSelector

Structure storing data that allows for identification of substitution data with point name and period for which values are to be downloaded from database

Example:

```
>>> shade_selector = soapclient.factory.create('ShadeSelector')
>>> print (shade_selector)
(ShadeSelector){
   pointId =
      (PointId){
         sid = None
         iess = None
         idcs = None
         zd = None
      }
   period =
      (TimePeriod){
         from =
            (Timestamp){
               second = None
            }
         till =
            (Timestamp){
               second = None
            }
      }
 }
```

Table below lists all attributes and types accepted by each attribute

Attribute	Description	Type	Notes
pointId	Point identifier	PointId	Can be created with factory function
period	Time for which substitution value is valid	TimePeriod	Can be created with factory function

ShadeCopyRequest

Structure storing source and destination point data as well as a period of time for which substitution data is to be copied

Example:

```
>>> shade_copy_request =
soapclient.factory.create('ShadeCopyRequest')
```

```
>>> print (shade_copy_request)
(ShadeCopyRequest){
   srcPointId =
      (PointId){
         sid = None
         iess = None
         idcs = None
         zd = None
      }
   dstPointId =
      (PointId){
         sid = None
         iess = None
         idcs = None
         zd = None
      }
   period =
      (TimePeriod){
         from =
            (Timestamp){
               second = None
            }
         till =
            (Timestamp){
               second = None
            }
      }
 }
```

Table below lists all attributes and types accepted by each attribute

Attribute	**Description**	**Type**	**Notes**
srcPointId	Source point identifier	PointId	Can be created with factory function
dstPointId	Destination point identifier	PointId	Can be created with factory function
period	Time for which substitution value is valid	TimePeriod	Can be created with factory function

Login and status functions

login(user_name, password)

Login function allows user to obtain authstring used later in with other functions. Authstring will become invalid if not used. Use ping function to keep it valid

Example:

```
>>> authstring = soapclient.service.login('admin','')
>>> print (authstring)
a84e9d3c-721d-5e96-8ffd-60db1207d6c1
```

logout(authstring)

Function allowing user logout

Example:

```
>>> soapclient.service.logout(authstring)
```

ping(authstring)

Function allowing to keep authstring valid

Example:

```
>>> soapclient.service.ping(authstring)
```

authenticate(username, password)

Checks user name and password.

Example:

```
>>> authResCorrect = soapclient.service.authenticate('admin','')
>>> print (authResCorrect)
True
>>> authResIncorrect =
soapclient.service.authenticate('admin','foo321')
>>> print (authResIncorrect)
False
```

getServerTime(authstring)

Function allowing to check EDS server time and timezone. Time.seconds attribute is a Unix timestamp

Example:

```
>>> response = soapclient.service.getServerTime(authstring)
>>> print (response)
(reply){
   time =
      (Timestamp){
         second = 1466688892
      }
   zone =
      (TimeDuration){
```

```
         seconds = 3600
      }
   offset =
      (TimeDuration){
         seconds = 7200
      }
 }
```

It is possible to refer to response attributes e.g:

```
>>> print (response.zone.seconds)
3600
```

getServerStatus(authstring)

Function allowing to check current WebAPI server status

Example:

```
>>> response = soapclient.service.getServerStatus(authstring)
>>> print (response)
(reply){
   startTime =
      (Timestamp){
         second = 1466687394
      }
   soapConnectionCount = 0
   soapHttpsConnectionCount = 0
   sessionCount = 1
   requestCount = 0
   requestRunningCount = 0
   srvConnection = "LOGGED_IN | SYNCHRONIZED | STATIC_CHANGED"
   objConnection = "LOGGED_IN"
   globalObjectCount = 5738
   pendingObjectCount = 0
   liveDataConnectionCount = 0
 }
```

It is possible to refer to response attributes e.g:

```
>>> print (response.srvConnection)
LOGGED_IN | SYNCHRONIZED | STATIC_CHANGED
```

getServerConfig(authstring)

Function allowing to check EDS server configuration parameters

Example:

```
>>> response = soapclient.service.getServerConfig(authstring)
```

```
>>> print (response)
(reply){
   srvHost = "localhost"
   srvPort = 43000
   objHost = "localhost"
   objPort = 43051
   archHost = "localhost"
   archPort = 43001
 }
>>> print (response.srvHost)
localhost
```

getSecurityGroups(authstring)

Function allowing to list all 255 EDS security groups

```
>>> response = soapclient.service.getSecurityGroups(authstring)
```

getTechnologicalGroups(authstring)

Function allowing to list all 255 EDS security groups

```
>>> response = soapclient.service.getTechnologicalGroups
(authstring)
```

getUserSecurityGroups(authstring)

Function allowing to list all EDS security groups user belong to

```
>>> response = soapclient.service.getUserSecurityGroups(authstring)
```

Example:

```
>>> print (response)
[(Group){
   id = 0
   name = "admin"
   desc = "Administrators"
}]
```

getLicense(authstring)

Function allowing to list all EDS license parameters

```
>>> response = soapclient.service.getLicense (authstring)
```

getEffectiveUserProfile(authstring)

Allows querying EDS92 user effective profile. Effective profile is composition of all profiles with user security groups access.

Example:

```
>>> response =
soapclient.service.getEffectiveUserProfile(authstring)
>>> print (response)
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive"
version="17">
<Root class_id="0" tracking_level="1" version="0" object_id="_0">
<mValue></mValue>
<mEditable>1</mEditable>
<mCanHaveExtraSubNodes>1</mCanHaveExtraSubNodes>
<mSubNodes class_id="1" tracking_level="0" version="0">
<count>0</count>
<item_version>0</item_version>
</mSubNodes>
</Root>
</boost_serialization> s"
```

Request functions

All functions that don't finish immediately return request number that can be used to monitor request status. When request has been processed request id is used to get server response. Functions that are processed as requests:

requestEvents
requestTrend
requestTabular
requestScriptRun
requestCustomReport
requestGlobalReport
requestShadesClear
requestShadesCopy
requestShadesRead
requestShadesWrite

getRequestStatus(authstring, request_id)

Returns one of the following statuses:

`REQUEST-QUEUED`

`REQUEST-EXECUTING`

`REQUEST-FAILURE`

`REQUEST-SUCCESS`

dropRequest(authstring, request_id)

Cancels execution of request with request_id

requestShadesWrite(authstring, newShades)

Requests the server to write new or overwrite existing points substitution values. newShades argument is a structure comprised of PointID and ShadeValues structures described earlier in 'Factory functions' section.

Example:

Creation of all necessary data structures with factory functions:

```
>>> newShade = soapclient.factory.create('Shade')
>>> pointId = soapclient.factory.create('PointId')
>>> values = soapclient.factory.create('ShadeValue')
>>> value = soapclient.factory.create('PointValue')
>>> quality = soapclient.factory.create('Quality')
>>> period = soapclient.factory.create('TimePeriod')
```

Adding data to structures:

```
>>> period.till.second = 1583881200
>>> getattr(period, 'from').second = 1583794800
>>> values.period = period
>>> values.quality.value = "QUALITY-GOOD"
>>> values.value.av = 100
```

```
>>> newShade.values.append(values)
>>> pointId.iess = 'TEST_POINT'
>>> newShade.pointId = pointId
```

Example of complete Shade structure:

```
>>> newShade
(Shade){
   pointId =
      (PointId){
         sid = None
         iess = "TEST_POINT"
         idcs = None
         zd = None
      }
   values[] =
      (ShadeValue){
         value =
            (PointValue){
               av = 100
               dav = None
               bv = None
               pv = None
               ipv = None
            }
         quality =
            (Quality){
               value = "QUALITY-GOOD"
            }
         period =
            (TimePeriod){
               from =
                  (Timestamp){
                     second = 1583794800
                  }
               till =
                  (Timestamp){
                     second = 1583881200
                  }
            }
      },
 }
```

With all data written to structures requestShadesWrite function can be called:

```
>>> soapclient.service.requestShadesWrite(authstring, newShade)
551026
```

The number returned is request id which can be used to check operation status with getRequestStatus function. Operation is complete when getRequestStatus returns REQUEST-SUCCESS

requestTabular(authstring, request)

Requests the server to execute tabular trend i.e return values or aggregates for a given period of time with given step. Arguments are authentication string and TabularRequest structure that can be created with factory function. requestTabular returns request ID that can be used to track execution progress and get results when request is ready.

Example:

Commands in below example send a request to return 5 second averages for 24 hour period.

Creation of all necessary data structures with factory functions:

```
>>> tabRequest = soapclient.factory.create('TabularRequest')
>>> period = soapclient.factory.create('TimePeriod')
>>> item = soapclient.factory.create('TabularRequestItem')
>>> pointId = soapclient.factory.create('PointId')
```

Adding data to structures:

```
>>> period.till.second = 1583881200
>>> getattr(period, 'from').second = 1583794800
>>> tabRequest.period = period
>>> tabRequest.step.seconds = 5
>>> pointId.iess = 'TEST_POINT'
>>> item.pointId = pointId
>>> item.function = 'AVG'
>>> tabRequest.items.append(item)
```

With all data written to structures requestShadesWrite function can be called:

```
>>> soapclient.service.requestTabular(authstring, tabRequest)
351532
```

The number returned is request id which can be used to check operation status with getRequestStatus function. Operation is complete when getRequestStatus returns REQUEST-SUCCESS and results can be accessed with getTabular function

requestTrend(authstring, request)

Executes graphical trend. "pixelCount" parameter is just a suggestion to ArchSrv. When requesting trend for long time range 1 pixel can correspond to e.g. 1 day, that's why trend request may return up to 5 samples per pixel (value at the start of the range, maximum, minimum, value at the end, hole marker). See also getRequestStatus() and getTrend().

The function takes authentication string and a structure of TrendRequest type. The structure can be created by factory function.

Example:

Creation of all necessary data structures with factory functions:

```
>>> trendRequest = soapclient.factory.create('TrendRequest')
>>> period = soapclient.factory.create('TimePeriod')
>>> item = soapclient.factory.create('TrendRequestItem')
```

```
>>> pointId = soapclient.factory.create('PointId')
```

Adding data to structures:

```
>>> trendRequest.useAverages = False
>>> trendRequest.pixelCount = 1500
>>> period.till.second = 1583881200
>>> getattr(period, 'from').second = 1583794800
>>> trendRequest.period = period
>>> pointId.iess = 'TEST_POINT'
>>> item.pointId = pointId
>>> item.shadePriority.value = 'DEFAULT-SHADE-PRIORITY'
>>> trendRequest.items.append(item)
```

With all data written to structures requestTrend function can be called:

```
>>> soapclient.service.requestTrend(authstring, trendRequest)
765249
```

The number returned is request id which can be used to check operation status with getRequestStatus function. Operation is complete when getRequestStatus returns REQUEST-SUCCESS and results can be accessed with getTrend function

Points functions

getPointsSources(authstring)

Returns a list of point sources available in the system

Example:

```
>>> result = soapclient.service.getPointsSources(authstring)
>>> print (result)
[MON, Net0, Net1, Net2, Net3, SERVER, Simulate]
```

getPoints(authstring, filter, order, startIdx, maxCount)

Returns a list of points available in EDS database. Only authstring parameter is obligatory and if only authstring is given a list of all EDS points is returned.

Filter is a structure returned by soap client factory function (see PointFilter function in Factory functions section)

Example:

Below example uses filter to find all points belonging to source Net1 with name beginning with ‘81146’

```
>>> filter = soapclient.factory.create('PointFilter')
>>> filter.zdRe = 'Net1'
>>> filter.iessRe = '81146.'
>>> result = soapclient.service.getPoints(authstring, filter)
>>> print (result)
(reply){
```

```
points[] =
   (Point){
      id =
         (PointId){
            sid = 909948
            iess = "811461_IESS"
            idcs = "811461_IDCS"
            zd = "Net1"
         }
      rt = "POINT-TYPE-BINARY"
      value =
         (PointValue){
            bv = False
         }
      quality = "QUALITY-BAD"
      ts =
         (Timestamp){
            second = 0
         }
      tss =
         (TimeDuration){
            seconds = 0
         }
      at =
         (Timestamp){
            second = 0
         }
      atss =
         (TimeDuration){
            seconds = 0
         }
      desc = "811461_DESC"
      st = 33536
      xst1 = 0
      ar = "ARCHIVING-NONE"
      artd = "ARCHIVING-DEADBAND-NONE"
      sg[] =
         0,
         2,
      tg[] =
         0,
         198,
         199,
      df = 8482
      ap = 17
      aux = "811461_AUX DG=ab12ab K=7"
      sd = "on"
      rd = "off"
      foreground =
         (Color){
            argb = 4278190080
         }
```

```
         background =
            (Color){
               argb = 16777215
            }
      },
   matchCount = 1
   totalCount = 26
}
```

operatePoints(authstring, pointsNewValues)

Function allowing to set point values, the points need to be in control security group (192). Each change generates an event saved in database so if requent changes are needed publishPoints function should be used

publishPoints(authstring, pointsNewValues)

Function allowing to set point value and keep it valid for a time set in duration attribute

Example:

```
>>> pointval = soapclient.factory.create('PointNewValue')
>>> pointval.id.iess = 'FromWebAPI_a1'
>>> pointval.id.zd = 'WebApi'
>>> pointval.value.av = 4.5
>>> pointval.quality.value = 'QUALITY-GOOD'
>>> pointval.duration.seconds = 3600
>>> soapclient.service.publishPoints(authstring,pointval,val)
```

unpublishPoints(authstring)

Turns off automatic point refresh

Example:

```
>>> pointid = soapclient.factory.create('PointId')
>>> pointid.iess = 'FromWebAPI_a1'
>>> soapclient.service.unpublishPoints(authstring, pointid)
```

getModifiedPoints(authString, startIdx, maxCount, timestamp, dbSequence)

Returns a list of points that have been modified after specified timestamp. However, if input dbSequence is different from database sequence number in EDS92, then all points from EDS92 are returned with disregard of the value of gptTimeStamp parameter. This method can be used to synchronize EDS92 points' database with the client's side. Only users belonging to admin group are authorized to use this method.

Example:

```
>>> modifiedPoints =
soapclient.service.getModifiedPoints(authstring,1)
>>> print(modifiedPoints)
(reply){
   points[] =
      (Point){
         id =
            (PointId){
               sid = 2
               iess = "FI200"
               idcs = "FI200"
               zd = "Simulate"
            }
         rt = "POINT-TYPE-ANALOG"
         value =
            (PointValue){
               av = 1.0
            }
         quality = "QUALITY-GOOD"
         ts =
            (Timestamp){
               second = 1649406276
            }
         lts =
            (Timestamp){
               second = 1649406276
            }
         tss =
            (TimeDuration){
               seconds = 0
            }
         at =
            (Timestamp){
               second = 0
            }
         atss =
            (TimeDuration){
               seconds = 0
            }
         desc = None
         st = 0
         ar = "ARCHIVING-LOCAL"
         artd = "ARCHIVING-DEADBAND-PCT-RANGE"
```

```
            sg[] =
               0,
            tg[] =
               0,
            df = 0
            ap = 17
            aux = None
            un = None
            dp = 2
            ard = 0.0
            tb = 1.0
            bb = 0.0
            hl = 1.0
            ll = 0.0
            foreground =
               (Color){
                  argb = 4278190080
               }
            background =
               (Color){
                  argb = 16777215
               }
         },
      matchCount = 2
      totalCount = 2
      timestamp =
         (Timestamp){
            second = 1648032514
         }
      dbSequence = 1
}
```

getPointsWithCustomFilter(authstring (authstring, source, filterName)

Returns a list of points in EDS database which are filtred using custom filter predefined in file.

Example:

```
>>> pointsWithCustomFilter =
soapclient.service.getPointsWithCustomFilter(authstring,"test",'cust
om_filter')
>>> print(pointsWithCustomFilter)
(reply){
   matchCount = 0
   totalCount = 2
}
```

Substitution values (shade) functions

requestShadesWrite(authstring, newShades)

Function allows to write substitution values for a given point, the value is valid in certain time period. All required data structures can be created with factory functions. The function returns request

number which can be later used in getRequestStatus function to check whether write request was succesfully completed

```
>>> shade = soapclient.factory.create('Shade')
>>> point_id = soapclient.factory.create('PointId')
>>> shade_value = soapclient.factory.create('ShadeValue')
>>> point_value = soapclient.factory.create('PointValue')
>>> period = soapclient.factory.create('TimePeriod')
>>> point_id.iess = 'ANALOG_TEST'
>>> point_value.av = 100
>>> quality = 'QUALITY-GOOD'
>>> getattr(period,'from').second = int(time.time())
>>> period.till.second = int(time.time()) + 7200
>>> shade_value.value = point_value
>>> shade_value.quality = quality
>>> shade_value.period = period
>>> shade.pointId = point_id
>>> shade.values = shade_value
>>> request_id = soapclient.service.requestShadesWrite(authstring,
shade)
```

requestShadesRead(authstring, shadeSelector)

Allows for reading substitution values for given point in provided period of time. Function returns a request id that can be used to retrieve actual response. All required data structures are created with factory functions

```
>>> shade_selector = soapclient.factory.create('ShadeSelector')
>>> point_id = soapclient.factory.create('PointId')
>>> period = soapclient.fatory.create('TimePeriod')
>>> point_id.iess = 'TEST_ANALOG'
>>> getattr(period, 'from').second = 1473422083
>>> period.till.seconds = 1473429301
>>> shade_selector.pointId = point_id
>>> shade_selector.period = period
>>> request_id = soapclient.service.requestShadesRead(authstring,
shade_selector)
```

requestShadesRead(authstring, requestdD)

Retrieves shades read request results. See also requestShadeRead() and getRequestStatus(). If request succedes response get be read with getShades function:

```
>>> soapclient.service.getShades(authstring, request_id)
```

Output will be similar to the one below:

```
[(Shade){
   pointId =
      (PointId){
         sid = 3
         iess = "TEST_ANALOG"
         idcs = "TEST_ANALOG"
         zd = "Simulate"
```

```
      }
   values[] =
      (ShadeValue){
         value =
            (PointValue){
               av = 100.0
            }
         quality = "QUALITY-GOOD"
         period =
            (TimePeriod){
               from =
                  (Timestamp){
                     second = 1473422083
                  }
               till =
                  (Timestamp){
                     second = 1473429301
                  }
            }
      },
 }]
```

requestShadesCopy(authstring, shadesToCopy)

Function allowing to copy substitution values from one point to another. Function returns a request id that can be used to check request status. All required data structures are created with factory functions.

```
>>> shade_copy_request =
soapclient.factory.create('ShadeCopyRequest')

>>> source_point = soapclient.factory.create('PointId')
>>> dst_point = soapclient.factory.create('PointId')
>>> source_point.iess = 'TEST_ANALOG'
>>> dst_point.iess = 'TEST_ANALOG_1'
>>> period = soapclient.fatory.create('TimePeriod')
>>> getattr(period, 'from').second = 1473422083
>>> period.till.seconds = 1473429301
>>> shade_copy_request.srcPointId = source_point
>>> shade_copy_request.dstPointId = dst_point
>>> shade_copy_request.period = period
>>> request_id = soapclient.service.requestShadesCopy(authstring,
shade_copy_request)
```

requestShadesClear(authstring, shadesSelector)

Tells the server to clear points shades for a specific time periods. See also getRequestStatus().

```
>>> shade_copy_request =
soapclient.factory.create('ShadeCopyRequest')

>>> shade_selector = soapclient.factory.create('ShadeSelector')
>>> pointId = soapclient.factory.create('PointId')
>>> period = soapclient.factory.create('TimePeriod')
```

```
>>> pointId.iess = 'FI100'
>>> period.till.second = 1649677327
>>> getattr(period, 'from').second = 1649677317
>>> shade_selector.period = period
>>> shade_selector.pointId = pointId
>>> response =
soapclient.service.requestShadesClear(authstring,shade_selector)
>>> print(response)
657030
```

Tabular functions

getTabular(authstring,requestId)

Retrieves tabular trend request result. See also requestTabular() and getRequestStatus().

```
>>> response = soapclient.service.getTabular(authstring,requestId)
>>> print(response)
(reply){
   pointsIds[] =
      (PointId){
         sid = 1
         iess = "FI100"
         idcs = "FI100"
         zd = "Simulate"
      },
   rows[] =
      (TabularRow){
         ts =
            (Timestamp){
               second = 1649677317
            }
         values[] =
            (TabularValue){
               value = ""
               quality = "QUALITY-NONE"
               tss =
                  (TimeDuration){
                     seconds = 0
                  }
            },
      },
      (TabularRow){
         ts =
            (Timestamp){
               second = 1649677322
            }
         values[] =
            (TabularValue){
               value = ""
```

```
                quality = "QUALITY-NONE"
                tss =
                   (TimeDuration){
                      seconds = 0
                   }
             },
      },
}
```

Trend functions

getTrend(authstring,requestId)

Retrieves graphical trend request result. See also requestTrend() and getRequestStatus().

```
>>> response = soapclient.service.getTrend(authstring,requestId)
>>> print(response)
[(TrendRow){
   pointId =
      (PointId){
         sid = 1
         iess = "FI100"
         idcs = "FI100"
         zd = "Simulate"
      }
   average = ""
}]
```

getTrendGroups(authstring)

Retrieves a list of trend groups migrated to the specified configuration version. If configuration version is unspecified, it defaults to the current version.

```
>>> response = soapclient.service.getTrendGroups(authstring)
>>> print(response)
{}
```

Event functions

requestEvents(authstring, filter, maxCount)

Creates a request to download a list of events that meet given criteria. Authstring and filter parameters are obligatory; filter can be created with factory function described earlier.

Example:

```
>>> filter = soapclient.factory.create('EventFilter')
>>> period = soapclient.factory.create('TimePeriod')
>>> getattr(period,'from').second = time.time() - 86400
>>> period.till.second = time.time()
>>> filter.period = period
```

```
>>> request_id = soapclient.service.requestEvents(authstring,
filter, 100)
>>> print request_id
443472
>>> print soapclient.service.getRequestStatus(authstring,
request_id)
(reply){
   id = 443472
   status = "REQUEST-QUEUED"
   progress = 0.0
   message = "Waiting for execution."
 }
```

getEvents(authstring, request_id)

Function retrieving events returned by a request with request_id

Example:

```
>>> result = soapclient.service.getEvents(authstring, request_id)
>>> print (result[0])
(Event){
   category = "EVENT-CATEGORY-SYSTEM"
   type = "EVENT-SYSTEM-GRANT-LOGIN"
   priority = 4
   message = "TERM USER=admin IP=127.0.0.1:61735"
   pointId =
      (PointId){
         sid = 0
         iess = None
         idcs = None
         zd = None
      }
   floatValue = 0.0
   intValue = 0
   st = 0
   ts =
      (Timestamp){
         second = 1467026933
      }
   tss =
      (TimeDuration){
         seconds = 0
      }
   aux = 622000
   foreground =
      (Color){
         argb = 4278190080
      }
   background =
      (Color){
         argb = 16777215
```

```
        }
 }
```

addEvents(authstring, new_events)

Creates an event in EDS database; new_events parameter is created by soap client factory function.

Example:

```
>>> event = soapclient.factory.create('Event')
>>> event.category = 'EVENT-CATEGORY-CUSTOM'
>>> event.type = 'EVENT-CUSTOM-MESSAGE'
>>> event.ts.second = time.time()
>>> event.message = 'Custom message'
>>> event.floatValue = 0.0
>>> event.intValue = 0
>>> event.tss = 0
>>> soapclient.service.addEvents(authstring, event
```

Object functions

alterObject(authstring, sourceId, sourceName, file, newAttributes)

Function modifies an object in EDS. Source must be specified using either sourceId or sourceName; setting both sourceId and sourceName is an error. If some attribute in newAttributes is not set then it is not changed.

Example:

```
>>> sourceId = 3
>>> sourceName = None
>>> file = "MACRO306.edf"
>>> newAttributes =
soapclient.factory.create('AlterableObjectAttributes')
>>> newAttributes.name = "ObjectName.edf"
>>> rq = soapclient.service.alterObject(authstring, sourceId,
sourceName, file, newAttributes)
```

alterObjectSource

Function modifies a Source in EDS. As above, source must be specified using either sourceId or sourceName; setting both sourceId and sourceName is an error. If some attribute in newAttributes is not set then it is not changed.

Example:

```
>>> sourceId = 3
>>> sourceName = None
>>> newAttributes =
soapclient.factory.create('AlterableObjectSourceAttributes')
>>> newAttributes.desc = "New description of a Source"
>>> rq = soapclient.service.alterObjectSource(authstring, sourceId,
sourceName, newAttributes)
```

getObjectsMetadata

Function gets metadata of objects matching selected criteria. It chooses all objects matching the filter (factory function ObjectFilter), sorts them using the requested order and returns a part of the resulting list defined by startIdx and maxCount. The order parameter should be a semicolon-separated list of ObjectMetadata fields names, for example: "sourceId" or "file;-name". Adding "-" before a field name reverses the order. Full list of fields that may appear in the order parameter: sourceId, file, name, lastModified, sourceName, hasPermission.

Example:

```
>>> filter = soapclient.factory.create('ObjectFilter')
>>> filter.nameRe = "ObjectName.edf"
>>> response = soapclient.service.getObjectsMetadata(authstring,
filter)
>>> print(response)
(reply){
   objects[] =
      (ObjectMetadata){
         file = "MACRO306.edf"
         name = "ObjectName.edf"
         sourceName = "NameOfYourSource"
         sourceId = 3
         lastModified =
            (Timestamp){
               second = 1649325009
            }
         sg[] =
            0,
         tg[] =
            0,
         md5 = "9005c5a7a3361ddd1b9671246ed088da"
         hasPermission = True
      },
   matchCount = 1
   totalCount = 4
}
```

getObjectsSources

Returns a list of all object sources if filter is not set or all object sources matching filter. Note: point sources and object sources might be completely different, even though in many cases they have similar names. Use getPointSources() to get a list of point sources. The order parameter should be a semicolon-separated list of ObjectSource field names, in example: "id" or "kind;-name". Adding "-" before a field name reverses the order. Full list of fields that may appear in the order parameter: id, name, desc, kind, host, prefix, suffix, hasPermission.

Example:

```
>>> filter = soapclient.factory.create('ObjectSourceFilter')
>>> filter.nameRe = "NameOfYourSource"
>>> response = soapclient.service.getObjectsSources(authstring,
filter)
>>> print(response)
(reply){
   sources[] =
```

```
      (ObjectSource){
         id = 3
         name = "NameOfYourSource"
         desc = "New description of a Source"
         sg[] =
            0,
         tg[] =
            0,
         kind = "OBJECT-SOURCE-KIND-DISK"
         host = "localhost"
         prefix = None
         suffix = None
         options = 1
         hasPermission = True
      },
   matchCount = 1
   totalCount = 2
}
```

Diagram functions

openDiagram

Function opens the diagram and returns url to the diagram. Returned url is used in other Diagram functions.

Example:

```
>>> source = "bloki_001"
>>> file = "10003.edf"
>>> refreshRate = None
>>> previousUrl = None
>>> pointGroup = None
>>> httpUrl = "http://127.0.0.1:44090/"
>>> url = soapclient.service.openDiagram(authstring, source, file,
refreshRate, previousUrl, pointGroup, httpUrl)
>>> print(url)
http://127.0.0.1:44090/27434b15-b147-467b-87ec-2d950a7bb7f8/
```

closeDiagram

Function closes the opened diagram.

Example:

```
>>> rq = soapclient.service.closeDiagram(authstring, url)
```

openWindowDiagram

Function opens window diagram of opened diagram.

Example:

```
>>> source = "bloki_001"
>>> file = "10003.edf"
>>>
```

```
>>> refreshRate = None
>>> previousUrl = None
>>> pointGroup = None
>>> httpUrl = "http://127.0.0.1:44090/"
>>>
>>> url = soapclient.service.openDiagram(authstring, source, file,
refreshRate, previousUrl, pointGroup, httpUrl)
>>> print(url)
http://127.0.0.1:44090/b2853603-4a90-47be-b390-b85d68255189/
>>>
>>> rq = soapclient.service.openWindowDiagram(authstring, url,
source, file)
```

getGfxSrvFeatures

Returns features supported by connected graphics server. It uses url of the opened diagram.

Example:

```
>>> rq = soapclient.service.getGfxSrvFeatures(authstring, url)
>>> print(rq)
(reply){
   tagQueryKeepAlive = True
   canSuspendSession = True
   supportsFeatureNegotiation = True
   supportsSettingEntryField = True
   supportsMultiWindowDiagrams = True
   supportsOpenWindowDiagram = True
}
```

navigateDiagram

Function uses navigation of the opened diagram.

Possible navigations: [string] - enum { 'DIAGRAM-NAVIGATE-HOME', 'DIAGRAM-NAVIGATE-PREVIOUS', 'DIAGRAM-NAVIGATE-NEXT', 'DIAGRAM-NAVIGATE-UP', 'DIAGRAM-NAVIGATE-DOWN', 'DIAGRAM-NAVIGATE-LEFT', 'DIAGRAM-NAVIGATE-RIGHT' }

Example:

```
>>> navigation = 'DIAGRAM-NAVIGATE-HOME'
>>> rq = soapclient.service.navigateDiagram(authstring, url,
navigation)
```

handleDiagramClick

Fuction simulates click on the active area of the diagram.

Example:

```
>>> role = 'DIAGRAM-ROLE-MAIN'
>>> areaId = "1520852743648"
>>> rq = soapclient.service.handleDiagramClick(authstring, url,
role, areaId)
```

lockWindowDiagram

Function locks diagram opened in the window. It uses url of the opened main diagram, windowId of the diagram opened in the window it has to lock/unlock.

Example:

```
>>> locked = 1
>>> rq = soapclient.service.lockWindowDiagram(authstring, url,
window['id'], locked)
```

SetActiveWindowDiagram

Function sets as active diagram opened in the window. It uses url of the opened main diagram and windowId of the diagram opened in the window.

Example:

```
>>> rq = soapclient.service.setActiveWindowDiagram(authstring, url,
window['id'])
>>> print(rq)
```

resizeDiagram

Function resizes the opened diagram. It uses factory function DiagramResolution.

Example:

```
>>> role = 'DIAGRAM-ROLE-MAIN'
>>> newResolution = soapclient.factory.create('DiagramResolution')
>>> newResolution.width = 1920
>>> newResolution.height = 1080
>>>
>>> rq = soapclient.service.resizeDiagram(authstring, url, role,
newResolution)
```

setDiagramEntryFieldValue

Function sets entry field value for the opened diagram.

Example:

```
>>> role = 'DIAGRAM-ROLE-MAIN'
>>> areaId = "1520852743648"
>>> value = "1"
>>>
>>> rq = soapclient.service.setDiagramEntryFieldValue(authstring,
url, role, areaId, value)
```

setDiagramViewport

Function sets viewport for the opened diagram. It uses factory functions DiagramArea and DiagramPosition

Example:

```
>>> role = 'DIAGRAM-ROLE-MAIN'
>>> viewport = soapclient.factory.create('DiagramArea')
>>> viewport.topLeft = soapclient.factory.create('DiagramPosition')
>>> viewport.topLeft.x = 0.1
>>> viewport.topLeft.y = 0.1
>>> viewport.bottomRight =
soapclient.factory.create('DiagramPosition')
>>> viewport.bottomRight.x = 0.5
>>> viewport.bottomRight.y = 0.5
>>>
>>> rq = soapclient.service.setDiagramEntryFieldValue(authstring,
url, role, viewport)
```

Reports functions

CreateReportConfig

Function creates report config for uploaded report template. It uses factory functions ReportCofigDefinition and TimeDuration.

Example:

```
>>> sourceId = 3
>>> sourceName = None
>>> rdfFileName = "terminal1.rdf"
>>>
>>> config = soapclient.factory.create('ReportConfigDefinition')
>>> config.referenceTimeShift = ""
>>> config.runDelay = soapclient.factory.create('TimeDuration')
>>> config.runDelay.seconds = 0
>>> config.timeMaskExpression = ""
>>> config.eventsExpression = ""
>>> config.inputValues = ""
>>> config.outputMaskFileHtml =
'C:/Users/User1/Desktop/Raporty/fromapi.pdf'
>>> rq = soapclient.service.createReportConfig(authstring, sourceId,
sourceName, rdfFileName, config)
```

getReportsConfigs

Functions allows querying EDS92 for reports configs matching selected criteria. It uses factory function ReportConfigFilter which contains ObjectFilter factory function.

Example:

```
>>> filter = soapclient.factory.create('ReportConfigFilter')
>>> filter.objectFilter = soapclient.factory.create('ObjectFilter')
>>> filter.objectFilter.sourceId = 3
>>>
>>> rq = soapclient.service.getReportsConfigs(authstring, filter)
>>> print(rq)
(reply){
   configs[] =
      (ReportConfig){
         id = 1
```

```
         reportDefinitionSource = "NameOfYourSource"
         reportDefinitionFile = "terminal1.rdf"
         referenceTimeShift = None
         runDelay =
            (TimeDuration){
               seconds = 0
            }
         timeMaskExpression = None
         eventsExpression = None
         inputValues = None
         outputMaskFileHtml =
"C:/Users/User1/Desktop/Raporty/fromapi.pdf"
      },
   matchCount = 1
   totalCount = 1
}
```

deleteReportConfig

Function deletes global report configuration

Example:

```
>>> configId = 3
>>> rq = soapclient.service.deleteReportConfig(authstring,configId)
```

alterReportConfig

Function modifies existing report configuration. It uses factory function ReportConfigDefinition.

Example:

```
>>> configId = 4
>>> config = soapclient.factory.create('ReportConfigDefinition')
>>> config.referenceTimeShift = ""
>>> config.runDelay = soapclient.factory.create('TimeDuration')
>>> config.runDelay.seconds = 0
>>> config.timeMaskExpression = ""
>>> config.eventsExpression = ""
>>> config.inputValues = ""
>>> config.outputMaskFileHtml =
'C:/Users/User1/Desktop/Raporty/fromapiALTERED.pdf'
>>> rq = soapclient.service.alterReportConfig(authstring, configId,
config)
```

createGlobalReport.py

Function creates global report file. It uses factory functions: ReportDefinition, ReportDefinitionRow, ReportDefinitionCell

Example:

```
>>> sourceId = 3
>>> sourceName = None
>>> file = "globalReportFile"
```

```
>>> name = "globalReportName"
>>> rdf = soapclient.factory.create('ReportDefinition')
>>> rdf.rows = soapclient.factory.create('ReportDefinitionRow')
>>> rdf.rows.cells =
soapclient.factory.create('ReportDefinitionCell')
>>> rdf.rows.cells.content = "CONTENT"
>>>
>>> rq = soapclient.service.createGlobalReport(authstring, sourceId,
sourceName, file, name, rdf)
```

requestCustomReport

Function executes custom report and return requestId. It uses factory functions: CustomReportRequest, ReportDefinition, Timestamp ReportValue andReportArgument

Example:

```
>>> request = soapclient.factory.create('CustomReportRequest')
>>> request.rdf = soapclient.factory.create('ReportDefinition')
>>> request.rdf.rows =
soapclient.factory.create('ReportDefinitionRow')
>>> request.rdf.rows.cells =
soapclient.factory.create('ReportDefinitionCell')
>>> request.rdf.rows.cells.content = "CONTENT"
>>> request.dtRef = soapclient.factory.create('Timestamp')
>>> request.dtRef.second = 90061
>>> request.args = soapclient.factory.create('ReportArgument')
>>> request.args.name = "argName"
>>> request.args.value = soapclient.factory.create('ReportValue')
>>> request.args.value.string = "value"
>>>
>>>
>>> rq = soapclient.service.requestCustomReport(authstring, request)
>>> print(rq)
305602
```

getCustomReport

Function returns report result from requested custom report.

Example:

```
>>> run = soapclient.service.getCustomReport(authstring, requestId)
>>> print(run)
[(ReportResultRow){
   cells[] =
      "CONTENT",
}]
```

RequestGlobalReport

Function executes global report and returns requestId. It uses factory functions: GlobalReportRequests, Timestamp, ReportArgument, ReportValue.

Example:

```
>>> request = soapclient.factory.create('GlobalReportRequest')
>>> request.reportConfigId = 4
>>> request.dtRef = soapclient.factory.create('Timestamp')
>>> request.dtRef.second = 90061
>>> request.args = soapclient.factory.create('ReportArgument')
>>> request.args.name = "argName"
>>> request.args.value = soapclient.factory.create('ReportValue')
>>> request.args.value.string = "value"
>>>
>>>
>>> rq = soapclient.service.requestGlobalReport(authstring, request)
>>> print(rq)
59263
```