



Arlinger John

5 February 2021

Minutes StanForD

2021-02-05

Place:	Microsoft Teams (Uppsala), 2021-02-05, 10:00-15:00 CET
Participants:	John Arlinger, Secretary Johan J. Möller, Chairman Tapio Räsänen, Metsäteho Timo Melkas, Metsäteho Sorsa Juha-Antti, Metsäteho Pasi Palonen, Parker Juha Korhonen, Technion Lars-Erik Andersson, Dasa Jonas Thorsson, Dasa Henrik Persson, Logmax Antti Korkiakoski, JohnDeere Erik Kindlund, JohnDeere Niclas Eriksson, Komatsu Jacek Nawrot, Tigercat Josh Hildebrand, Tigercat Heikki Komulainen, Ponsse Janne Pursiainen, Ponsse Asko Leivonmäki, Exertus (Logset) Ian Wilson, ForestPHD Jeremy Gibson, ForestPHD Andreas Barth, CGI Per Knutsson, CGI Lars Henriksson, Biometria Mike O'Shea, Coillte Ronan O'Keeffe, Coillte Jonas Gustafsson, Sveaskog Jean Plamondon, FPInnovation

Decision summary:

- It was decided to publish the new specification for StanForD REST API 1.0. Observe that this specification has a versioning separate from the main standard documents and it can be updated independently of other standard documents (including schemas).
- It was decided to publishing new yarder and skidder schemas (soi/spr/yoi/yp) within version 3.6.

Decisions major version 4.0

The following decisions will affect the version 4.0 pre-release schemas:

- It was decided add new simple type to be used for all UserIDs. It was decided to look into other restrictions, e.g. tilde/non-latin/non-ASCII/non-unicode until next meeting.
- It was decided to add a new simple type for LogLengths to include a single fractional digit.
- It was decided that the xml data type “xsd:unsignedLong” is to be used for all keys (from 0 up to 18446744073709551615).
- It was decided to reject the proposal for new Tracking structure and to keep the present layout. The secretary will analyse how to handle Tracking in order to do schema specific adjustments (e.g. fpr vs mom as illustrated in proposal).
- It was decided to update the diameter vector.
- It was decided to keep StemType structures in pin and spi as it is without any changes. It was pointed out that this must be handled as an optional elements in all control systems.
- It was decided to update the StemCode structures for both stem and species group. Also decided that the secretary will clarify rules for handling sticky stem codes.
- It was decided to modify the ogi schema including use of a GeoJson string inside element MultiPoint/MultiLine/MultiPolygonGeoJson. It was decided to add a new schema called DesignTemplate.
- It was decided to exclude combined times and frequency short down times from mom and to develop a new schema for distributing user defined codes for down times and other work.
- Remaining issues on agenda postponed to next meeting.
- It was decided to
 - exclude combined times and frequency short down times in mom.
 - develop a new schema for distributing user defined down time codes

- exclude ShiftKey from the IndividualMachineWorkTime and to discuss how Shift, OperatorWorkTime, Login and MachineWorkTime are related to each other at the next meeting.

Action list (not updated since last meeting):

- The following guidelines should be followed when implementing LongTimberDefinition:
 - Must be possible to view and edit LongTimberDefinition in administrative tools.
 - Default value equal to DiameterMAXButt in administrative tools
 - LongLogButtMAX is never to be included in pin-file if value is 0 (it is optional).
 - Value 0 (zero) is to be interpreted as null.
- It is recommended to implement stem codes for both individual trees and groups of trees (so called “sticky registration of stem codes”). Whether the “sticky registration” is trigger by double click or any other operator interaction does not matter.
- Do not mix products and deliveries in forwarder applications.
 - Focus on Deliveries in the UI.
 - Avoid using pin-files. It often creates a lot of confusion.
- Map interface when creating Locations would be good to avoid incorrect coordinates.
- TextToMachine an urgent variable to be shown when creating new object in harvester and forwarder.
- It is strongly recommended that the control system does not automatically activate a modified pin if it is not active at the present harvesting object
- All randomly selected stems (both rejected and not rejected) must be saved in an hqc-file including a possible reason for rejection. This file should be possible to send directly to the forest company (in Sweden to Biometria).
- Coordinates in production files should always be activated by default in all forest machines (especially important after software updates when coordinate registration is commonly lost).

§1 Opening of meeting and agenda

Chairman Johan J Möller opened the meeting.

No “other issues” were raised.

§2 Presentation of participants

Presentation of meeting participants including a short weather report from around the world!

Pasi Palonen informed the meeting that this was his last meeting since Parker is discontinuing production of the Motomit controller.

§3 Information about decisions and discussions at previous meetings

Data types updated:

- tableName attribute in DataTableType
- columnOrder attribute in ColumnDataType
- DeliveryPresentationOrder element in CommonDeliveryDefinitionType
- LoadNumberOfItems in LoadNumberOfItemsType
- LongLogButtMAX in LongTimberDefinitionType
- TrackingKey element in TrackCoordinatesType
- ConstantA and FactorB from in SwedishZaccoType

Other updates included in 4.0 pre-release:

- Removed optional attribute agency from UserIDType.
- Made ContactInformation element optional in LoggingOrganisationType
- Made countryCodeCategory attribute mandatory in CountryCodeType
- Added a new mandatory attribute modificationRestricted:
 - EstonianVolumeParametersType
 - ProductUserIDs, DeliveryUserIDsType and LocationUserIDsType
- Modifications of DiameterSection structure
- Spelling errors corrected (ShiftKey and Object filter)
- Updates of bq
 - RandomMINLength and RandomLogNotification excluded
 - LengthCalibration removed

Any changes that makes it problematic to use 3.1 instruction and 4.0 hpr?

- Non-defined datatypes now e.g. integers
- DiameterSection moved in oin!
- New restrictions on UserIDs
- Please look for more problematic areas!

§4 StanForD API. New communication methods

It was decided to publish the new specification for StanForD REST API 1.0 as enclosed in appendix 1. Observe that this specification can be updated independently of other standard documents (including schemas).

§5 New skidder and yarder schemas in version 3.6

It was decided to publish new yarder and skidder schemas (soi/spr/yoi/ypr) within version 3.6.

The proposal for new skidder and yarder messages including instructions and production is enclosed in appendix 2.

§6 Proposals new major version 4.0

NEW SIMPLE DATA TYPES

It was discussed at last meeting to study the possibility of implementing new simple types for handling commonly register data categories such as Lengths, Diameters, Volumes, Keys (numerical), PresentationOrders. Focus has so far been on log lengths, UserIds and Keys.

It was decided that the secretary will look for other elements/attributes used repeatably for the same type of measurements, like volumes per log, volume per load, volume per stem etc. Having common simple types simplifies maintaining the standard in the future. Will be discussed at next meeting.

Format of UserIDs

Heikki Komulainen, Ponsse (2019-08-20)

Ponsse have had several problems with UserID format over the last few years. The SF2010 implementation is database based and some of these issues are associated with reserved characters of database, some of these are our bugs, and there is also a few times when incoming message contains “invalid” userID. Some examples:

- Whitespace characters before or after the actual userID value
 - Our systems removes the whitespaces
- UserID value contains only whitespaces (one or more)
 - → Our system removes whitespaces, so result is empty string
 - In UI this value looks empty, even we do not remove the whitespace
- UserID contains database reserved characters like percentage or underscore.
 - These characters affects problems when doing database level comparison of two userIDs
- Character casing (uppercase/lowercase) issues.
 - This was our own bug, SF2010 already defines that userid comparison is case intensive.

Of course, all of these issues can be handled in a way or another in code, but some other manufacturer may do it in different way so in long run it is much better if SF2010 schemas contains some more accurate constraints.

Ponsse did not have any proposal for new constraints, but at least length of the userid and allowed/prohibited characters would be good to define.

It was decided to add a new simple type (created by Juha-Antti Sorsa) to forbid the characters: % ? < > : " ' / \ | * + and only whitespace character that is allowed is one space between valid substrings. No length restrictions are included.

It was decided that the following type is to be used for all UserIDs:

```
<xs:simpleType name="RestrictedStringType">
  <xs:restriction base="xs:string">
    <xs:pattern value="([\S-[%?&lt;&gt;;&quot;&apos;/\|*+]])+
      ( ([\S-[%?&lt;&gt;;&quot;&apos;/\|*+]])+)*"/>
  </xs:restriction>
</xs:simpleType>
```

It was also decided to look into other restrictions, e.g. tilde/non-latin/non-ASCII/non-unicode until next meeting

LogLength

Mikael Eliasson, LogMax 2016-04-07

LogMax noted when studying length measurements that it would be an advantage to be able to adjust the calibration somewhere between 0 and +1 cm or between 0 and -1 cm. Since all length measurements are registered as an integer with unit cm (without any decimals) it is difficult to do a "perfect" adjustment.

It was suggested to update the definition LogLength in LogMeasurementHarvestingQualityControlType and LogMeasurementType to allow for decimals (nonNegativeDecimal). This means that log lengths in hpr and hqc are registered in the same way.

It was decided to this new simple type for LogLengths as illustrated below:

```
<xs:simpleType name="NonNegativeDecimal">
  <xs:restriction base="xsd:decimal">
    <xas:minInclusive value="0"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="singleDigitNonNegativeDecimal">
  <xs:restriction base="NonNegativeDecimal">
    <xs:fractionDigits value="1" fixed="true"/>
  </xs:restriction>
</xs:simpleType>
```

Observe the simple type above allows for registering also a non negative integer.

StemKeys

Per Knutsson, CGI, 2020-09-08

StemKey has data type xsd:PositiveInteger. There is no limitation on the size of this datatype. However, databases and development tools normally have a limitation on the size of integers.

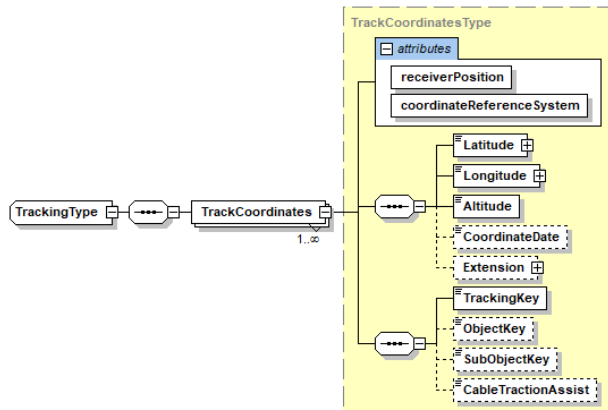
In some cases the StemKeys implemented in some controllers has very large values. In order to avoid having uncertainty about whether a specific data type is large enough it was suggested that Keys are given a maximum value e.g., comparable with int64 or bigint (up to 9223372036854775807).

It was decided that the following xml data type is to be used “xsd:unsignedLong” (from 0 up to 18446744073709551615)

TRACKING

Niclas Eriksson, Komatsu, 2017-11-29, Henrik Persson, LogMax, 2017-11-30 and 2019-02-28.

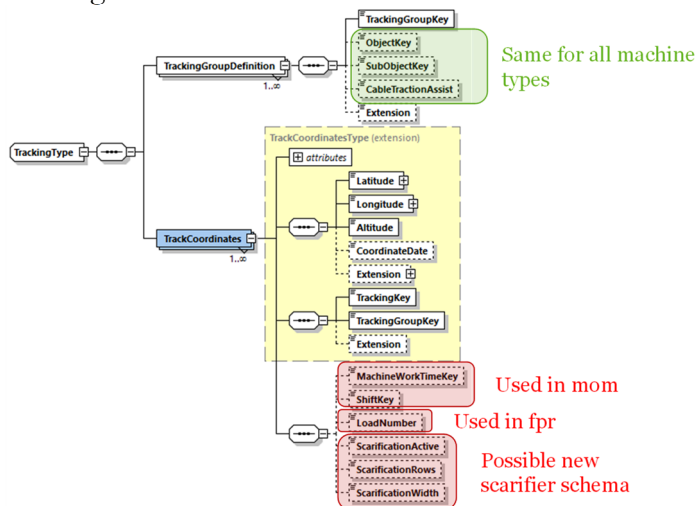
The present Tracking structure is illustrated below:



Repeating e.g. ObjectKey for each position means a significant redundancy. It was suggested that we study the possibility of introducing a new TrackingGroupDefinition with e.g. elements TrackingSettingKeys, ObjectKey, SubObjectKey, TrackingDescription, ShiftKey, IndividualMachineWorkTimeKey and Extension.

Observe that IndividualMachineWorkTimeKey does not exist today.

Figure below illustrates a suggested new structure for registering additional tracking data:



Some of the new elements are registered for each tracking position in the example above (e.g. ShiftKey, MachineWorkTimeKey, LoadNumber, ActiveScarification) and thus not included in the TrackingGroupDefinition since they are they are used only in certain schemas and they are frequently updated.

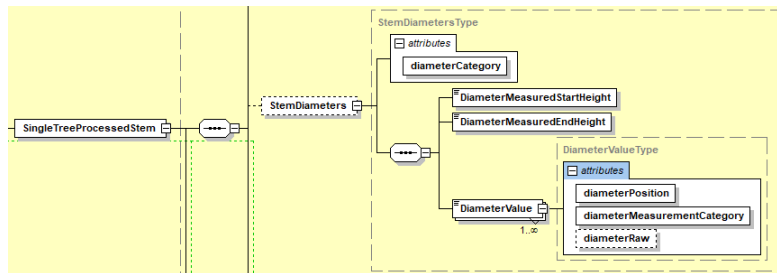
Comments:

- MachineWorkTimeKey to be added.
- ScarificationActive, ScarificationRows, ScarificationWidth does not exist today but there has been some informal discussions regarding introduction of scarifiers recently.
- LoadNumber registered after a load has been registered in the fpr.

It was decided to reject the new suggested structure and to keep the present layout. The secretary will analyse how to handle Tracking in order to do schema specific adjustments (e.g. fpr vs mom as illustrated in proposal above.

DIAMETERVECTOR ELEMENTS

Present structure for diameter vector in hpr is illustrated below:



Shortened element and attribut names

Fredrik Tjernberg, Komatsu, 2015-09-07

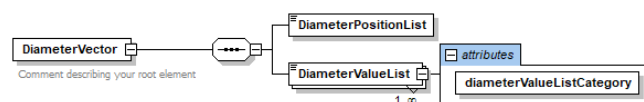
We could minimize the size of StemDiameter-vector by modifying element and attribute names according to the following list:

- DiameterValue = d
- diameterPosition = dp
- diameterMeasurementCategory = dc
- diameterRaw = dr

Observe that the decrease in file size on a zipped file is significantly smaller than the effect on an uncompressed file.

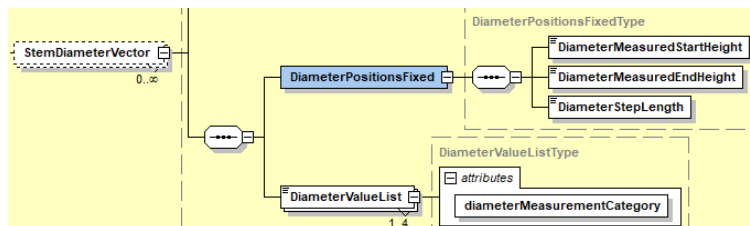
List element for diameter vector

Another solution (John Arlinger and Juha-Antti Sorsa, 2020-09-01) is to implement a list element as illustrated below. The individual values in the list are separated by a “space”. Observe that diameterValueListCategory includes the following enumeration values: Average, First, Second and Raw.



DiameterVector	
xsi:noNamespaceSchemaLocation	TestSchema_DiameterVector_v2.xsd
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
DiameterPositionList	0 10 20 30 40 50 60 70
DiameterValueList	
diameterValueListCategory	Average
Text	100 99 96 96 95 89 75 74

Another, even more compact, solution could be to exclude the position list and replace the position list with a start and end height as well as a “step length” according to the following structure:



StemDiameterVector	
diameterCategory	Over bark
DiameterPositionsFixed	
DiameterMeasuredStartHeight	0
DiameterMeasuredEndHeight	70
DiameterStepLength	10
DiameterValueList	
diameterMeasurementCategory	Average
Text	100 99 96 96 95 89 75 74

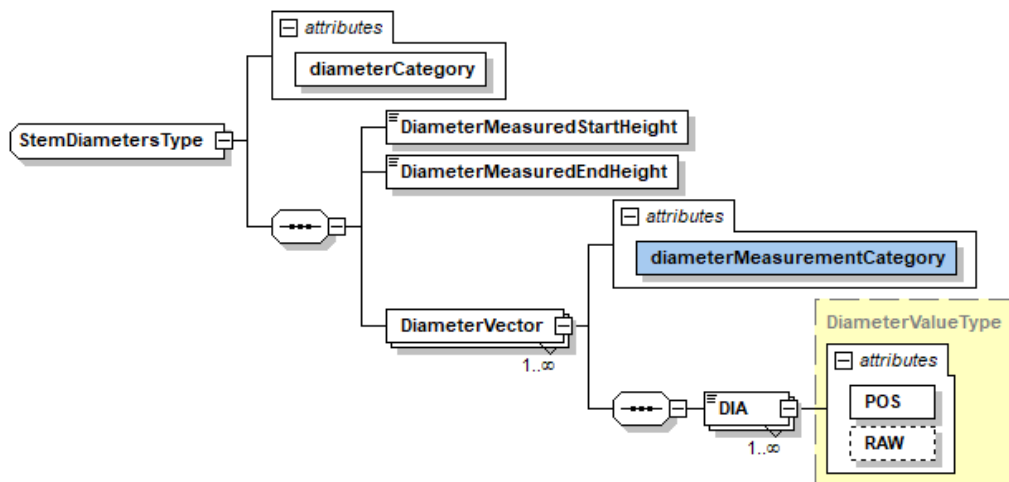
Henrik Persson stated at last meeting that using a list-construction means a step away from the basic xml-principles where, e.g. xslts (stylesheets) is difficult to use. He stated that he would prefer the solution suggested by Fredrik Tjernberg.

Jonas Thorsson and John Arlinger pointed out that the list structure decreases the file sizes and improves parsing speed.

Shortened names and new diameter vector structure

Additional proposal, Henrik Persson, 2020-12-11.

Create a new repeatable element DiameterVector and move attribute diameterMeasurementCategory up to this level from the old element DiameterValue which is renamed to DIA as illustrated below.



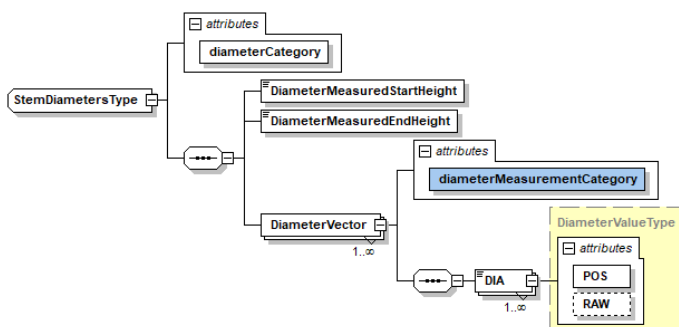
Analysis of file sizes and parsing time

Below is an hpr example file including 849 stems describing file size of raw files as well as time consumption for parsing stems using a SAX-parser developed using Delphi (Pascal).

☐ Name	Size ^	Time to parse (SAX)
✗ NoDiaVector.hpr	1 227 KB	
✗ ListSpecial_DiameterVector.hpr	1 696 KB	200 ms
✗ ShortenedName_NewDiaVector....	3 818 KB	600 ms
✗ ShortenedName_DiaVector.hpr	6 360 KB	740 ms
✗ Standard_DiaVector.hpr	12 740 KB	950 ms

The last alternative (“ShortenedName_NewDiaVector”) where names are shortened and diameterMeasurementCategory is not repeated for each diameter seems to be a good compromise and was therefore recommended by the secretary. This solution means that we decrease file size, increase speed of parsing but we keep style sheet possibilities.

It was decided to use the following new structure for diameter vectors:

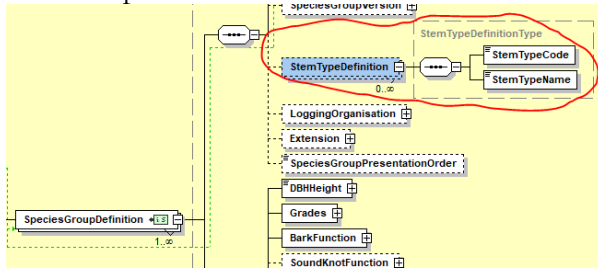


STEMTYPECODE

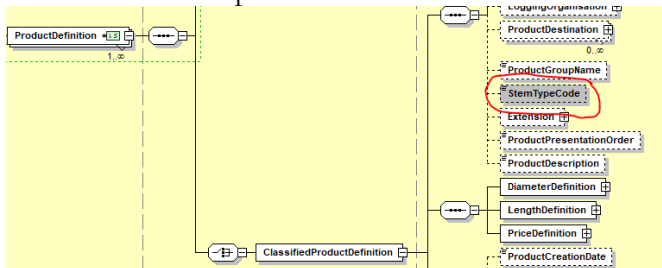
John Arlinger and Peter Sundlöf, Ponsse, 2019-02-01.

Structure of StemType probably should be changed since there is no referencing between spi and pin as in most other similar cases.

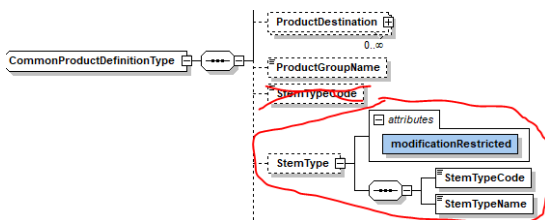
Present Spi:



Present Pin and Hpr:



It was suggested that StemTypeCode and StemTypeName is included in the pin and totally excluded from the StemTypeDefinition in the spi.



The following main comments were presented by Metsäteho:

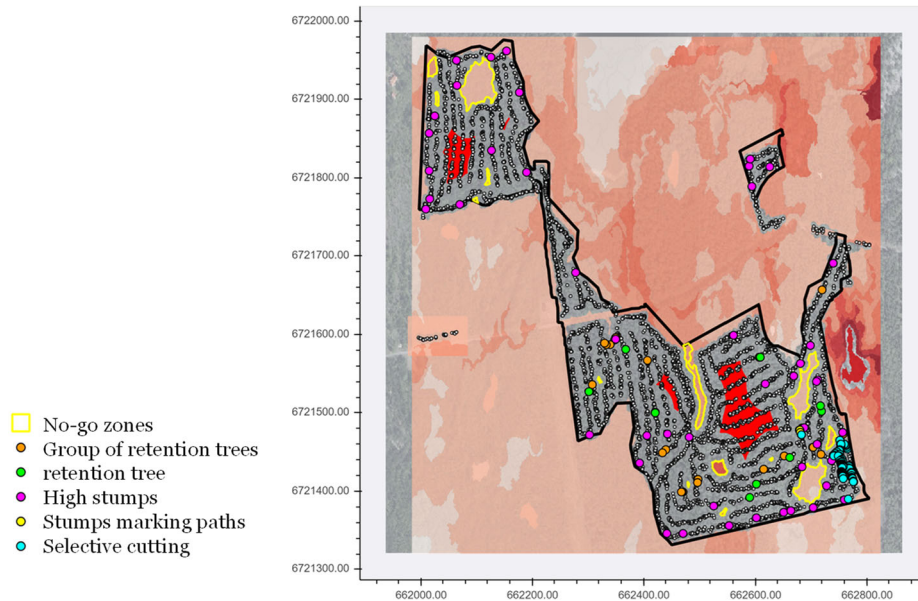
- Is this any problem today? No
- These codes are widely used in Finland and any changes mean significant amount of work and risks.

Based on Finnish comments it was decided to keep the structure as it is, without any changes. It was pointed out that this must be handled as an optional elements in all control systems.

STEMCODE

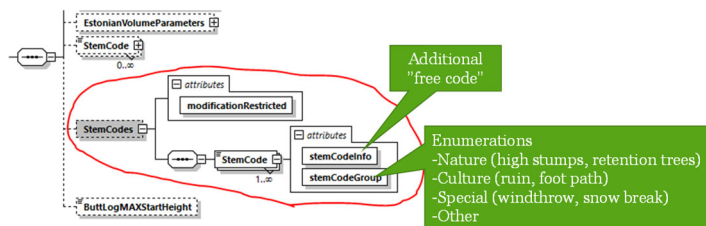
John Arlinger and Johan Möller 2020-09-03. Niclas Eriksson, Komatsu, 2020-10-23.

There is an increasing interest in using StemCodes in Sweden as illustrated below:



The StemCode is today very simple. The use could probably be improved by adding some additional identity attributes, both for control system and for receiving system

It was decided to add the following structure to the SpeciesGroupDefinition:

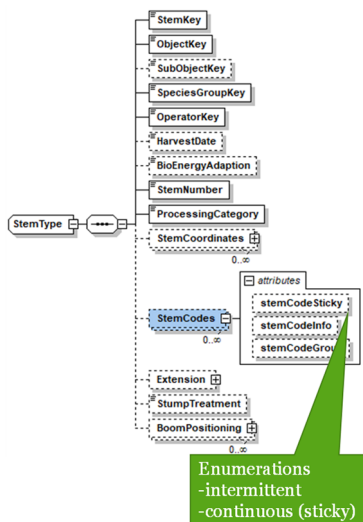


The standardized enumeration values in attribute stemCodeGroup:

- Nature (high stumps, retention trees etc)
- Culture (ruin, foot path etc)
- Special harvesting (selective cutting, windthrow, snow breakage etc)
- Other

Another extension could be to add boolean attribute “stemCodeSticky” to element StemCode in hpr. The value “true” is to be used when “sticky function” is used. Sticky function should be implemented so that it is in use until operator de-activates the function. Should in other words not be de-activated automatically if species is changed or if a single (“non-sticky”) stem code is registered.

It was decided to add the following structure to the StemType:



Hpr-example illustrating how to use the new attributes:

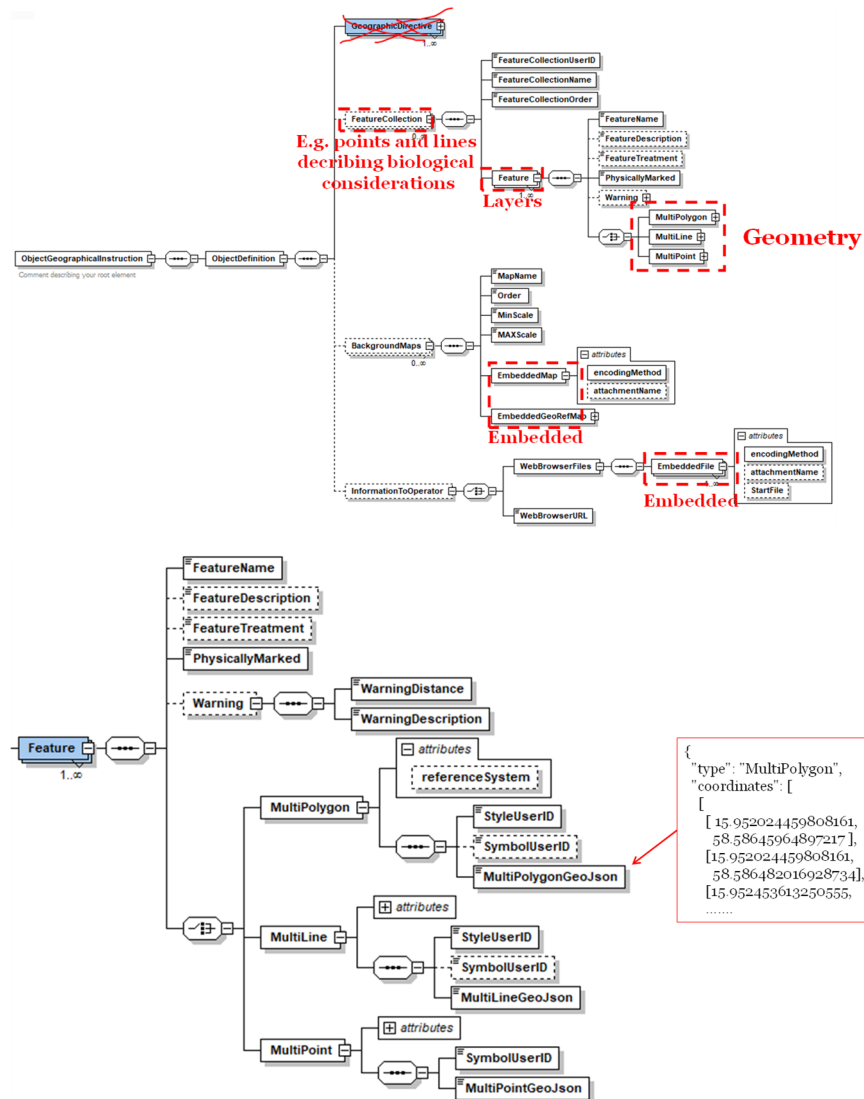
ProcessingCategory	StemCoordinates	StemCode
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 3214 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 3214 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 3214 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 3214 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (2)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
		2 false XYZ Nature Retention tree
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 true 7 Special Windthrow
SingleTreeProcessing	StemCoordinates	StemCode (1)
		stemCodeSticky stemCodeInfo stemCodeGroup Rbc Text
		1 false ABD Nature High stump

Decided that the secretary will clarify rules for handling sticky stem codes.

SIGNIFICANT RE-STRUCTURING OF OGI-SCHEMA

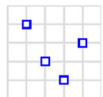
Secretary gave a brief presentation of proposal.

It was decided to implement the following new structure in the ogi:



It was decided to use a GeoJson string inside element MultiPoint/MultiLine/MultiPolygonGeoJson as illustrated below:

MultiPoint:

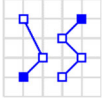


```

{
  "type": "MultiPoint",
  "coordinates": [
    [10, 40], [40, 30], [20, 20], [30, 10]
  ]
}

```

MultiLine:



```
{
  "type": "MultiLineString",
  "coordinates": [
    [[10, 10], [20, 20], [10, 40]],
    [[40, 40], [30, 30], [40, 20], [30, 10]]
  ]
}
```

MultiPolygon;

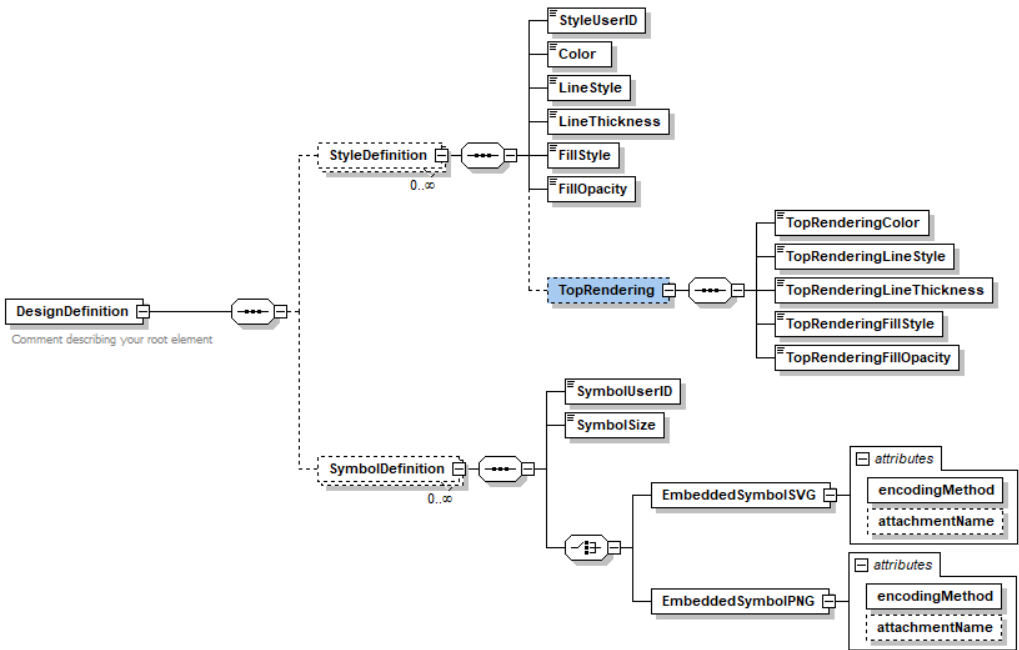


```
{
  "type": "MultiPolygon",
  "coordinates": [
    [
      [[40, 40], [20, 45], [45, 30], [40, 40]]
    ],
    [
      [[20, 35], [10, 30], [10, 10], [30, 5], [45, 20], [20, 35]],
      [[30, 20], [20, 15], [20, 25], [30, 20]]
    ]
  ]
}
```

Another fundamental change that was suggested is to move to formatting (designing) to a separate Design message and to utilize StyleUserID and SymbolUserID as a reference between the structure above and the Design message. The Design structure is illustrated below. Observe that a new parameter FillOpacity has been added when comparing with the old standard. Symbols (svg

or png) can be included inside the message using element EmbeddedSymbolSVG/PNG using encodingMethod Base64.

It was decided to add a new schema called DesignTemplate according to this structure:



The following figure illustrates how individual features are connected to specific symbol or style in the DesignDefinition.

Element	Feature 1	Feature 2	Feature 3
Name (label)	Main landing	Cutting	Route 66
Treatment	Both sides of road	Clear fell	
Description		Pine forest	
GeometryType	Point	Polygon	Line
Geometry	(30 10)	((30 10, 40 40, 20 40, 10 20, 30 10))	(30 10, 10 30, 40 40)
Order	1	2	3
DesignCategoryUserID		CuttingArea	RoadLine
SymbolUserID	Landing		Road
PresentationGroupUserID	Logging	Logging	Roads

DesignCategory-UserID	Line-Color	Line-Style	Line-Thickness
CuttingArea	Blue	Solid	3
StripRoadLine	Brown	Dot	1
RoadLine	Black	Solid	2
PowerLine	Red	Dash	1

SymbolUserID	Symbol	Symbol-Size
NoLogging		5
Landing		5
Road		4

The intention is that Skogforsk will develop a default Design definition that can be used by any actor.

SIGNIFICANT RE-STRUCTURING OF MOM-SCHEMA

A number of problems caused by the present structure of the mom schema has been noted since the introduction of StanForD2010. The main problematic areas are:

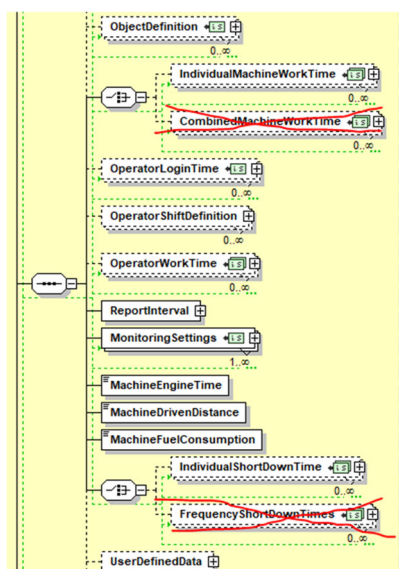
- Two main dialects exist since work times can be registered using IndividualMachineWorkTimes OR CombinedMachineWorkTimes.

- Complex user interface since a very large number of reasons for down times and other work has been standardized in “StanForD2010CodeList_V*p*.xsd”
- Market specific code requirements are difficult (takes a long time) to implement since all codes must be added to “StanForD2010CodeList_V*p*.xsd”.

Combined times excluded

Excluding “combined” times from the mom schema would make the standard more consistent and simpler to use.

It is thus suggested that CombinedWorkTime and FrequencyShortDownTimes are excluded from the mom schema:



New enumeration “WinchAnchoring”

Proposal from Heikki Komulainen, Ponsse and discussed in the Finnish StanForD group meeting 2020-10-29.

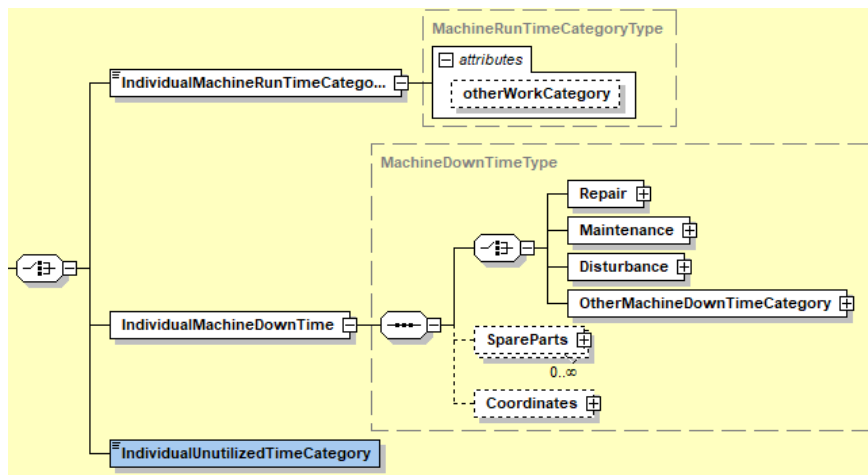
There is a market need to report the downtime caused by winch anchoring and releasing in mom with a specific code:

- to be used both for harvesting and forwarding
- harvesting at slopes is more common in many parts of the world and winch work is an essential part of the daily operations

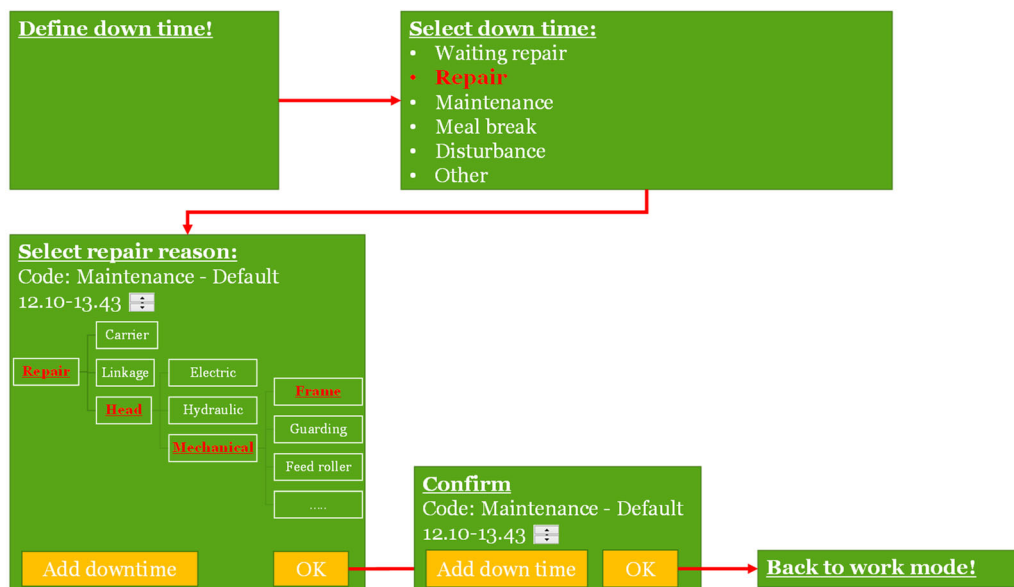
It was suggested that value “Winch anchoring” is added to OtherMachineDownTimeCodeCategory.

Simplified down time reasons

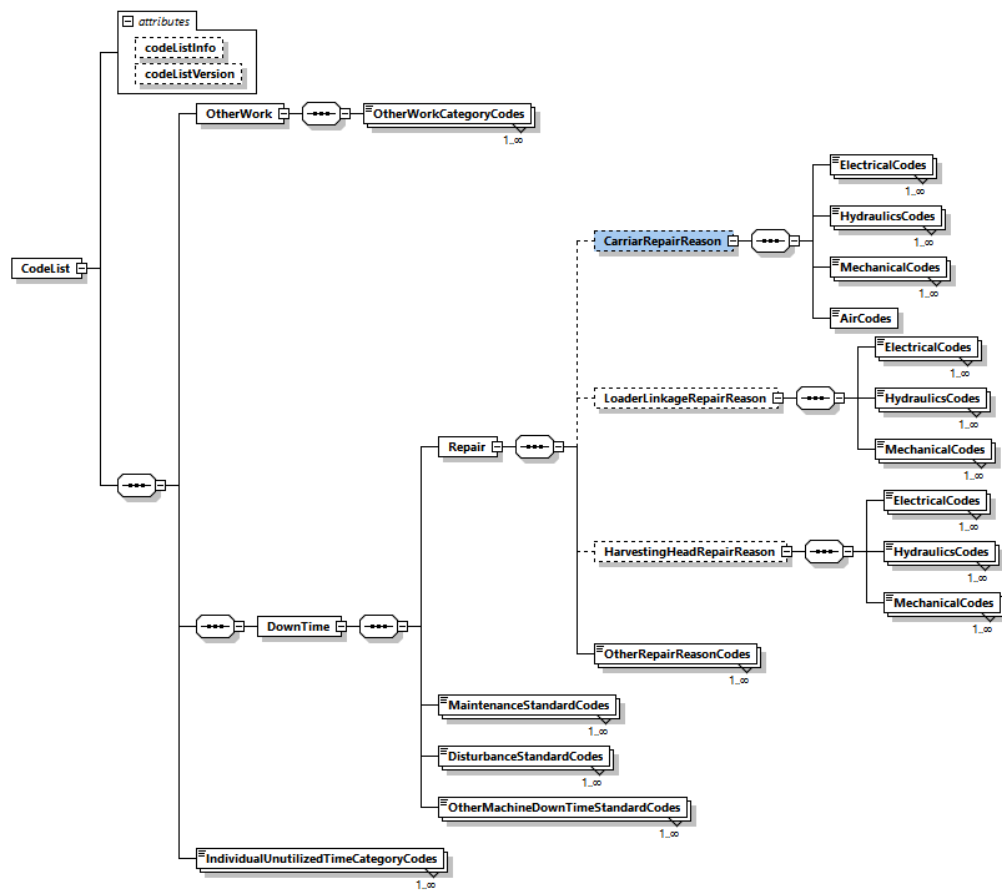
Another improvement could be to make it possible to simplify registration of down time reasons. Basically, not requiring the operator to go into any more details than illustrated below where “No reason” could be used for “Shortened registration”, “Repair”, “Maintenance”, “Disturbance” and “Other Machine down time”:



Below is a flow chart illustrating how a down time is normally registered today. The operator must navigate through a large number of steps in order to select e.g. a specific repair reason.

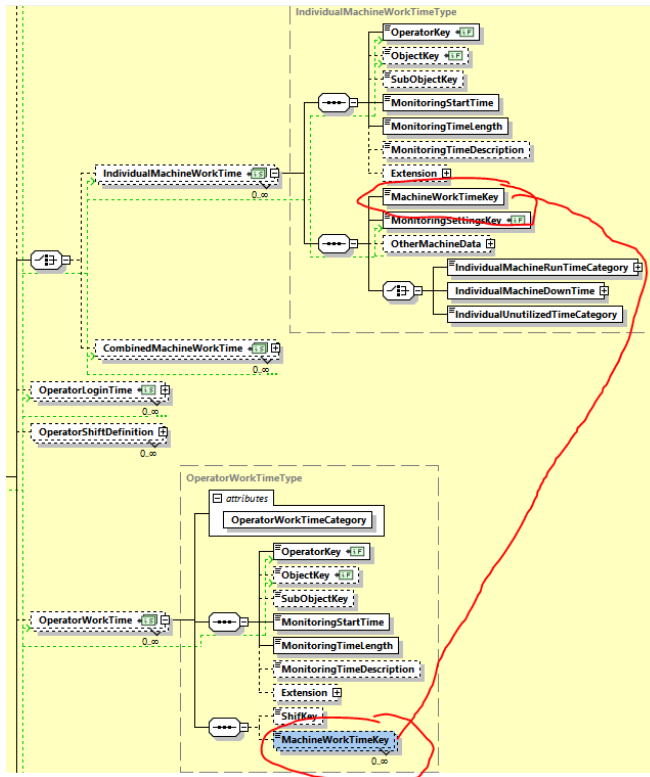


A new schema is suggested for user defined mom codes as illustrated below. This means that the same structure as today is used but codes can easily be adjusted depending on market needs and the interface can easily be simplified.



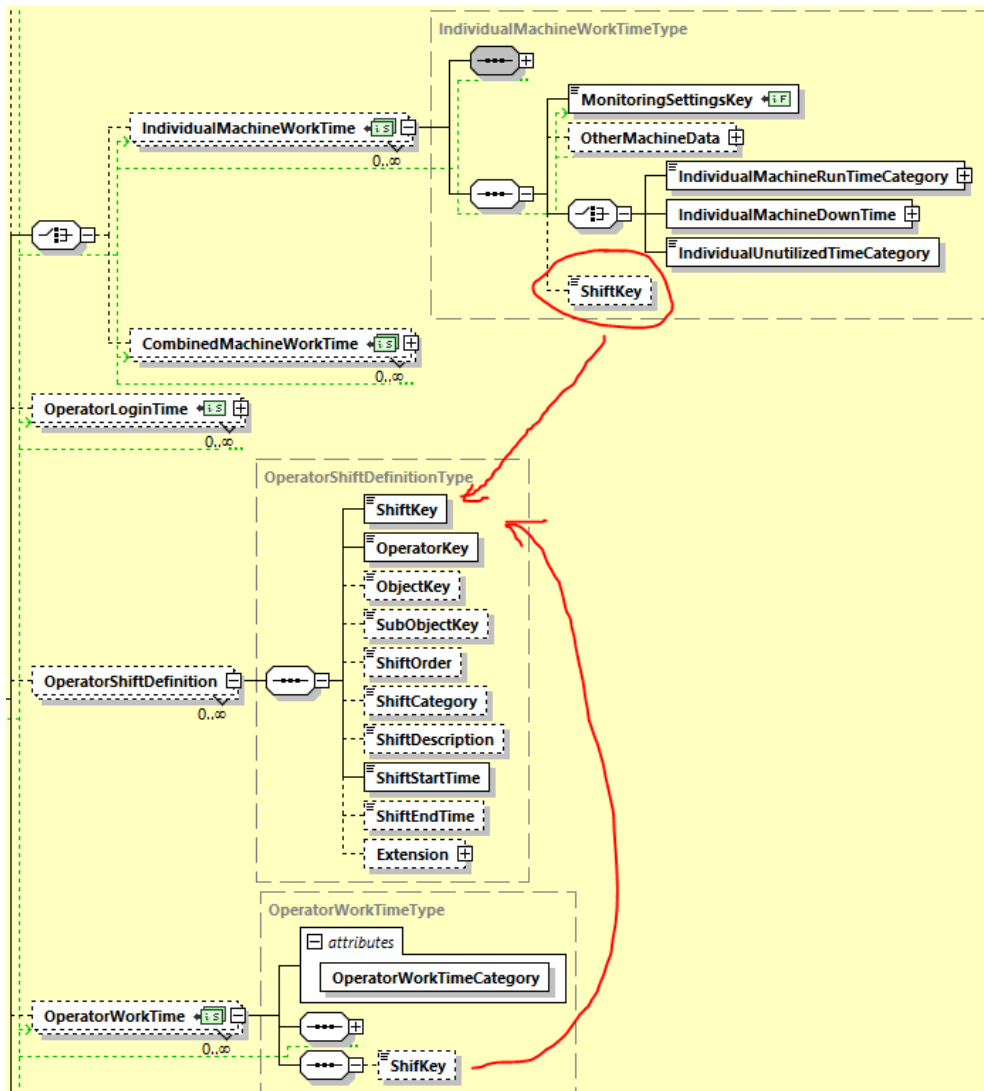
Reference between Operator and MachineWorkTime

Discussion with Henrik Persson 2020-12-11. The possibility of adding MachineWorkTimeKey for each time and add this to OperatorWorkTime was discussed as illustrated below:



The secretary noted that we added a ShiftKey to the IndividualMachineWorkTime at the last meeting as illustrated below.

This means that, before adding more references between MachineWorkTime, OperatorWorkTime and OperatorShiftDef, we need to discuss in more detail how these different structures are related to each other.



It was decided to:

- Exclude combined times and frequency short down times.
- Develop a new schema for distributing user defined down time codes. A use case to be developed to illustrate how to use this schema.
- Exclude ShiftKey from the IndividualMachineWorkTime and to discuss how Shift, OepratorWorkTime, Login and MachineWorkTime are related to each other at the next meeting.

SIGNIFICANT RE-STRUCTURING OF FPR-SCHEMA

Not presented due to running out of time.

It was decided to discuss this issue at the next meeting.

STARTING FORWARDING OBJECT USING HPR

Not presented due to running out of time.

It was decided to discuss this issue at the next meeting.

SUBOBJECT

Discussion StanForD-meeting 2017-11-22

SubObjects are, in several machine systems, mandatory. This means that a SubObject is always created per default when creating a new object.

It was suggested that to make SubObjectKey (under Stem) and SubObject (under Object) mandatory in files sent from forest machines in 4.0.

SubObjects are used by some Sweish companies, e.g. SubObjectUserID is registered by Biometria,

Not presented due to running out of time.

It was decided to discuss this issue at the next meeting.

IMPLEMENTATION RULES

Not discussed due to running out of time.

Decided that it will be discussed at future meetings.

SPELLING ERRORS

Bark functions

Anders Nylund 2014-04-08.

Exclude country specific names for different bark functions.

Older bark functions excluded.

Postponed, to be dealt with jointly with possible addition of a new bark function during the spring 2021.

§7 Items for discussion

DIAMETERSECTIONUSERID

Discussed and updates to structures in oin and hpr decided at previous meeting. The idea of including a volume vector not discussed at present meeting 2021-02-05 due to running out of time.

Decided that it will be discussed at future meetings.

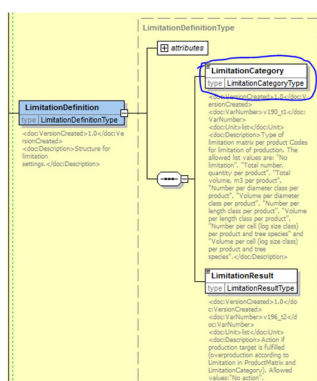
NEW ENUMERATIONS TO LIMITATIONCATEGORYTYPE

Proposal raised by Metsä Group and Stora Enso and prepared by Trimble Forestry (Jyrki Vainio). Discussed at Finnish StanForD meeting 2020-10-29.

Proposal for new type of limitation matrix. This is done by adding new enumeration values to LimitationCategoryType:

- Relative volume in cell (log class) per total diameter class volume

- Relative number in cell (log class) per total diameter class numbers
- Relative volume in cell (log class) per total product volume
- Relative number in cell (log class) per total product numbers



Discussed at previous meeting. Not discussed at present meeting 2021-02-05 due to running out of time.

Decided that it will be discussed at future meetings.

Observe that this is not necessarily a 4.0 issue. Could be added also with 4.1 version.

UNCLASSIFIED STEMS

Discussed at previous meeting. Not discussed at present meeting 2021-02-05 due to running out of time.

Decided that it will be discussed at future meetings.

Observe that this is not necessarily a 4.0 issue. Could be added also with 4.1 version.

DISTRIBUTION BUCKING AND MULTI TREE PROCESSING

Not discussed due to running out of time. Decided that it will be discussed at future meetings.

Observe that this is not necessarily a 4.0 issue. Could be added also with 4.1 version.

§8 StanForD2010 projects / implementations

POSSIBLE FORWARDER TEST IN SWEDEN (SVEASKOG AND SKOGFORSK)

Johan Möller presented an idea to arrange a forwarder test in Sweden during late spring 2021. The test will be carried out together with Sveaskog.

The idea is to run the test including the following parts:

1. New object using manual registration of unloaded volume
 - Start with complete Envelope include pre-defined locations.
 - Open and activate updated Delivery (fdi).

2. New object using manual registration of unloaded volume
 - New Delivery (fdi) added
 - Start with new foi without any correctly defined locations
 - Open and activate updated foi (additional Delivery).
 - Manually create locations with correct coordinates.
 - Take an updated foi (new delivery added) into use.
3. New object using registration of unloaded mass with scale and testing scale control (fqc) if scale measurements are available
 - Start with new foi using SenderXC
4. New object using automatic registration of unloaded volume based on hpr-data (if GIS-functionality is available)
 - Start with new foi
 - Control if volume is included in mom
5. Other tests
 - Control conversion between mass and volume.
 - Check if tracking is included
 - Evaluate GUI for delivery vs product
 - GIS-application (hpr-data presented independently of manufacturer brand)
 - Correct identities in fpr and mom.
 - How is the delivery library working?
 - Registration of negative volumes possible?
 - Simplicity when starting an object with a new foi.
 - Other functionalities that the manufacturer wants to be tested/shown.

§11 Other issues

No other issues.

§12 Next meeting

2021-04-21, 10:00-16:00 CET, Teams

2021-06-15, 10:00-16:00 CET, Team

By default, all financially contributing members will receive an Outlook invitation to all meetings in advance.

§13 End of meeting

The chairman thanked everyone for attending the meeting.

StanForD File REST-API version 0.1

NEW COMMUNICATION METHODS

Version 1.0

Contents

Contents.....	1
Introduction	5
Base-line requirements on authentication.....	5
Authentication for local API.....	6
Authentication for remote API	6
Authorization.....	6
Request	6
Response data	7
StanForD data responses.....	7
List responses	7
Error responses.....	7
400 (Bad request)	8
401 (Unauthorized).....	8
403 (Forbidden).....	8
404 (Not found)	8
405 (Method not allowed).....	8
Special cases with empty/no response	8
Compressing data in Response.....	8
API	9
/.....	9
Get.....	9
Post.....	9
Put	9
Delete	9
Head	9
/Capabilities	9
Get.....	9
Post.....	10
Put	10
Delete	10
Head	10
/File/v0.1.....	10
Get.....	10
Post.....	10
Put	10

Delete.....	10
Head.....	10
/File/v0.1/status/synchronization/<BaseMachineManufacturerID>.....	10
Get.....	10
Post.....	11
Put	11
Delete.....	11
Head.....	11
Retrieving data from machine	11
/File/v0.1/HPR	11
Get.....	11
Post.....	11
Put	12
Delete.....	12
Head.....	12
/File/v0.1/HPR/<HPR-file-ID>	12
Get.....	12
Post.....	12
Put	12
Delete.....	12
Head.....	12
/File/v0.1/HQC	12
Get.....	12
Post.....	13
Put	13
Delete.....	13
Head.....	13
/File/v0.1/HQC/<HQC-file-ID>	13
Get.....	13
Post.....	13
Put	13
Delete.....	13
Head.....	13
/File/v0.1/MOM	13
Get.....	13
Post.....	14
Put	14

Delete.....	14
Head.....	14
/File/v0.1/MOM/<MOM-file-ID>	14
Get.....	14
Post.....	14
Put	14
Delete.....	14
Head.....	14
Sending data to machine	14
/File/v0.1/SPI.....	14
Get.....	14
Post.....	15
Put	15
Delete.....	15
/File/v0.1/SPI/<BaseMachineManufacturerID>	15
Get.....	15
Post.....	15
Put	15
Delete.....	15
Head.....	15
/File/v0.1/PIN.....	15
Get.....	15
Post.....	16
Put	16
Delete.....	16
/File/v0.1/PIN/<BaseMachineManufacturerID>	16
Get.....	16
Post.....	16
Put	16
Delete.....	16
Head.....	16
/File/v0.1/OIN.....	16
Get.....	16
Post.....	17
Put	17
Delete.....	17
Head.....	17

/File/v0.1/OIN/< BaseMachineManufacturerID >	17
Get.....	17
Post.....	17
Put	17
Delete.....	17
Head.....	17
Summary of file APIs	17
Example APIs.....	17
Sending pin-file	18
Retrieving hpr-file.....	18
Example access control list (authorization registry).....	19
Local API.....	19
Remote API.....	19
Example method for retrieving machine data	20
Decision process.....	20

Introduction

The work on developing a new communication interface for StanForD was started with an open meeting in August 2019. The work continued inside a smaller work group with representatives from most manufacturers as well as Skogforsk, Biometria, Metsäteho and UPM.

The StanForD REST-API is a variant of a RESTful data communications interface. This means that the communication should be state-less and consistent. This means that the same command on the same data should generate the same response every time.

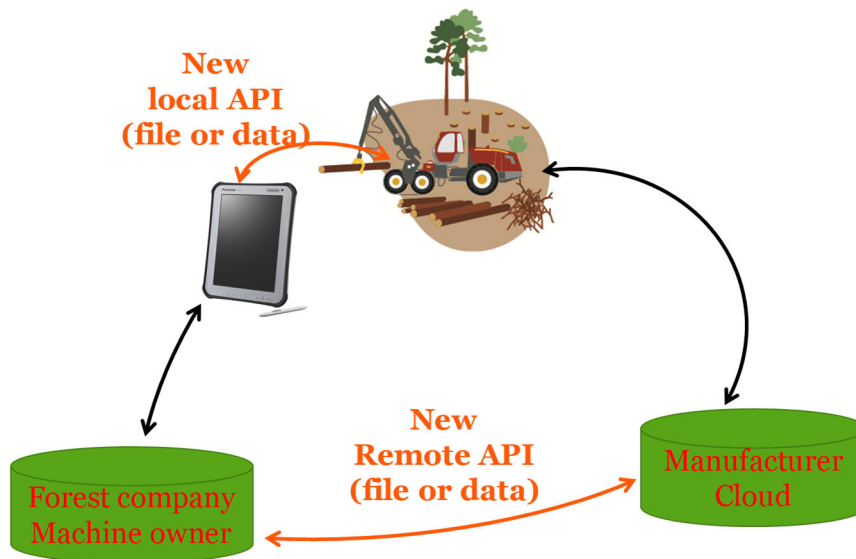
The API uses HTTPS as communication protocol. The HTTPS protocol handles the transmission security for the communication.

The minimum StanForD-version that can be used in this REST-API is 3.0. This means that e.g. versions 2.0 and 2.1 are not supported.

The following terms are used in this document:

- File API, an API for re-distributing a StanForD file conforming to the StanForD2010 schemas. The file is normally created by control system independently of API.
- Data API, an API for extracting a dataset based on the StanForD2010 standard but that may only include a subset of data comparing to the complete schemas.
- Local API, an API used for communication between device inside machine and control system of machine.
- Remote API, an API used for communication with a server/cloud outside machine.

The differences between local and remote APIs is illustrated below.



Base-line requirements on authentication

- TLS (HTTPS) is a requirement on all communication for remote services.
- Any connection to the API without proper authentication shall result in error code 401 (Unauthorized).
- The same API calls (requests) to be used in local and remote APIs. This means that the same logic is to apply in both cases.
- The access control list must define what types of files a user can retrieve.

- It must be possible to also have users (normally machine owner) that can access all LoggingOrganisation.BusinessIDs using some type of wild card.

Authentication for local API

The authentication for a local API is done through HTTP Basic Authentication (RFC7617) and is included in the HTTP header of every question. The Realm shall always be “StanForD”. The User-id field is optional, and not necessary in the authentication. If User-id is not used, the colon “:” before the Password shall be the first character in the sequence to be Base64 encoded. This will make it possible to embed an API-key in the HTTP header as a Password without a User-id.

A local API is assumed to always be used on local network behind some type of firewall.

Authentication for remote API

Different possible authentication solutions have been discussed for remote APIs. No decision has been possible to reach regarding a single standardized solution. It has thus been decided to establish the following baseline requirements for remote API authentication:

- Different manufacturers may implement different frameworks for authentication in remote services.
- Calls to remote services should not be restricted to one single domain. Small forest companies might be using several different service providers such as Biometria, CGI and temporary IT consultants.
- Basic authentication not recommended to be used in remote APIs.

Authorization

All data authorization is based on Logging Organization. Each client login is associated with an access control list for each machine, which authorize different clients to different Logging Organizations. The element ObjectDefinition/LoggingOrganisation/ContactInformation/BusinessID in each data type are then used to identify which Logging Organization that owns the data, and thereby which clients that shall have access to it.

Files without or with empty element LoggingOrganisation/ContactInformation/BusinessID are accessible only to users that have access to all logging organizations (e.g. a machine owner).

Files including several ObjectDefinitions with different ObjectUserIDs will be made available for **all** included logging organizations in the case of using file APIs.

An example of an access control list is included at the end of the document.

The access control list (registry) must define what types of files a user can retrieve.

Request

Requests to the server shall be as HTTP requests to the server URL using one of the following methods GET, POST, PUT, DELETE or HEAD. The URL and methods are described in the chapter API.

Each request must contain the client authentication, to keep the state-less nature of the protocol.

Parameters sent to the server in GET requests are sent in the URL of the API call.

Parameters sent to the server in POST requests are sent URL encoded in the http request header.

Data sent to the server is sent in the http request body in the form of StanForD2010 xml data.

Header must include:

- Content-Type “application/xml” or “text/xml”.
- Content-Length which indicates the size of the body.

- Content-Encoding if compressed files are included.
- Content-Disposition if communicating files as illustrated below:
`setHeader("Content-Disposition", "attachment; filename=\"filename.xml\"");`

All date parameters (StartTime and EndTime) are to be according to UTC as illustrated below:
 YYYY-MM-DDTHH:MM:SSZ e.g. 2020-10-01T15:22:09Z

Exactly the same requests to be used in local and remote APIs. This means that the same logic is to apply in both API types.

Response data

All requests to the server shall generate a response. The minimum response is an HTTP message with an Error code and reason-phrase in the status-line according to RFC7230 section 3.1.2. If the request generates response data, the data shall be sent in the message body.

Header must include:

- Content-Type “application/xml” or “text/xml”.
- Content-Length which indicates the size of the body.
- Content-Encoding if compressed files are included.
- Content-Disposition if communicating files as illustrated below:
`setHeader("Content-Disposition", "attachment; filename=\"filename.xml\"");`

All dates in responses are to be according to UTC as illustrated below:
 YYYY-MM-DDTHH:MM:SSZ e.g. 2020-10-01T15:22:09Z

There are three different response types (data, list and error) as described in sections below.

StanForD data responses

Data is sent in the message body in the form of StanFord2010 xml data.

List responses

When a request results in a reply from the server in the form of a list or collection, the response will contain an XML-formatted response message in the format:

```
<Response>
  <Entry>"Entry 1"</Entry>
  <Entry>"Entry 2"</Entry>
  ...
</Response>
```

Observe that the Entry list can be empty when e.g. no file or machine ids are available based on given parameters. The Entry list is also empty when posting files (spi, pin and oin) to a server.

Error responses

There are a number of error responses the server can give. Normal HTTP status codes are used. Every error response will contain an XML-formatted error message in the format:

```
<Error>
  <Code>"Error code"</Code>
  <Message>"Error message"</Message>
  <Entry>"Entry 1"</Entry>
  <Entry>"Entry 2"</Entry>
  ...
</Error>
```

The specific uses of the error responses are described in the API chapter.

400 (Bad request)

The error code 400 signals that the request contained syntactical or parameter range errors. The Message element shall contain a clear text description of the error reason. The faulty parameter(s) shall be listed in the Entry element(s).

401 (Unauthorized)

The error code 401 signals that the request lacked any authentication credentials. The Message element shall only contain the text “Unauthorized”.

403 (Forbidden)

The error code 403 signals that the file sent to the server in the wrong file format or file version. If the file format is faulty, the Message element shall contain the text “Unsupported file type”, and the Entry elements will contain the supported file types. If the file version is unsupported, the Message element shall contain the text “Unsupported file version” and the Entry elements shall contain the supported file versions.

404 (Not found)

The error code 404 signals that the resource that was requested via the URL does not exist on the server. The Message element shall only contain the text “Not found”. This error code is e.g. to be used if requested file does not exist.

405 (Method not allowed)

The error code 405 signals that the method in the request is not supported on that URL. The Message element shall contain the text “The <method> method is not supported on this resource” and the **Entry** element shall contain the valid methods for this URL.

Special cases with empty/no response

Response from server when posting a file:

- No response body to be returned at all. Status code 200 in header.

Response when no files or machines are available given certain parameters (e.g. within a time interval):

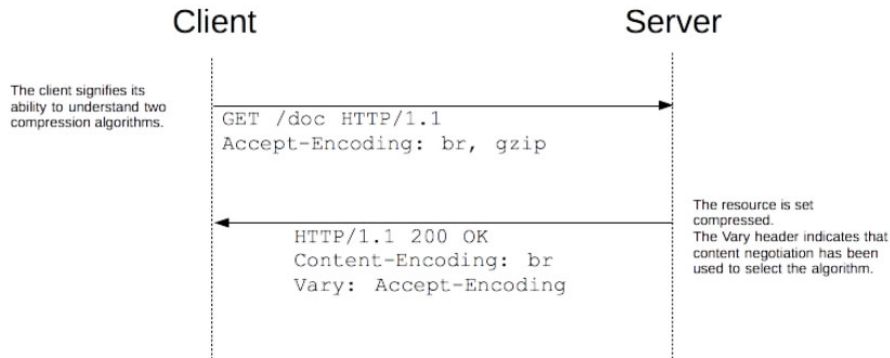
- <Response/>, i.e. no entry-elements included.

Response when a machine or file does not exist:

- Error 404

Compressing data in Response

Since StanForD-files can be of significant size it is relevant to be able to compress data included in the API response. There is functionality for compressing data built into http. The client can specify to the server what type of compression is accepted using http Accept-Encoding in the request header and the type of compression can be returned using Content-Encoding in the response header as illustrated below. The server shall by default compress data if accepted.



No other specific specification of compression methods is thus needed within this specification.

API

/

Root path of REST API. This node does not have to be the root node of the URL. I.e. the URL “https://www.example.com/StamForD/API/” might be the root node, as well as “https://api.example.com/”.

Get

Return a list of valid API versions. E.g. File_v0.1, File_v0.2. So far, only version 0.1 is available.

```

<Response>
  <Entry>"File_v0.1"</Entry>
  ...
</Response>
  
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/Capabilities

Get

Returns a list of which API versions and StanForD file versions that are supported by the server. Also returns the number of days that a file is kept before deletion from server. The response is in the form of an XML data structure.

```

<Capabilities>
  <APIs>
    <API>"File_v0.1"</API>
    <API>"Data_v0.1"</API>
  </APIs>
  <FileVersions>
    <FileVersion>"3.0"</FileVersion>
    <FileVersion>"3.4"</FileVersion>
    <FileVersion>"3.5"</FileVersion>
  </FileVersions>
</Capabilities>
  
```

```
<FileDaysToExpiry>90</FileDaysToExpiry>
</Capabilities>
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1

Root node of File API. Might be multiple versions available on same server. This makes it possible to develop the API and keep compatibility with older versions.

Get

Returns a list of file types that are supported by the server. E.g. HPR, MOM, SPI, PIN, OIN...

```
<Response>
  <Entry>"HPR"</Entry>
  <Entry>"MOM"</Entry>
  ...
</Response>
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/status/synchronization/<BaseMachineManufacturerID>**Get**

A request for the date and time of last complete synchronization between machine and remote server (no pending files in queue waiting to be sent to machine). The request is also useful when asking remote server for production files generated by the machine.

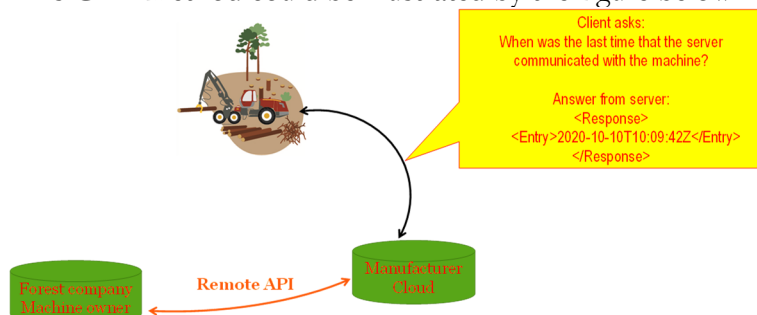
The request is only relevant for remote APIs and NOT for local APIs.

Returns a date and time for last synchronization.

```
<Response>
  <Entry>2020-10-10T10:09:42Z</Entry>
</Response>
```

If BaseMachineManufacturerID does not exist, the server returns Error 404 (Not found).

This GET method could be illustrated by the figure below:



Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

Retrieving data from machine

Hpr, hqc and mom are include below. The same type of calls are also to be implemented for other StanForD2010 files e.g. fpr, fqg, bpr.

/File/v0.1/HPR

Get

A Get will return a list of available HPR-files (HPR-file-IDs). Available parameters to refine the list are; BaseMachineManufacturerID, ObjectUserID, StartDate, EndDate. If one or more parameters are given, the HPR-files with the selected search criteria are returned.

The default values for the parameters are; BaseMachineManufacturerID =<all>, ObjectUserId=<all>, StartDate=1970-01-01T00:00:00.OZ, EndDate=<now>.

The StartDate and EndDate parameters are looking at the CreationDate element in the HPR-files in local APIs while the time stamp when file is received by the server (ReceptionDate) is used in remote APIs. Only files which creation/reception dates are between the StartDate (CreationDate >= StartDate) and the EndDate (CreationDate< EndDate) will be included in the list.

```
<Response>
  <Entry>"98jk430dg2Q"</Entry>
  <Entry>"509xca97f7g"</Entry>
  ...
</Response>
```

No Entry elements are included in cases where e.g. no files are available given parameters used in the request:

```
<Response>
</Response>
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/HPR/<HPR-file-ID>**Get**

A Get will return a data stream containing the selected HPR-file data.

HPR-file-ID is never to change! Same for all other file-IDs.

Return Error 404 (Not found) if file-ID does not exist on the server.

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

A Head will return only the header which includes Content-Length. Can be used before GET in order to determine if file is too large to handle.

HPR-file-ID is never to change! Same for all other file-IDs.

Return Error 404 (Not found) if file-ID does not exist on the server.

/File/v0.1/HQC**Get**

A Get will return a list of available HQC-files (HQC-file-IDs). Available parameters to refine the list are; BaseMachineManufacturerID, ObjectUserID, StartDate, EndDate. If one or more parameters are given, the HPR-files with the selected search criteria are returned.

The default values for the parameters are; BaseMachineManufacturerID =<all>, ObjectUserId=<all>, StartDate=1970-01-01T00:00:00.0Z, EndDate=<now>.

The StartDate and EndDate parameters are looking at the CreationDate element in the HQC-files in local APIs while the time stamp when file is received by the server (ReceptionDate) is used in remote APIs. Only files which creation/reception dates are between the StartDate (CreationDate >= StartDate) and the EndDate (CreationDate < EndDate) will be included in the list.

```
<Response>
  <Entry>"925Q63gd43A"</Entry>
  <Entry>"45asdfv21eQW"</Entry>
  ...
</Response>
```

No Entry elements are included in cases where e.g. no files are available given parameters used in the request:

```
<Response>
</Response>
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/HQC/<HQC-file-ID>**Get**

A Get will return a data stream containing the selected HQC-file data.

HQC-file-ID is never to change! Same for all other file-IDs.

Return Error 404 (Not found) if file-ID does not exist on the server.

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

A Head will return only the header which includes Content-Length. Can be used to avoid downloading too large files when using the method GET.

HQC-file-ID is never to change! Same for all other file-IDs.

Return Error 404 (Not found) if file-ID does not exist on the server.

/File/v0.1/MOM**Get**

A Get will return a list of available MOM-files (MOM-file-IDs). Available parameters to refine the list are; BaseMachineManufacturerID, ObjectUserID, StartDate, EndDate. If one or more parameters are given, the MOM-files with the selected search criteria are returned.

The default values for the parameters are; BaseMachineManufacturerID =<all>, ObjectUserId=<all>, StartDate=1970-01-01T00:00:00.0Z, EndDate=<now>.

The StartDate and EndDate parameters are looking at the CreationDate element in the HQC-files in local APIs while the time stamp when file is received by the server (ReceptionDate) is used in remote APIs. Only files which creation/reception dates are between the StartDate (CreationDate >= StartDate) and the EndDate (CreationDate < EndDate) will be included in the list.

```
<Response>
  <Entry>"8ds32twlVd2"</Entry>
  <Entry>"G59xca97fV7g"</Entry>
  ...
</Response>
```

No Entry elements are included in cases where e.g. no files are available given parameters used in the request:

```
<Response>
</Response>
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/MOM/<MOM-file-ID>**Get**

A Get will return a data stream containing the selected MOM-file data.

MOM-file-ID is never to change! Same for all other file-IDs.

Return Error 404 (Not found) if file-ID does not exist on the server.

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

A Head will return only the header which includes Content-Length. Can be used before GET in order to determine if file is too large to handle.

MOM-file-ID is never to change! Same for all other file-IDs.

Return Error 404 (Not found) if file-ID does not exist on the server.

Sending data to machine

Oin, pin and spi are include below. The same type of calls are also to be implemented for other StanForD2010 files e.g. foi, fdi, boi and env.

Observe that the POST API for distributing a file to all machines within a LoggingOrganization is only to be implemented for spi, pin, fdi and not for oin, foi, boi or env.

/File/v0.1/SPI**Get**

A Get with no additional parameters will return a list of available BaseMachineManufacturerID. Since each machine is allowed to have different definitions of each Species, SPI-files must be handled on Machine level.

```
<Response>
  <Entry>"r6434565"</Entry>
  ...
</Response>
```

No Entry elements are included in cases where no machines are available:

```
<Response>
</Response>
```

Post

A Post with an optional BusinessID parameter set to a Logging Organization will add the SPI-file to the inbox of all Machines associated with the Logging Organization. A Post without BusinessID will add the PIN-file to all Machines that the user has access to. The Post must contain a valid SPI-file in the payload and can include optional parameter BusinessID.

No response required in case of successful Post (status code 200).

If the server cannot handle the StanForD-version of SPI-file, the server returns Error 403 (Forbidden).

If the user does not have access to the BusinessID, the server returns Error 400 (Bad request).

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

/File/v0.1/SPI/<BaseMachineManufacturerID>**Get**

Returns Error 405 (Method Not Allowed).

Post

A Post must contain a valid SPI-file in the payload. The SPI-file will be added to the Machines inbox.

No response required in case of successful Post (status code 200).

If the server cannot handle the StanForD-version of SPI-file, the server returns Error 403 (Forbidden).

If BaseMachineManufacturerID does not exist, the server returns Error 404 (Not found).

Put

Returns Error 405 (Method Not Allowed).

Delete

Returns Error 405 (Method Not Allowed).

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/PIN**Get**

A Get with no additional parameters will return a list of available BaseMachineManufacturerID. Since each machine is allowed to have different definitions of each Product, PIN-files must be handled on Machine level.

```
<Response>
  <Entry>"r6434565"</Entry>
  ...
</Response>
```

No Entry elements are included in cases where no machines are available:

```
<Response>
</Response>
```

Post

A Post with an optional BusinessID parameter set to a Logging Organization will add the PIN-file to the inbox of all Machines associated with the Logging Organization. A Post without BusinessID will add the PIN-file to all Machines that the user has access to. The Post must contain a valid PIN-file in the payload and can include optional parameter BusinessID.

No response required in case of successful Post (status code 200).

If the server cannot handle the StanForD-version of PIN-file, the server returns Error 403 (Forbidden).

If the user does not have access to the BusinessID, the server returns Error 400 (Bad request).

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

/File/v0.1/PIN/<BaseMachineManufacturerID>**Get**

Returns Error 405 (Method Not Allowed)

Post

A Post must contain a valid PIN-file in the payload. The PIN-file will be added to the Machines inbox.

No response required in case of successful Post (status code 200).

If the server cannot handle the StanForD-version of PIN-file, the server returns Error 403 (Forbidden).

If BaseMachineManufacturerID does not exist, the server returns Error 404 (Not found).

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/OIN**Get**

A Get with no additional parameters will return a list of available BaseMachineManufacturerID. Since each machine is allowed to have different definitions of each Object, OIN-files must be handled on Machine level. A specific OIN is thus never sent to all machines.

```
<Response>
  <Entry>"r6434565"</Entry>
  ...
</Response>
```

No Entry elements are included in cases where no machines are available:

```
<Response>
</Response>
```

Post

Returns Error 405 (Method Not Allowed)

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

/File/v0.1/OIN/< BaseMachineManufacturerID >**Get**

Returns Error 405 (Method Not Allowed)

Post

A Post must contain a valid OIN-file in the payload. The OIN-file will be added to the Machines inbox.

No response required in case of successful Post (status code 200).

If the server cannot handle the StanForD-version of PIN-file, the server returns Error 403 (Forbidden).

If BaseMachineManufacturerID does not exist, the server returns Error 404 (Not found).

Put

Returns Error 405 (Method Not Allowed)

Delete

Returns Error 405 (Method Not Allowed)

Head

Returns Error 405 (Method Not Allowed)

Summary of file APIs

Table below gives an overview of available APIs for different file categories. XXX stands for any of the specified file types.

File types	File/v0.1/XXX GET	File/v0.1/XXX POST	File/v0.1/XXX/ <XXX-file-ID> GET	File/v0.1/XXX/ <BaseMachineManufacturerID> POST
Hpr, hqc, fpr, fqc, bpr, mom	List of all available files	*	Specific file downloaded	*
Pin, spi, fdi	List of all available machines	File uploaded to all machines in LoggingOrganization	*	File uploaded to specific machine
Oin, foi, boi, env	List of all available machines	*	*	File uploaded to specific machine

*Method not allowed

Example APIs

Below follows an example communication between a client and a server where the client will send a PIN-file to all machines in logging organization 'BigCompany1' and then get the HPR-files generated between 2019-11-23 00:00:00Z and 2019-11-23 23:59:59Z.

Sending pin-file**Client to Server*****POST*** /File_v0.1/PIN?BusinessID=BigCompany1 ***HTTP/1.1****Host: fleet.example.com:443**Authorization: Basic VXNlcjE6MTIzNDU2**Content-Length: 29587**Content-Type: application/xml**Content-Disposition: attachment; filename="BigSawMill.pin"**<?xml version="1.0" encoding="utf-8"?><ProductInstruction...***Response from Server*****HTTP/1.1 200 OK****Content-Length: 0***Retrieving hpr-file****Client to Server*****GET*** /File_v0.1/HPR?StartDate=2019-11-23T00:00:00Z&EndDate=2019-11-23T23:59:59Z***HTTP/1.1****Host: fleet.example.com:443**Authorization: Basic VXNlcjE6MTIzNDU2**Content-Length: 0***Response from Server*****HTTP/1.1 200 OK****Content-Length: 91**Content-Type: application/xml**<Response>**<Entry>"Object1-1.hpr"</Entry>**<Entry>"Object1-2.hpr"</Entry>**</Response>***Client to Server*****GET*** /File_v0.1/HPR/Object1-1.hpr ***HTTP/1.1****Host: fleet.example.com:443**Authorization: Basic VXNlcjE6MTIzNDU2**Content-Length: 0**Accept-Encoding: gzip, br, deflate***Response from Server*****HTTP/1.1 200 OK****Content-Length: 3225343**Content-Encoding: gzip**Content-Type: application/xml**Content-Disposition: attachment; filename="Object1-1.hpr"**<?xml version="1.0" encoding="UTF-8"?><HarvestedProduction...***Client to Server*****GET*** /File_v0.1/HPR/Object1-2.hpr ***HTTP/1.1***

Host: fleet.example.com:443
 Authorization: Basic VXNlcjE6MTIzNDU2
 Content-Length: 0
 Accept-Encoding: gzip, br, deflate

Response from Server

HTTP/1.1 200 OK

Content-Length: 1832664

Content-Encoding: gzip

Content-Type: application/xml

Content-Disposition: attachment; filename="Object1-2.hpr"

<?xml version="1.0" encoding="UTF-8"?><HarvestedProduction...

Example access control list (authorization registry)

In order to keep track of the rights for accessing data some type of list must be kept by the server. The following example is split into two separate sections for local and remote APIs since there are some fundamental differences. The local API is only used for communicating with one single machine while the remote API can be used for communicating with a large number of machines simultaneously.

It is suggested that we recommend that these two access lists are kept separate (not synchronized) in order to avoid that changes to the same user can be made both remotely and locally. Having the possibility to view also the local list also by a remote solution is probably not a problem but the local access control list is to be edited in the machine.

Observe that additional limitations may very well be implemented such as a specific time interval, specific harvesting objects etc.

Local API

Access control list is set by operator in machine. User-ID and Password is probably presented in interface of control system and manually copied to local client.

Below are examples of access control lists for two separate machines.

BaseMachine-ManufacturerID: 5fd4nx643t56

User	Pass-word	BaseMachine-ManufacturerID	LoggingOrg.-BusinessID	File types
MachineOwner	8me02	5fd4nx643t56	* (All)	* (All)
Holmen	432g5	5fd4nx643t56	105	hpr, hqc, pin, spi, oin
Sveaskog	3j48cxv	5fd4nx643t56	106	hpr, hqc, pin, spi, oin
Södra	76jer6	5fd4nx643t56	110	hpr, hqc, pin, spi, oin
Biometria	87cso9	5fd4nx643t56	105	hpr, hqc, pin, spi
			106	hpr, hqc, mom, pin, spi
			110	hpr, hqc, pin, spi

BaseMachine-ManufacturerID: SS39jkfd435

User	Pass-word	BaseMachine-ManufacturerID	LoggingOrg.-BusinessID	File types
MachineOwner	7u65	SS39jkfd435	* (All)	* (All)
Sveaskog	Hert6	SS39jkfd435	106	hpr, hqc, pin, spi, oin
Biometria	5ejhd	SS39jkfd435	106	hpr, hqc, pin, spi
Storaenso	!5nn6	SS39jkfd435	349erds9	Mom, hpr, hqc

Remote API

Access control list set by machine owner.

User	Pass-word	BaseMachine- ManufacturerID	LoggingOrg.- BusinessID	File types
MachineOwner	63gt65	5fd4nx643t56	* (All)	* (All)
		SS39jkfd435	* (All)	* (All)
Holmen	Xy4h7	5fd4nx643t56	105	hpr, hqc, pin, spi, oin
Sveaskog	Xert6	5fd4nx643t56	106	hpr, hqc, pin, spi, oin
		SS39jkfd435	106	hpr, hqc, pin, spi, oin
Södra	X6jer6	5fd4nx643t56	110	hpr, hqc, pin, spi, oin
Biometria	Xejhd	5fd4nx643t56	105	hpr, hqc, pin, spi
			106	hpr, hqc, mom, pin, spi
			110	hpr, hqc, pin, spi
		SS39jkfd435	106	hpr, hqc, pin, spi
Storaenso	!Xnn6	SS39jkfd435	349erds9	Mom, hpr, hqc

It is possible, based on the access list above, to compile a list which defines which machines “belongs” to a specific logging organization. This type of list is relevant in cases when there is a need to upload e.g. a pin file to all machines working for a specific logging organization (POST /File_v0.1/PIN?BusinessID=106).

LoggingOrganization - BusinessID	BaseMachine - ManufacturerID
105	5fd4nx643t56
106	5fd4nx643t56 / SS39jkfd435
110	5fd4nx643t56
349erds9	SS39jkfd435

Example method for retrieving machine data

Observe that the only completely safe method to evaluate whether all data has been reported to the client is by checking that no StemNumbers or LoadNumbers are missing for a specific harvesting object.

A simple algorithm run every hour for retrieving and checking files could be to:

1. Check for all file-ids generated the last 24 hours.
2. Download all file-ids that has not been previously downloaded.
3. Import data into database
4. Check that no StemNumbers/LoadNumbers are missing from objects (a complete interval should exist from 1 to the last StemNumber in the file with EndDate).

Observe that the algorithm above is based on having a list with all historically downloaded file-ids.

Decision process

Decisions to release new version of the StanForD APIs will be made parallel with new StanForD2010 versions. This means that the first “release” version of this document will at the earliest be published in February 2021.

The technical work group will discuss how to start working on a data API.



Arlinger John
Ian Wilson

30 November 2020

New yarder and skidder messages

Introduction

New schemas for feller bunchers were added to the standard in 2018 (boi, bpr) and 2019 (bqc). Including other tree length machines, such as skidders and yarders, were discussed already in 2018 but it was decided to postpone these machine types.

It was suggested by Jeremy Gibson and Ian Wilson (2019-07-01) to start working on new schemas also for yarders and skidders.

The old production proposal (based on the fpr) for skidders from 2018 is illustrated below.

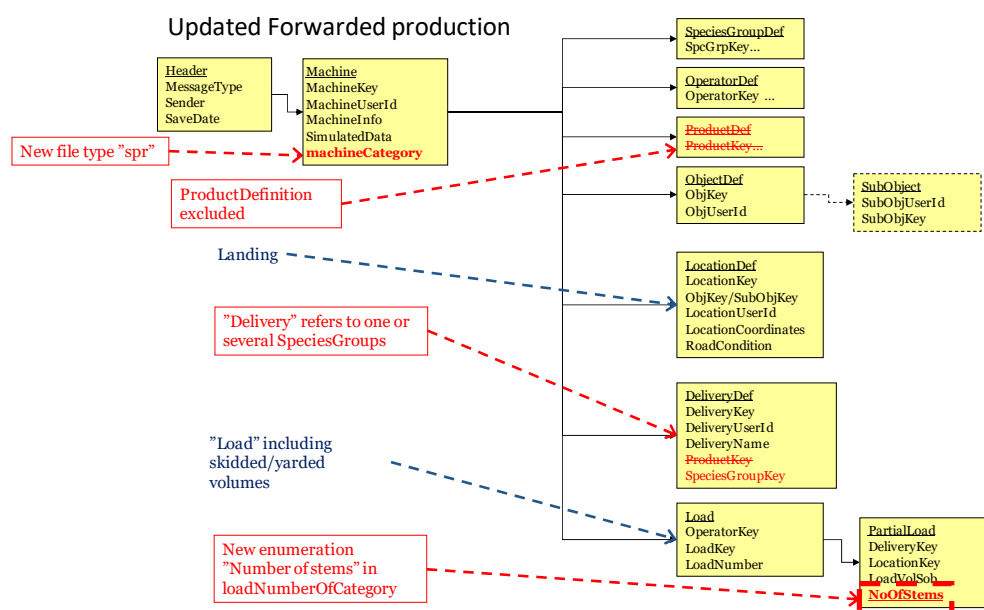


Figure 1. Early preliminary proposal for new skidder and yarder production message.

In discussions between Ian, Jeremy and John Arlinger it was deemed best to keep yarders and skidders separate as was also discussed at the meeting 2018-11-20 (table below).

Table 1. File extensions also to be used for tree length machines are described below.

	Object Instruction	Production Report	Quality Control	Operational Monitoring
Harvester	Oin+Pin+Spi	Hpr	Hqc	Mom
Forwarder	Foi+Fdi	Fpr	Fqc	Mom
Feller Buncher	Boi+Spi	Bpr	Bqc	Mom
Skidder	Soi+Spi	Spr	-	Mom
Yarder	Yoi+Spi	Ypr	-	Mom

This document includes proposals for production reports and object instructions. Adding support for these machines also in operational monitoring reports (mom) will be postponed until version 4.0

Skidder production

The basic idea is to report each skidder-load with information about species and number of stems. This means that the Deliveries and Products are irrelevant when comparing with the forwarder.

The following existing elements and attributes need new enumeration values:

- Add “spr” to attribute messageCategory
- Add “Skidder” to attribute machineCategory
- Add “Number of stems” to attribute loadNumberOfCategory

When comparing with the present standard for forwarder production the following new parameters are deemed to be required:

- **Winch**
 - Brand/Model/Year
- **Extraction Method**
 - Grapple
 - Choker/Winch
- **Stem Presentation**
 - Un-bunched – stems are generally not grouped into a bunch and are typically felled by a person with a chainsaw
 - Part Bunched – stems are roughly grouped into a bunch and are typically mechanically felled using a Dangle Head feller buncher
 - Bunched – stems are precisely grouped into a bunch and are typically mechanically felled using a Fixed Head feller buncher
 - Machine Fed – stems are placed or lifted by a separate machine into the grapple or for the grapple to grab them

It is suggested to add Winch information to the Machine and to add ExtractionMethod and StemPresentation to the Load as illustrated below. In most cases ExtractionMethod and StemPresentation are not changed within an object but they sometimes are depending e.g. on the terrain variability. This information is set by the operator.

Observe that the figure below is a simplification of the schema. The idea is to keep all elements and attributes from the fpr-schema except for those explicitly mention in previous section.

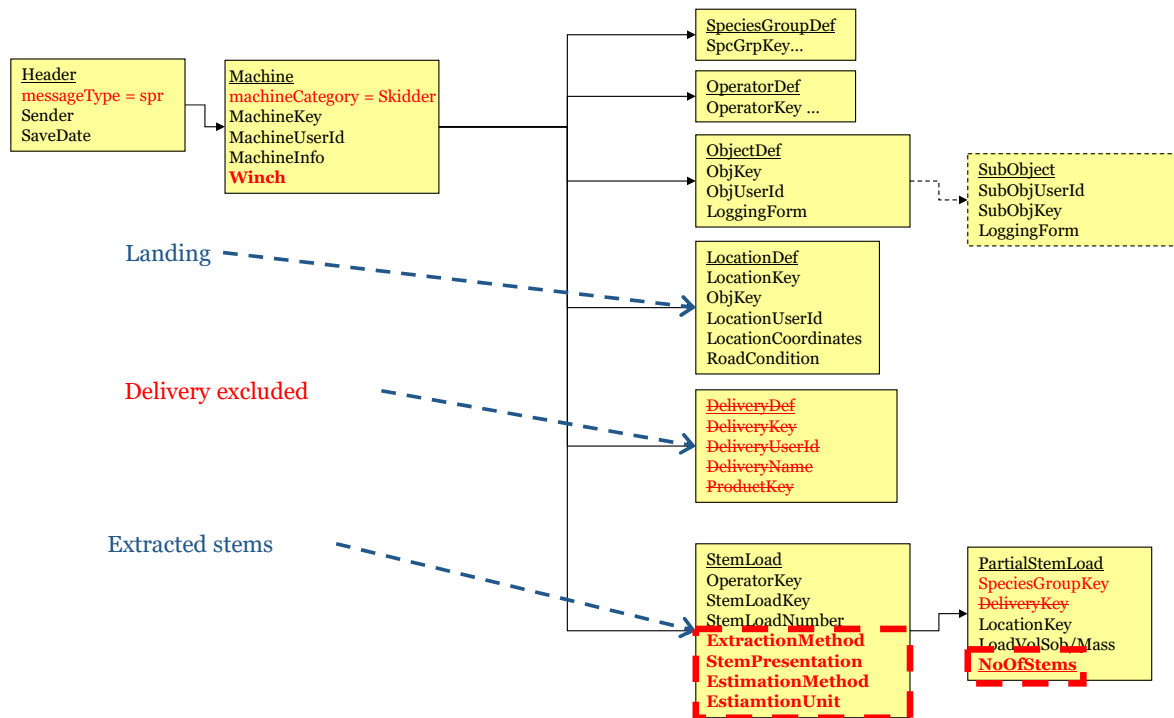
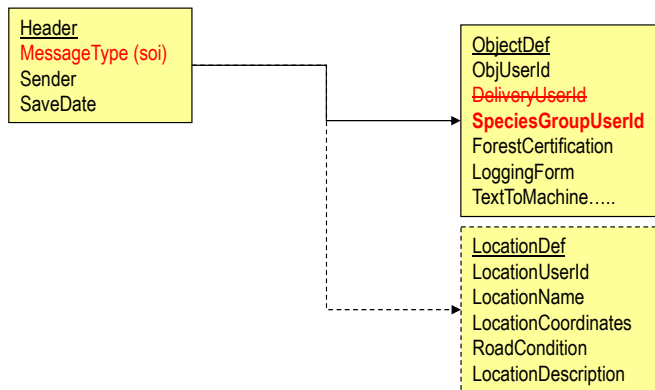


Figure 2. New or updated elements or attribute in red color. Observe that DeliveryDefinition is to be excluded.

Observe that machineCategory will not include the enumeration value "Skidder" until major version 4.0. Instead the value "Other" has to be used in the skidder production message.

Skidder instruction

The object instruction for skidders (soi) is almost identical with the forwarder instruction with the exception that element ProductUserID is replaced with element SpeciesGroupUserID as illustrated below:



Observe that the skidder also needs an spi-file in order to include the necessary information regarding the relevant SpeciesGroup.

Yarder production

The basic idea is to report each yarder-load with information about species and number of stems. This means that the Deliveries and Products are irrelevant when comparing with the forwarder.

The following existing elements and attributes need new enumeration values:

- Add “ypr” to attribute messageCategory
- Add “Yarder” to attribute machineCategory
- Add “Number of stems” to attribute loadNumberOfCategory

When comparing with the present standard for forwarder production the following new parameters are deemed to be required:

- **Carriage**
 - Brand/Model/Year
- **Extraction Method**
 - Grapple
 - Choker/Winch
- **Stem Presentation**
 - Un-bunched – stems are generally not grouped into a bunch and are typically felled by a person with a chainsaw
 - Part Bunched – stems are roughly grouped into a bunch and are typically mechanically felled using a Dangle Head feller buncher
 - Bunched – stems are precisely grouped into a bunch and are typically mechanically felled using a Fixed Head feller buncher
 - Machine Fed – stems are placed or lifted by a separate machine into the grapple or for the grapple to grab
- **Tail Hold**

- Manual – The yarders wire ropes on the "backline" are shifted manually by people
- Mobile – The yarders wire ropes on the "backline" are shifted via a machine (excavator/bulldozer) that is driven by a person i.e. it's not manned all the time.
- Mobile Tele-Operated – The yarders wire ropes on the "backline" are shifted via a machine (excavator/bulldozer) that has been fitted with a remote control system and camera's so that the Yarder operator can remotely start the excavator up and drive it to the next Yarder line while never having to leave the Yarder
- **Rigging System** (how the wire rope systems are operated)
 - Northbend
 - Southbend
 - Shotgun
 - Highlead
 - Grabinski
 - Slack pulling carriage

It is suggested to add Carriage information to the Machine and to add ExtractionMethod , StemPresentation, RiggingSystem and TailHold to the Load as illustrated below. In most cases these values are not changed within an Object but they sometimes are depending e.g. on the terrain variability. This information is set by the operator.

Observe that the figure below is a simplification of the schema. The idea is to keep all elements and attributes from the fpr-schema except for those explicitly mention in previous section.

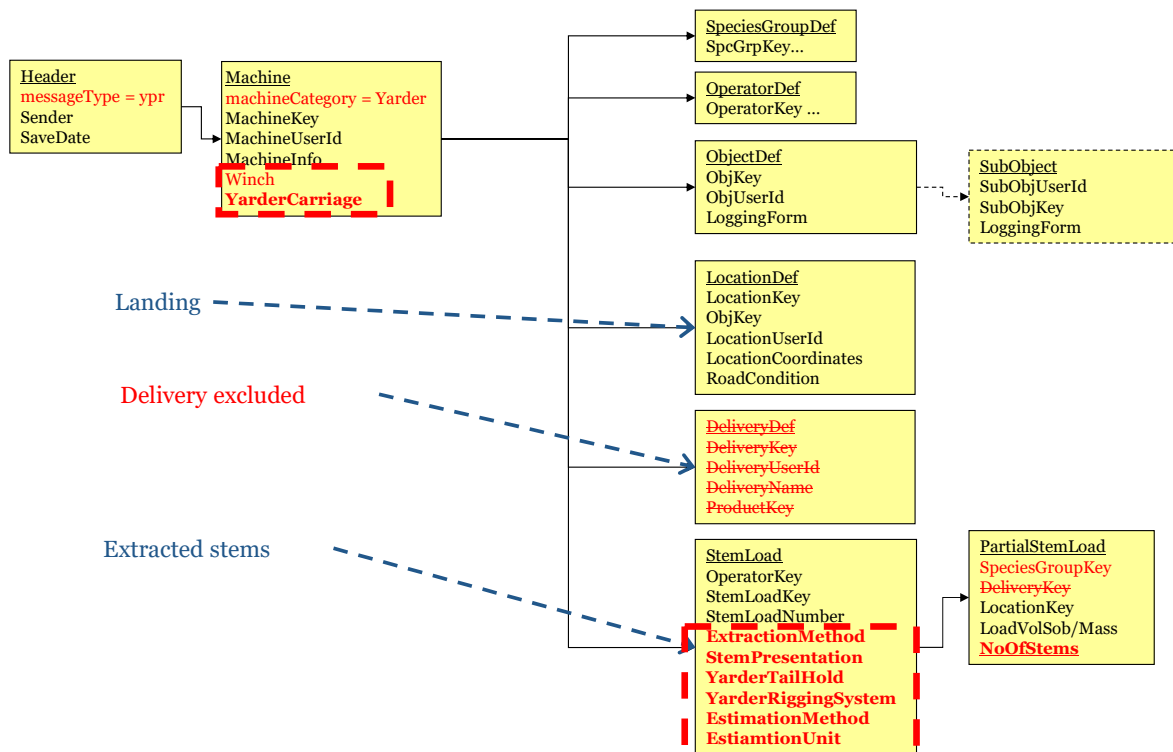
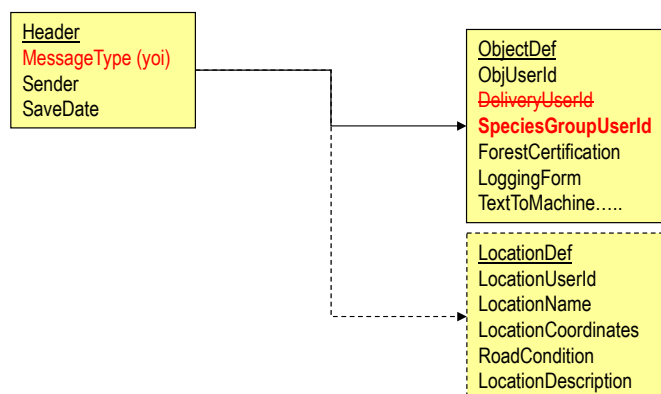


Figure 3. New or updated elements or attribute in red color. Observe that DeliveryDefinition is to be excluded.

Observe that machineCategory will not include the enumeration value "Yarder" until major version 4.0. Instead the value "Other" has to be used in the yarder production message.

Yarder instruction

The object instruction for yarders (yoi) is almost identical with the forwarder instruction with the exception that element ProductUserID is replaced with element SpeciesGroupUserID as illustrated below:



Observe that the yarder also needs an spi-file in order to include the necessary information regarding the relevant SpeciesGroup.

Detailed description of new parameters

EXTRACTION METHOD

Observe that a skidder can be equipped with both winch and grapple and therefore might use both of them even though using a winch is becoming less frequent.

Grapple

Skidder or yarder using a grapple to extract logs.





Choker

Skidder or yarder using a choker and/or winch to extract logs.



STEM PRESENTATION

Type of stem presentation which describes how stems are positioned before being picked up by skidder or yarder.

Un-bunched

Stems are generally not grouped into a bunch and are typically felled by a person with a chainsaw.



Part Bunched

Stems are roughly grouped into a bunch and are typically mechanically felled using a Dangle Head feller buncher.



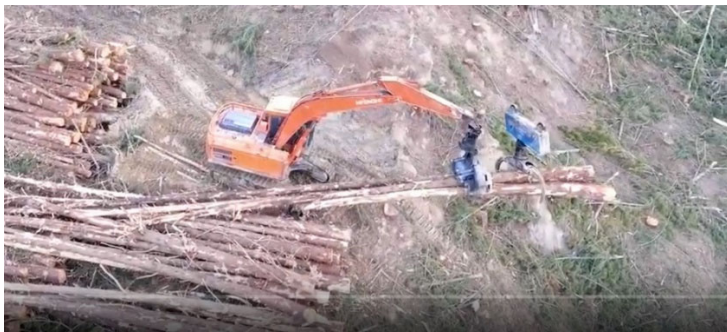
Bunched

Stems are precisely grouped into a bunch and are typically mechanically felled using a Fixed Head feller buncher.



Machine Fed

Stems are placed or lifted by a separate machine into the grapple or for the grapple to grab.



TAIL HOLD

Type of tail hold describes how the wire ropes on the back line are moved from one extraction line to another.

Manual

The wire ropes on the backline are shifted manually by people across to the next Yarder extraction line. Slow and time consuming compared to the other 2 methods.

Mobile

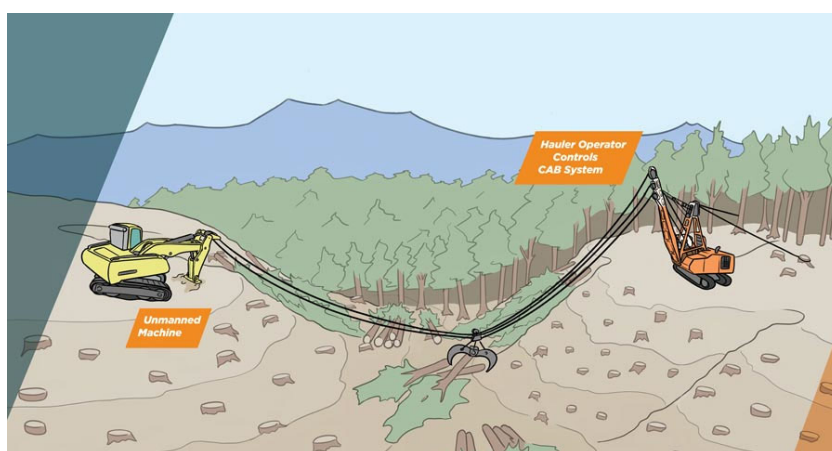
The wire ropes on the back line are shifted by machine across to the next Yarder extraction line. This machine is normally an excavator or bulldozer that gets driven by a person i.e. it's not manned all the time and a person usually will drive from the Yarder out to where the Tail

Hold machine is located, then shift the excavator/bulldozer across to the next Yarder line, then drive back to the Yarder.



Mobile Tele Operated

Similar to above, but the machine has been fitted with a remote control system and camera's so that the Yarder operator can remotely start the excavator up and drive it to the next Yarder extraction line while never having to leave the Yarders cab (this method is typically much quicker and doesn't need to take someone away from doing their normal job).



Rigging System

This parameter describes how the wire rope systems are operated. No detailed description of different rigging systems in this document.

Northbend

Southbend

Shotgun

Highlead

Grabinski

Slack pulling carriage

New Object geographical instruction

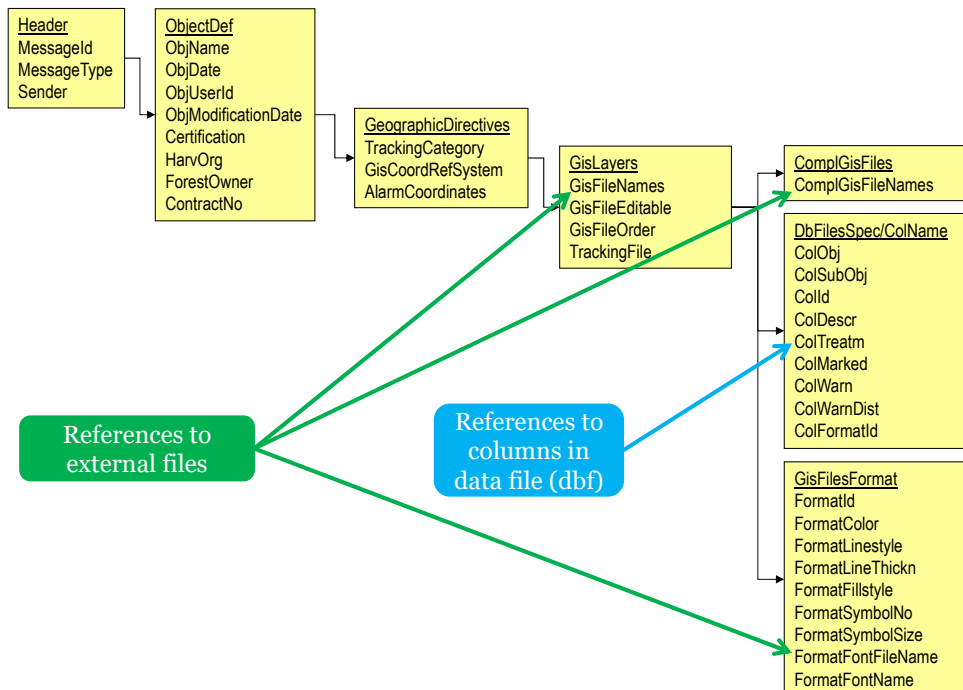
Schema re-structuring of ogi, StanForD2010 4.0

Contents

Contents.....	1
Schema re-structuring.....	2

Schema re-structuring

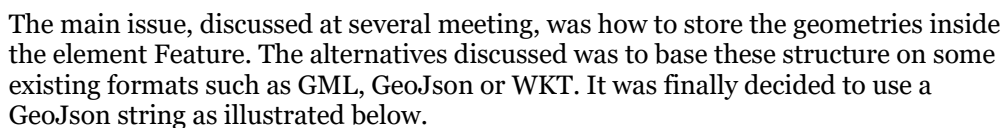
The possibility of re-modelling ogi has been discussed in order to make it simpler and not dependent on a number of other file types such as shp- and font-files as illustrated below. Another important driving force is that the present standard version, partly due to the complex structure, has been implanted in a heterogenous way.



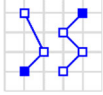
Other driving forces are recent proposals to:

- Implement a standardized set of symbols using formats such as svg and png.
- Use the format for simple and frequent (daily) updates of areas not to be harvested, e.g. protecting koala habitats (“sleeping areas” that can change daily).
- Use the format for frequent (twice a day) operator feedback regarding based on analysing hpr-data in central server .
 - Thinning intensity
 - Nature conservation
- Open possibility to include text-based information aimed at the operator using html or url rather than today a complex pdf.

The new suggested structure means replacing the old element GeographicDirectives with a few new elements as illustrated below (observe that a number of existing ObjectDefinition-elements are not included in the example):



3

MultiLine:

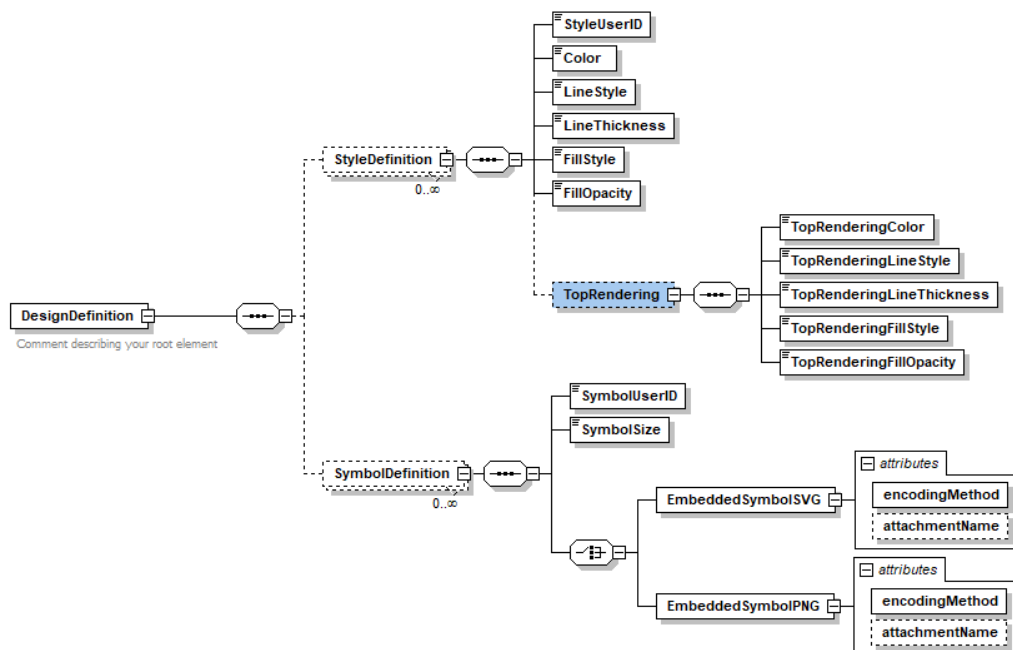
```
{
  "type": "MultiLineString",
  "coordinates": [
    [[10, 10], [20, 20], [10, 40]],
    [[40, 40], [30, 30], [40, 20], [30, 10]]
  ]
}
```

MultiPolygon;

```
{
  "type": "MultiPolygon",
  "coordinates": [
    [
      [[40, 40], [20, 45], [45, 30], [40, 40]]
    ],
    [
      [[20, 35], [10, 30], [10, 10], [30, 5], [45, 20], [20, 35]],
      [[30, 20], [20, 15], [20, 25], [30, 20]]
    ]
  ]
}
```

Another fundamental change that was suggested is to move formatting (design) to a separate Design message and to utilize StyleUserID and SymbolUserID as a reference between the structure above and the Design message. The Design structure is illustrated below. Observe that a new parameter FillOpacity has been added when comparing with the old standard. Symbols (svg or png) can be included inside the message using element EmbeddedSymbolSVG/PNG using encodingMethod Base64.

It was decided to add a new schema called DesignTemplate according to this structure:



The following figure illustrates how individual features are connected to specific symbol or style in the DesignDefinition.

Element	Feature 1	Feature 2	Feature 3	DesignCategory- UserID	Line- Color	Line- Style	Line- Thickness
Name (label)	Main landing	Cutting	Route 66	CuttingArea	Blue	Solid	3
Treatment	Both sides of road	Clear fell		StripRoadLine	Brown	Dot	1
Description		Pine forest		RoadLine	Black	Solid	2
GeometryType	Point	Polygon	Line	PowerLine	Red	Dash	1
Geometry	(30 10)	((30 10, 40 40, 20 40, 10 20, 30 10))	(30 10, 10 30, 40 40)				
Order	1	2	3				
DesignCategoryUserID		CuttingArea	RoadLine				
SymbolUserID	Landing		Road				
PresentationGroupUserID	Logging	Logging	Roads				

SymbolUserID	Symbol	Symbol-Size
NoLogging		5
Landing		5
Road		4

The intention is that Skogforsk will develop a default Design definition that can be used by any actor.