John Arlinger, Skogforsk
Johan J. Möller, Skogforsk
Juha-Antti Sorsa, Metsäteho
Tapio Räsänen, Metsäteho

6 February 2019

Introduction to StanForD 2010

STRUCTURAL DESCRIPTIONS AND
IMPLEMENTATION RECOMMENDATIONS

DRAFT!

SKOGFORSK

DRAFT!

SKOGFORSK

Contents

Background

The objective is to give the reader an introduction to StanForD 2010 as well as implementation recommendations.

The development of StanForD 2010 started 2006-08-25 with a decision to launch a pre-study. A year later it was decided to continue the project towards a final new StanForD version. A project group consisting of representatives from the machine manufacturers as well as Skogforsk, Metsäteho and SDC was formed. Representatives from Logica and Haglöf have also been present at project group meetings. The first version of StanForD was accepted during the spring of 2011.

This document was written during the development of StanForD 2010. It is a living document continuously updated. It is thus not an official standard document. The official standard documents are (www.skogforsk.se/StanForD 2010:

- XML schemas
- StanForD 2010 naming and design
- StanForD 2010 rules

The main objectives of the new standard version are to achieve:

- Improved structures that supports data management requirements of today
- Better structural descriptions
- Stricter priorities and implementation rules
- A system for management of standard versions
- Reduction of old variables and complex structures

This does not however mean that we will simplify the use of harvesters and forwarders in a significant way. Controlling a forest machine is complex and has to be complex since the financial values handled by the machines are very large.

It is not possible and actually not desirable to totally leave the old StanForD behind since many parts of the old standard works very well. Instead it has been the aim to use the well-functioning parts of the present standard while leaving the problematic parts behind while at the same time open the standard for future development.

A detailed analysis of old variables and file types was carried out when starting the development. A large number of old StanForD-variables were excluded based on the fact that:

- their priority is low (4)
- the old prd-structure is replaced by the pri-structure
- only absolute and not relative values are to be used
- that aggregated data should be avoided
- no binary numbers are to be used

Observe that the term logging organisation in this document refers to the organisation buying logging services from a machine owner (often a contractor). The logging organisation is sometimes also the machine owner as

well as the forest owner. Logging organisation is sometimes also referred to as a forest company. The following old terms have been changed in the new standard:

- Assortment replaced by the term product
- Species is replaced by the term species group.
- Transport object replaced by the terms delivery and location

Introduction

MESSAGES

It is useful to think in terms of different typical messages in different information transaction situations when working on a new standard. Not having a hierarchal level above individual variables or elements probably makes it more difficult to build, understand and use the standard. It has therefore been deemed necessary to develop a set of messages for common communication needs.

The following types of messages have been defined (the use of some of the messages is illustrated in figure 1) using separate schemas:

- Product instruction
- Object instruction
- Species group instruction
- Object geographical instruction
- Forwarding object instruction (new 2010-06)
- Forwarding delivery instruction (new 2010-06)

- Harvested production
- Harvesting quality control
- Total harvested production
- Forwarded production
- Forwarding quality control (new 2010-01)
- Object geographical report (new 2010-02)
- Operational monitoring

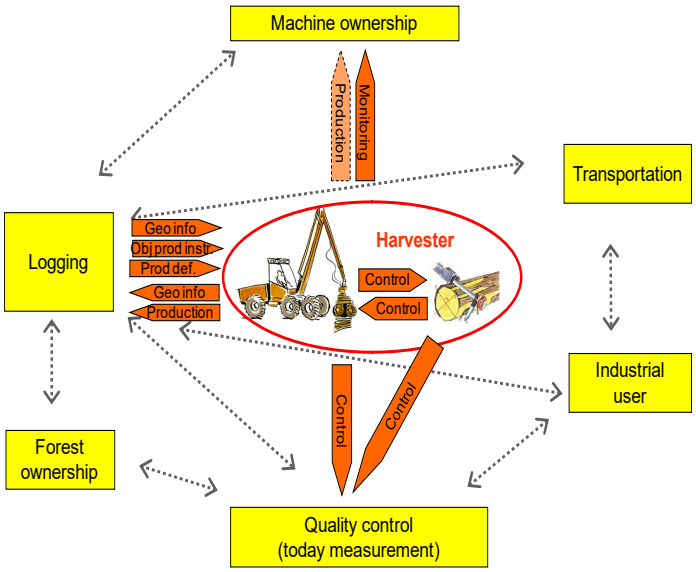


Figure 1. Illustration of how standardized messages are used in different information flows. The dashed arrows illustrates information flows that are based on other data standards as for example papiNet.

Below are some basic rules regarding StanForD messages:

- StanForD data will always, as a first step, be sent as one message from one single machine. It is thus logical to have the *Machine* as the highest hierarchal level within a message.
- StanForD messages can be used for merging data from several machines, but this is a secondary objective. Merging data is to be carried out by separate administrative software applications.
- This means that production data from several machines and several objects/sub-objects can be included in the presented structure. However, normal production reporting will be per machine and harvesting object.
- The different messages are to be kept clearly separated. This for example means that quality control data is always registered in the *harvesting quality control message* and not in the *harvested production message*.

Observe that the data structures should be flexible, making it for example possible to use the structures for merging harvester and forwarder production into one single message covering a specific harvesting object. However these "merged" messages will not be defined in StanForD 2010.

The idea has been to construct reusable structures (groups of variables) in the different messages. This means that we for example use exactly the same way of defining a harvesting object in all types of messages. These structures are drawn as boxes in the diagrams in sections *Messages sent to machines* and *Messages sent from machines*. An example of such a structure is the *ObjectDefinition*. All of these common structures are stored in one single xml schema (*StanForD 2010CommonDefinitions*)

Priorities

It will be necessary to implement a priority system for different elements since a message structure as described above has been implemented.

A possible solution is that a set of variables within a certain structure/group must be included (mandatory). It is probably also necessary to implement some basic rules concerning which structures/groups are mandatory within a certain type of message.

Examples illustrating priorities:

- A message for controlling bucking (*object instruction*) must include references to what products (assortments) are to be used
- A *harvested production message* must include data about harvested logs
- Length and diameter classes must be included when defining a product (*ProductDef* in diagrams below).

It should be possible for the user/operator to set what complementary variables are to be included in any message.

Naming conventions

Manual handling of separate files will probably still be needed in a foreseeable future. An operator will for example need to be able to save two separate files for monitoring and production data on a USB memory and carry it to an office computer in areas where no wireless communication possibilities exist. A standardized naming convention for messages is therefore needed when saving data in separate files.

A system with file extensions, similar to the old StanForD, will be used (table 1). It will make it possible to easily associate different files with different applications.

Table 1. The following file-extensions will be used for the new StanForD files

Extension	Name	Comment
".pin"	product instruction	comparable with ap1
".oin"	object instruction	comparable with apt and oai
".spi"	Species group instruction	comparable with apt and ap1
".ogi"	object geographical instruction	comparable with ghd
".foi"	forwarding object instruction	new
".fdi"	forwarding delivery instruction	new
".hpr"	harvested production	comparable with pri
".hqc"	harvesting quality control	comparable with stm and ktr
".thp"	total harvested production	very simple prd
".fpr"	forwarded production	comparable with prl
".fqc"	forwarding quality control	new
".ogr"	object geographical report	comparable with ghd
".mom"	monitoring data	comparable with drf

Extension	Name	Comment
".env"	Xml envelope, including other files	new
".udi"	User defined data instruction	new

It would be an advantage to know from the filename whether e.g. an hpr is zipped or not since StanForD 2010 files are regularly compressed. A "z" should be added to the file extension of individual StanForD files that are compressed. A compressed hpr-file would thus be given the file extension ".hprz"

ROAD MAP

Schema status

A corner stone when it comes to versioning of StanForD 2010 is the fact that we have included the status of a certain version in the schemas.

The following status levels are included in StanForD 2010:

- **Draft** can be used for testing and demonstration at any time by any one.
- **Release** is the final schema version never to be modified, a pre-release becomes a release after a testing period of normally 2-3 months.
- **Pre-release** is only created or modified after a meeting and based on decisions taken at a meeting. Once something has been added to a pre-release we have to take a formal independent decision at a subsequent meeting in order to exclude or modify it. A pre-release becomes a release after a testing period of 2-3 months.

Below is the rule from StanForD_2010_naming_and_design –document.

[Rule 8.2] The xsd:schema version attribute MUST use the following template:
<xsd:schema ... version="draft" |" pre-release" | "release"_<major>.<minor> >
Where:
draft | pre-release | release – is used based upon the status.
<major> - sequential number of the major version.
<minor> - sequential number of the minor version.

An attribute versionDate is also included in two different locations.

A timestamp has been added to attributes of schema headers:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:skogforsk:StanForD 2010"
xmlns:doc="urn:skogforsk:StanForD 2010:doc"
xmlns:sfd="urn:skogforsk:StanForD 2010"
targetNamespace="urn:skogforsk:StanForD 2010"
elementFormDefault="qualified"
attributeFormDefault="unqualified"
version="release_2.1"
sfd:versionDate="2012-09-06">
```

An attribute versionDate can be found in the attribute list of each message in (will be mandatory in version 3.0):

```
<xsd:attributeGroup name="MessageAttributeGroup">
    <xsd:attribute name="areaUnit" type="AreaUnitType" use="required"/>
```

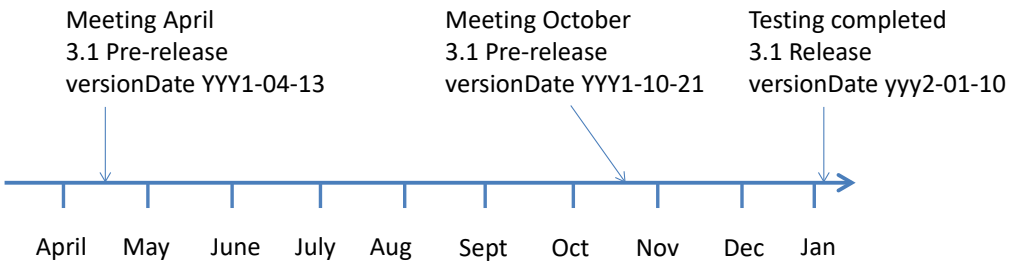
```
        <xsd:attribute name="diameterUnit" type="DiameterUnitType" use="required"/>
        <xsd:attribute name="lengthUnit" type="LengthUnitType" use="required"/>
        <xsd:attribute name="volumeUnit" type="VolumeUnitType" use="required"/>
        <xsd:attribute name="weightUnit" type="WeightUnitType" use="required"/>
        <xsd:attribute name="version" type="StanForD 2010VersionType" use="required"/>
        <xsd:attribute name="versionDate" type="xsd:date" use="optional"/>
</xsd:attributeGroup>
```

Release schedule

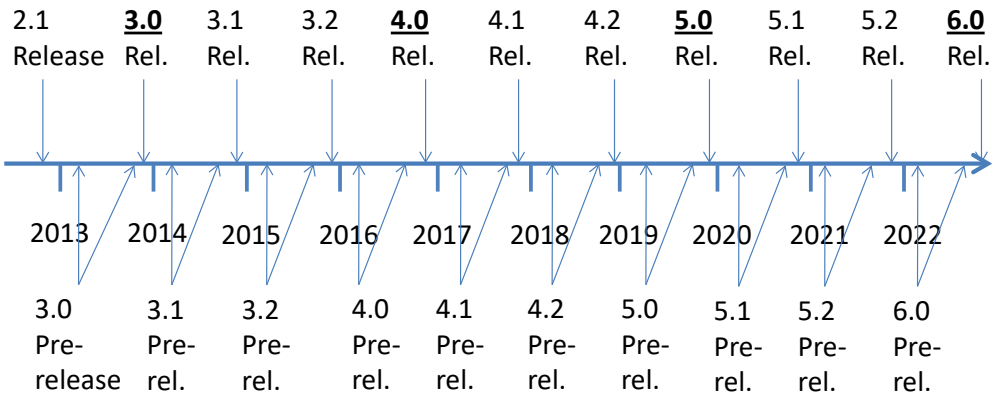
The basic rules for updating StanForD 2010 will be:

- Never more than one new ”release” (minor or major) per year.
- Maximum number of major releases is once per three years.
- Two meetings: March and October
- All decisions at meetings are included in “pre-release”. No other changes allowed in “pre-release”.
- A “pre-release” will become a “release” after a two/three month testing period.

The figure below illustrates the normal updating of StanForD 2010 during a year assuming that we use status type pre-release and versionDate attribute.



The rules can also be illustrated by the following figure below.



IDENTIFICATION

Unique identification of different objects like a machine, a product (assortment), a harvesting object, a tree species, a log etc is a weak area within the present standard. This is at least partly dependent on the fact that no one could imagine how and to what extent forest machines were to be integrated in

other information systems when the first standard was developed during the late 1980´s.

The old id-variables can be divided into two basically different groups. The variables can be:

- Set by machine (var270_t3, var306_t1, var121_t6)
- Set by logging organization (var21_t1 and var121_t2)

It has been decided that the following two new identity concepts are defined in StanForD 2010:

- **Key** is always set automatically in forest machine
- **UserId** is set by user of data (logging organization, operator, machine owner)

Key

A *key* is always set automatically in the machine. It is never to be generated by a logging organization, an operator, a machine owner or any other user.

A *key* is normally a running number to be used for harvesting object, stem and log (compare with logs and stems in present pri-file). Each machine, operator, product, tree species group, location, delivery and load is also to be given a unique *key*. The *keys* for stems, location, delivery, loads, operators, products and tree species group must be unique per machine and they are never reset, while the *key* for a log is reset for each stem.

A unique machine *key* is also needed, perhaps based on some kind of computer id. The machine *key* must be a globally unique identifier (GUID). It may also in some way include for example chassis no and manufacturer. The machine *key* must be hard coded by the manufacturer. It should normally be changed if for example the memory units of the bucking computer are replaced since the computer must keep track of historical data, for example the previous harvesting object or stem *key*. Observe that there exists support within Windows to generate GUID.

A certain combination of *keys* must always be unique (for example machine – stem – log). All logs from all harvesters will then have a unique id. The total production of all arvesters could then be collected in one data base without getting any id-conflicts. A better example from the old standard is when analyzing ktr-files from several different machines, harvesting objects and forest companies together.

> *A unique identity per stem is needed when checking for multiple occurrences of the same stem. It is today very difficult to identify an individual stem since there are no fool proof way of identifying machines and harvesting objects. A very large number of different variables has been used, Skogforsk has in these analyses for example used a combination of:*
> *- object id (var21_t1, var32_t2)*
> *- start date (var16)*
> *- save date (var12),*

- machine no (var3_t1/t2)
- caliper id (var3_t4)

The following keys will be used in order to identify a log if we follow the suggestions above:

- A globally unique machine/computer id (*MachineKey*)
- A running (sequential/serial) number to be used for stem and log (*StmKey* and *LogKey*)

A machine unique id to be used for harvesting object (*ObjKey*) could be a running number. This variable will make it possible to identify the harvesting object of a certain stem in the example above.

Any changes or modifications to a product (assortment) must mean that the *ProductKey* is updated. This is also true for species groups and operators.

Summary

A *key* is always set automatically in the machine.

A *key* is in most cases a running number

All *Keys* must be updated when:

Any modification is done
and
Definition has been used in harvesting

Exception: ObjectKey, SubObjectKey, LocationKey are only updated when ObjectDefinition, SubObjectDefinition and LocationDefinition are created. The reason for these exceptions are that it must be possible for the user of data to update for example an ObjectUserId-element and still be able to keep track of all logs from that harvesting object using the ObjectKey. This may for example happen if the harvesting is started "manually" (without an oin-file) before ObjectUserId has been communicated with the operator.
This exception will mean that we may not have any perfect historical log regarding modifications as we do have with for example products.

Table 2. Description of all keys in the standard

Element name	Data type	Description
MachineKey	String	Machine specific globally uniqe identity (GUID). Must be updated if memory of previously used Keys are lost. Possible for manufacturers to use this in order to identify indivudal machines. Other users of data should use MachineUserId or MachineIdOwner.
StemKey	Positive integer	Ascending number for each stem, must be unique for specific MachineKey. Machine specific identity per stem.
LogKey	Positive integer	Running number must be reset for each stem, LogKey 1 is the first log in a stem (butt log). Log identity per stem.
StemBunchKey	Positive integer	Ascending number for each stem, must be unique for specific MachineKey
ObjectKey	Positive integer	Ascending number set when new harvesting object is created in machine, never to be modified after creation
SubObjectKey	Positive integer	Ascending number set when new harvesting sub-object is created in machine, never to be modified after creation
LocationtKey	Positive integer	Ascending number set when new location is created in forwarder, never to be modified after creation

OperatorKey	Positive integer	Ascending number for each operator, updated if OperatorDefinition is modified in any way, must be unique for specific MachineKey
ProductKey	Positive integer	Ascending number for each product, updated if ProductDefinition is modified in any way, must be unique for specific MachineKey
SpeciesGroupKey	Positive integer	Ascending number for each species group, updated if SpeciesGroupDefinition is modified in any way, must be unique for specific MachineKey
DiameterSectionKey	Positive integer	Ascending number for each product, updated if DiameterSectionDefinition is modified in any way, must be unique for specific MachineKey
DeliveryKey	Positive integer	Ascending number for each species group, updated if DeliveryDefinition is modified in any way, must be unique for specific MachineKey

UserId

Forest companies or other users of data use *UserId* for identification purposes. *UserIds* may be used for harvesting objects, products, tree species~~, operators,~~ deliveries and machines.

The basic idea is that only one instance of a specific *UserId* should be available to use in a machine at a time. An old product or species group is for example to be replaced if a new product or species group with the same *UserId* is received by a machine. This means that for example only one instance of *ProductUserId* can be selected and used at a given time

Examples of *Keys* and *UserIds*

1. A new product "SawTimber" with *ProductUserId* "0110_xvy" is sent to a harvester.

2. A *ProductKey* is automatically given by harvester. The *key* is a running number, meaning that it is given a number representing the total number of products that has been sent to the specific machine.

3. Harvester application checks if any product with *ProductUserId* "0110_xvy" exists (is available) in harvester:

    a. No product with *ProductUserId* "0110_xvy" exists
    → new product added to "available products"

    b. Product with id *ProductUserId* "0110_xvy" already exist
    → old product replaced by new product added as an "available products"

The following diagrams (2-4) illustrates another similar example of how *UserIds*, *ModificationDates* and *Keys* are handled in a harvester when updating and using products. The following example is very similar to the Apteri-system as presently used in Finland. Observe that both *UserId* and *ModificationDate* are to be considered when updating the list of product (figure 3). An existing product is not to be replaced by a new product sent to the harvester if they have identical *UserId* and *ModificationDate*. The operator must have the possibility to deny the replacement of an existing product.

Step 1) New products sent to harvester

Name	UserId	ModifcationDate	
Pulpwood	PW	08-11-24 08:00	Already exists!
PineSaw_Cl1_b	PS_Cl1	08-11-25 11:00	Already exists!
PineSaw_Cl4	PS_Cl4	08-11-25 11:22	New!

List of products in harvester

Name	UserId	ModificationDate	Key
PineSaw_Cl1	PS_Cl1	08-11-24 08:00	1
PineSaw_Cl2	PS_Cl2	08-11-24 08:00	2
SpruceSaw_Cl1	SS_Cl1	08-11-24 08:00	3
SpruceSaw_Cl2	SS_Cl2	08-11-24 08:00	4
PulpWood	PW	08-11-24 08:00	5

Figure 2. Three new products (pin) are sent to a harvester. The UserId of the first (PW) and the second product (PS_Cl1) already exists among the available products while the third product (PS_Cl4) is new.

Step 1) New products sent to harvester

Name	UserId	ModifcationDate
Pulpwood	PW	08-11-24 08:00
PineSaw_Cl1_b	PS_Cl1	08-11-25 11:00
PineSaw_Cl4	PS_Cl4	08-11-25 11:22

New list of products in harvester

Name	UserId	ModificationDate	Key	
PineSaw_Cl1b	*PS_Cl1*	08-11-25 11:00	*6*	Replacing PineSaw_Cl1
				Warning if new product is old!
PineSaw_Cl2	PS_Cl2	08-11-24 08:00	2	
PineSaw_Cl4	*PS_Cl4*	08-11-25 11:22	7	New product added
SpruceSaw_Cl1	SS_Cl1	08-11-24 08:00	3	
SpruceSaw_Cl2	SS_Cl2	08-11-24 08:00	4	
PulpWood	PW	08-11-24 08:00	5	No replacement of Pulpwood
				Same modification date!

Figure 3. The list of available products is updated. Observe that the old product PW is not replaced by the old since they have the same modification date.

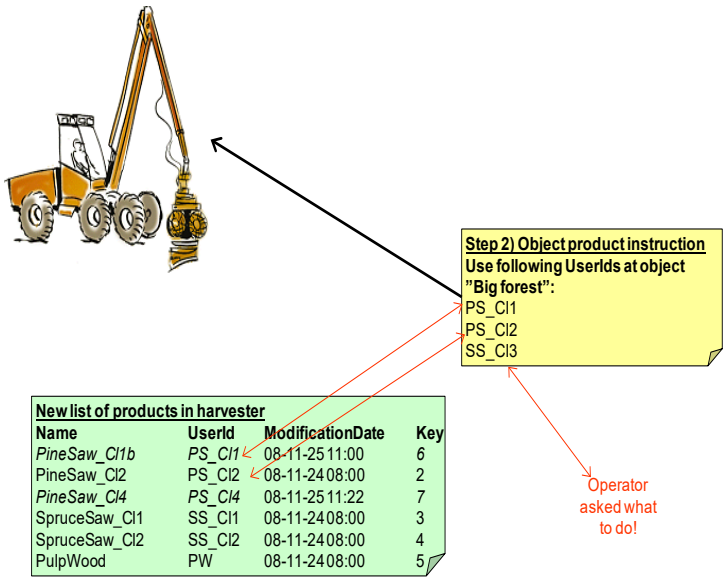


Figure 4. An object instruction (opi) is sent to the harvester. The instruction includes references to three products. Since the third UserId does not exist in harvester the operator is asked by the bucking application what to do.

The following diagrams (5-6) illustrates another similar example of how *UserIds*, *ModificationDates* and *Keys* are handled when updating and using object instructions. Observe that both *UserId* and *ModificationDate* are to be considered when updating the list of objects. An existing object is not to be replaced by a new object sent to the harvester if they have identical *UserId* and *ModificationDate*. The operator always must have the possibility to deny the replacement of an existing object as well as an update of the active object presently in use.

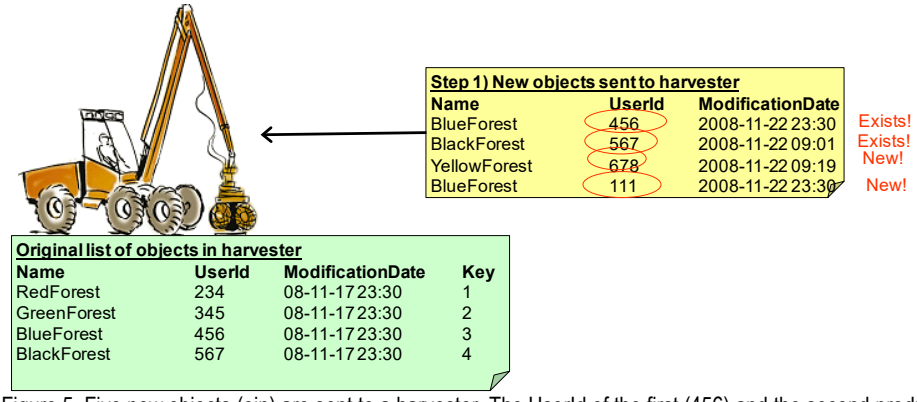


Figure 5. Five new objects (oin) are sent to a harvester. The UserId of the first (456) and the second product (567) already exists among the object while the third and fourth object are new.

Step 1) New objects sent to harvester

Name	UserId	ModificationDate
BlueForest	456	2008-11-22 23:30
BlackForest	567	2008-11-22 09:01
YellowForest	678	2008-11-22 09:19
BlueForest	111	2008-11-22 23:30

New list of objects in harvester

Name	UserId	ModificationDate	Key	
RedForest	234	08-11-17 23:30	1	
GreenForest	345	08-11-17 23:30	2	
BlueForest	456	08-11-22 23:30	3	Update active object definition in use
BlackForest	567	08-11-22 09:19	4	Replaced by new "BlackForest"
YellowForest	678	08-11-22 09:19	5	
BlueForest	111	08-11-22 23:30	6	New objects, added

Figure 6. The list of objects is updated. Observe that a second object named BlueForest now exist since the ObjectUserIds are different .

The following diagrams (7-9) illustrate another example of how *UserIds* and *Keys* are used in the case of production reporting from a harvester. Figure 8 specifically illustrates how the *MachineKey* and the *StmKey* can be used in order to identify multiple occurences of the same stems.

Day 1, production reporting
MachineKey: BM_450_2312hj21

ObjUserId	ObjKey	StemKey
Forest_333	9	1
Forest_333	9	2
Forest_333	9	3

Day 2, production reporting
MachineKey: BM_450_2312hj21

ObjUserId	ObjKey	StemKey
Forest_333	9	4
Forest_333	9	5
Forest_333	9	6

Figure 7. Production data sent from harvester on the same site day 1 and 2. Each stem has a unique StemKey and can thus be added to a production database.

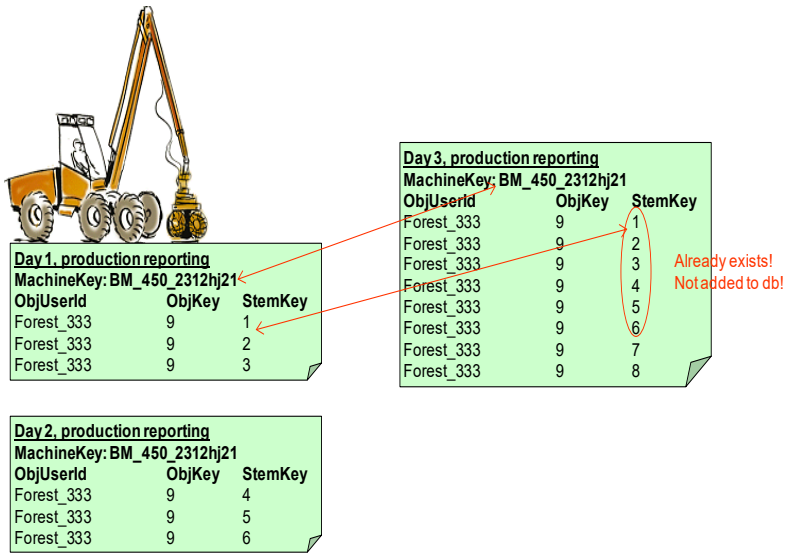


Figure 8. Production data sent from harvester also on day 3. It can be concluded by analyzing all relevant keys that the first six stems were included by mistake on the 3rd day. These stems (1-6) should thus not be added to the production database since they were already added on day 1 and 2.

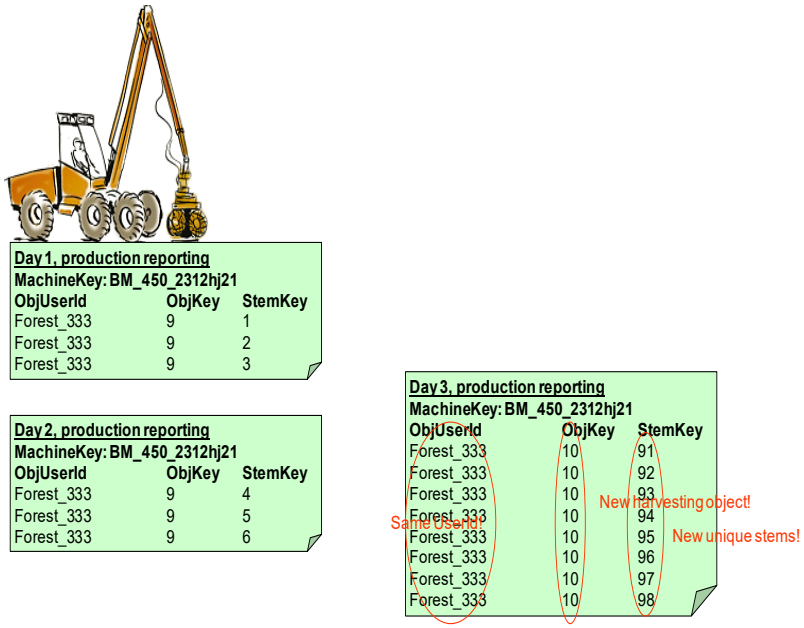


Figure 9. Production data sent from harvester also on day 3. It can be concluded by analyzing the StemKey that all eight stems are unique. By analyzing the ObjKey we can also conclude that the data comes from a new harvesting object. All stems should be added to the production database. The fact that the ObjUserId is the same as on day 1 and 2 is probably due to a mistake by the operator or by the person responsible for creating the harvesting directives.

Administrating pin- and spi-files

Open and read product instruction (pin)or species group instruction (spi). Do the following checks for all products in a new pin-file:

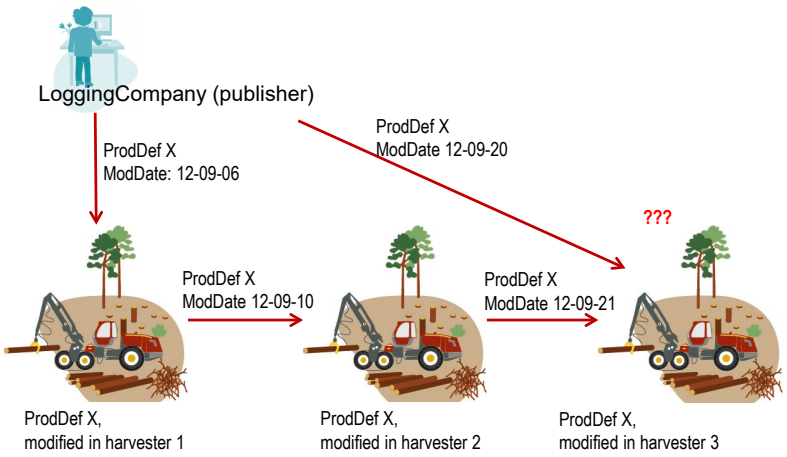
1) Check if new UserId is unique.
   a) Add to product database if unique and set new unique Key
   b) Available = true

2) Check if UserId and ModificationDate already exist.
   a) Do not add to product database

3) Check if UserId with different ModificationDate exist.
   Ask user whether new product should replace old product.
   If answer is OK then add new Product, set new Key
   a) Available = true for new Product
   b) Available = false for old Product

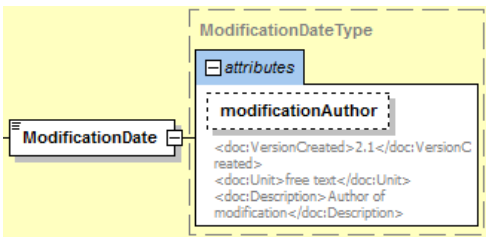
ModificationDate must always be updated if any modifications to ProductDefinition are carried out. No automatic update of *ProductDefinition* at ongoing site should be implemented; operator must have possibility to deny an updated product.

The structures mean that a ProductDefinition could be sent to a harvester and then modified in that harvester or another harvester several times. This case could be described in the following figure:



The only thing stated clearly is that if both ProductUserID and ModificationDate are the same there is no need at all for updating the "product database" in the harvester. The operator must always have the final decision when one product in the "database" is to be replaced even if the product in the machine is younger than the new one sent to the machine.

By using an optional free text attribute (ModificationAuthor), where it is stated who did the last modification, it is possible to inform the operator not only about when the product was last modified but also the author of these changes.



The operator could for example get the following question when importing a new product definition:

"Do you want to replace present product "Pine small sized timber" with product "Pine small sized timber"?
The present version was modified by "Holmen skog" at 2012-09-20 12:15:02.
The imported version was modified by "MaxiXplorer 941.1" at 2012-09-21 09:27:39.

HARVESTING OBJECTS

The smallest common denominator for a harvesting object is a harvesting contract with a forest owner within a certain geographical area and time. Sub-objects are only used for keeping production data from different sub areas apart. Implementing sub-objects could be viewed as giving the user the possibility to use a simple identity switch with no obligations to change object instruction.

A harvesting object:

- covers one business transaction/contract between a buyer and a seller
- one bucking and geographical instruction per object
- stem number is always reset when starting on new object

A sub-object:

- only used for keeping production data from separate geographical sub-units apart
- can be excluded

Messages sent to forest machine

This section describes all messages that are sent to a forest machine, normally from the logging organization. However, observe that the object geographical instruction message may also be updated and sent from the forest machine.

It has been decided to define three separate messages for controlling optimized bucking in harvesters. These two messages, *product instruction, species group instruction* and *object instruction*, are not to overlap each other. This means that the definitions of different products (assortments) are only to be included in the *product instruction* and not in the *object instruction*. The *product instruction* is comparable with the old ap1-file while the *object instruction* is comparable with the oai-file. Species specific information is registered in the *Species group instruction*, this information has previously been registered in the apt- or ap1-file. The old apt-file can be seen as a combination of these three new files.

An individual *object instruction, species group instruction* or *product instruction* message should function in all harvesters. Manufacturer specific settings must be avoided, meaning that these messages should not include any manufacturer specific data.

Some types of data presently included in the old apt-file that will be set in machine and thus not included in the *product instruction*:

- Machine and contractor id (var3, var34)
- Length of stem used in calculation (var101_t1)
- Length of stem measured before estimation (var102_t1)
- Upper and lower tolerance limit for deviation in estimated diameter (var103_t1 and var104_t1)

PRODUCT INSTRUCTION

The structure for *product instruction* primarily covers the present file types ap1, apm and fpm.

The *ProductKey* is used as a unique key in all production messages (running number set in machine, globally unique when used in combination with *MachineKey*). The *ProductUserId* is used as an identifier in *production definition* and *object instruction* messages (set by logging organization). Observe that the *ProductKey* is set in the machine and not by the logging organization.

No rules for changing or updating products within a certain harvesting object will be needed. It will be possible to do any changes or updates to the products if we implement the identification rules as described above. This is based on the assumption that log tallies are not generated any more and that a new *ProductKey* is given each time a product is updated or modified.

The *ProductKey* and the *ModificationDate* must be updated if a *ProductDefinition* is modified in any way. An attribute ModificationRestriction exists for each definition making it possible for logging organizations to restrict operator modifications.

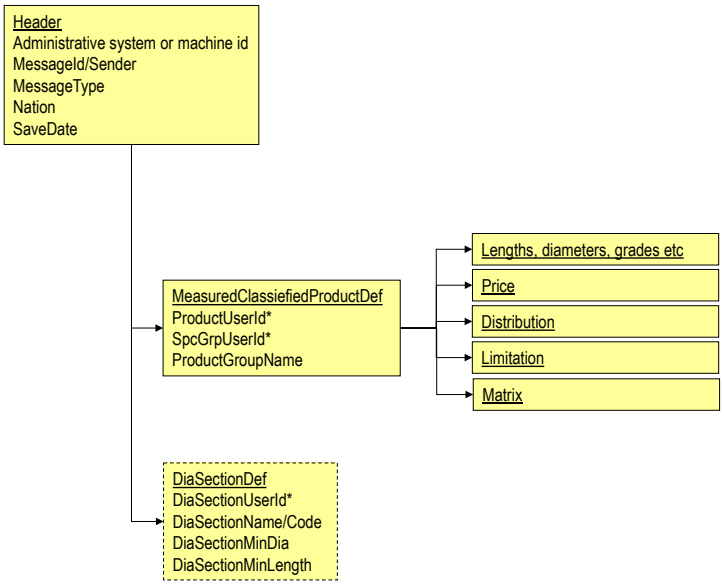


Figure 10. Layout of the product instruction message. No references to the harvesting object is included in this message. All Keys are set in the machine, only UserIds (*) are sent to the machine.

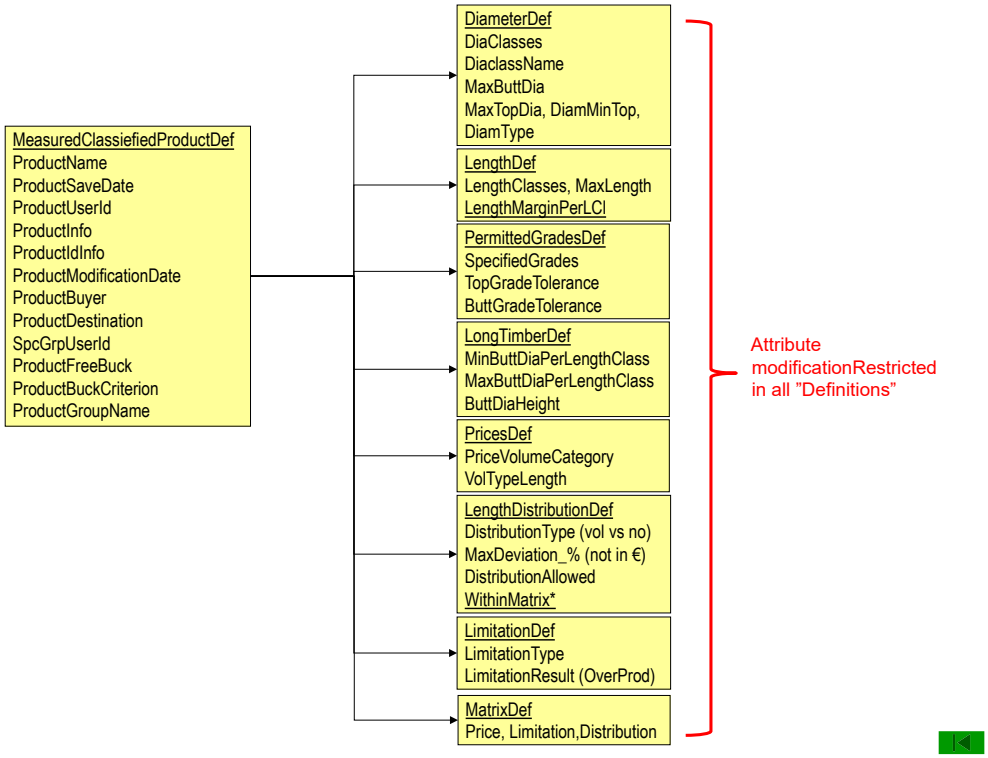


Figure 11. Detailed diagram describing the definitions of a product.
* New variable replacing FromMatrix (var197), if true only within optimal matrix, if false also use all other non-optimal matrixes in distribution bucking (true should be default)

Observe that there is only distribution matrixes per diameter class included in the diagram above, this means that only a desired length distribution within a diameter class is expressed. The possibility to include a distribution matrix with the desired distribution for the whole matrix (old var191_t1 and var191_t2) is excluded.

Price volumes

Present situation

Price volumes and categories is quite complex in old StanForD. All relevant variables in the present old standard are included below in table 3.

Table 3.Price volume variables in old StanForD standard

VarNo	Description	Unit	New group	New name
v164_t1	Principle for registered diameter 0 = Bucked length, cm 1 = Required length (length class) 2 = Bucked random length, dm	code	DiamDef	DiamType
V164_t4	Distance from log top	Cm		DiamTopPosition
v131_t1	Lower diameter limit. See also DiamClassesMax	Mm		DiamClasses
v131_t1	Maximum diameter use in combination with DiamClasses	Mm		DiamClassesMax
v134_t1	Primarily used together with volumes and classes dependent on mid diameter	mm		DiamMinTop
v134_t2	Maximum diameter in the large end of the log per price matrix	Mm		DiamMaxButt

VarNo	Description	Unit	New group	New name
v132_t1	Lower length limit of length class	Cm	LengthDef	LengthClasses
v132_t1	Maximum length use in combination with LengthClasses	Cm		LengthClassMax
v135_t1	Additional length margin, can not be a negative number /length class	Cm		LengthMargin-PerClass
v163_t1	Volume calculation based on: 0 = Bucked length, cm (default in Finland) 1 = Required length as per var132 2 = Bucked random lengths, dm	code	PriceDef	VolTypeLength
v161	Price category/price matrix/tree species where 1 = price/m3 (volume by small-end diameter); 2 = price/m3 (solid); 3 = price/log; 4 = price/m3 (Norwegian price category) 5 = price/m3 (Swedish top and butt end measuring); 6 = price/m3 (solid, measured at midpoint, price due to SED, HKS diameter) 7 = price/m3 (solid, measured at midpoint, price due to midpoint diameter, HKS diameter) 8 = price/m3 (solid, measured at midpoint, price due to midpoint diameter, (Danish) 9 = price/board feet (American price category) 10 = price/m3 (solid, diameter measured at midpoint, price due to SED) diameter in mm 11 = price/log (presented volume as code 4, Norwegian price category) 12 = Price/bundled m3 (bulk volume calculated with diameter and length of bundle) 13 = price/m3 (Estonian Nilson's volume unit) All the codes are described in detail in appendix (Old types 2-4 are supposed to be hardcoded)	code		PriceVolume-Category

Var161_t1 basically describes how to calculate a certain volume. A good example of the problems we have today is code 10 (M3sm) which has the following definition:
Price/m3. Solid volume, diameter (mm) measured at midpoint for calculating the volume, price due to small-end diameter (mm).

This price category not only describes how a volume should be calculated but also how it is to be classified based on its small end diameter. Other similar codes are 3,7,10 and 11.

New standard

To improve the situation the old variables were divided up into three separate element groups: DiameteDefinition, LengthDefinition and PriceDefinition. This means breaking the present var161_t1 up into several new variables according to table 4.

Table 4. Elements for defining diameter and length classification as well as pricing of logs

Group	**Name**	**Description**	**Unit**	**OldVarNo**
Diameter-Def	diamClassCategory (attribute)	Type of diameters in DiamClasses (used for determining cell in price matrix) Top, Midpoint	code	v161_t1
	DiamClasses-	Lower diameter limit. See also DiamClassesMax	Mm	v131_t1

Group	Name	Description	Unit	OldVarNo
	LowerLimit			
	DiamClassesMax	Maximum diameter. Use in combination with DiamClasses	Mm	v131_t1
	DiamClassAdjust	1=Log belongs to first diameter class smaller or equal to log diameter (227 mm => class 220 mm) 2=Log belongs to closest diameter class, normal rounding, (227 mm => class 230 mm)		
	DiameterUnderBark	Diameter classes under bark	boolean	
	DiamMinTop	Minimum diameter at small end. Primarily used together with volumes and classes dependent on mid diameter.	mm	v134_t1
	DiamMaxButt	Maximum diameter in the large end of the log per price matrix	mm	v134_t2
	DiameterTop-Position	Position from top end of log where top diameter is measured. Valid for diamClassCategory values "Top" and " Norwegian mid"	Cm	V164_t4
LengthDef	LengthClassAdjust	1=Log belongs to first length class smaller or equal to log length (427 cm => class 400 cm) 2=Log belongs to closest length class, normal rounding, (427 cm => class 430 cm)		
	LengthClasses-LowerLimit	Lower length limit of length class	Cm	v132_t1
	LengthClassMargin	Additional length margin, cannot be a negative number /length class. The additional length margin is ONLY used to avoid too short logs.	Cm	v135_t1
	LongLogButtMIN	Lower limit for butt diameter per length class and product<	Mm	v165_t1
	LongLogButtMAX	Upper limit for butt diameter per length class product	MM	v166_t1
	LongLogButtHeight	Height above stump of measuring point for butt diameter per product	Cm	v167_t1
	LengthClassMAX	Maximum length use in combination with LengthClasses	Cm	v132_t1?
PriceDef	VolumeDiameter-Adjust	Diameters in log price volume calculation "Measured diameter in mm" "Measured diameter rounded downwards to cm" identical with HKS (227 mm => 220 mm)	code	
	VolumeDiam-Category	Diameters used in price volume calculation: Solid volume "Top" "Mid" "Calculated Norwegian mid" "Calculated Estonian mid" "Mid diameter" is measured at a position defined according to VolumeLengthCategory.	code	v161_t1
	volumeDiameter-TopPosition	Position from top end of log where top diameter for volume calculation is measured. Valid for VolumeDiameterCategory values "Top", "Norwegian mid" and "Estonian mid"	cm	V164_t4
	VolumeLength-Category	Lengths used in price volume calculation: "Physical length, cm" (default in Finland) "Length as defined in LengthClasses" "Rounded downwards to nearest dm-module" (e.g. 427 cm => 420 cm) "Rounded to nearest mid-dm" (e.g. 427 cm => 425 cm)	code	v163_t1
	VolumeUnderBark	Price volume under bark: "True" or "False"	boolean	

Below are some examples of how the suggested new variables might be used for some of the old price categories as used in different countries (table 5).

Table 5. Examples of how to use the new elements based on the present price volume types

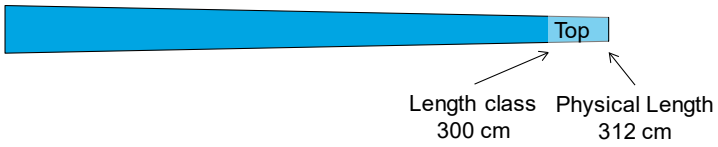
Name	Description	Swe sub	Swe to	Fin[1] sob	Nor[3] mid	Ger mid	Ger top	Den[2] mid	Est mid	Fra[4] mid
Old codes (var161_t1) Country code (var6_t1)		2 752	1 752	130 246	4 578	7 276	6 276	136 208	12 233	135 250
DiamClassCategory Diameters in DiamClasses (price matrix)	1=Top, 2=Midpoint	1	1	1	1	2	1	2	1	2
DiamClassAdjust	1=diameter class smaller or equal to log diameter (DiameterClassAdjustment1) 2=closest diameter class, normal rounding (DiameterClassAdjustment2)	1	1	1	1	1	1	1	1	1
DiameterUnderBark Diameter classes under bark.	True/false	True	True	False	True	True	True	False	True	False
DiameterTopPosition Position from top end of log where top diameter is measured.	Cm	10	10	0	10	0	0	0	0	0
LengthClassAdjust	1= length class smaller or equal to log length (LengthClassAdjustment1) 2=closest length class, normal rounding (LengthClassAdjustment2)	1	1	1	1	1	1	1	1	1
VolumeDiamAdjust Diameters in log price volume calculation	1=Measured diameter 2=Measured diameter rounded down to cm,	1	1	1	2	2	2	1	1	2
VolumeDiamCategory Diameters used in price volume calculation:	1 = Solid volume 2 = Top 3 = Mid 4 = Calculated Norwegian mid 5 = Calculated Estonian mid	1	2	1	4	3	3	3	5	3
VolumeLengthCategory Lengths used in price volume calculation:	1 = Physical length 2 = Length as defined in LengthClasses 3 = Rounded downwards to nearest dm-module 4 = Rounded to nearest mid-dm	1	2	1	3	2	2	1	1	2
VolumeUnderBark Price volume under bark.	True/false	True	True	False	True	True	True	False	True	False
volumeDiameter-TopPosition Position from top end of log where top diameter for volume calculation is measured.	Cm	10	10	0	10	0	0	0	0	0
BarkFunctionCategory	1=None 2=Swedish Zacco 3=German 4=Skogforsk 2004, Scots pine 5=Skogforsk 2004, Norway spruce	2/4/5	2/4/5	1	2/4/5	3	3	1	2/4/5	1

1. Same in Denmark
2. Used for long timber (whole stems)

3. Primarily used for normal saw logs. Volume is calculated based on a cylinder with a theoretical diameter on the middle of the log (M).
   Length: Length (L) is the total actual length in decimetres. If L is in centimetres it is truncated into decimetres (class bottom).
   Diameter: By definition a diameter class is 1 cm (independent of the actual price matrix). Registered diameter (Dt) is measured 10 cm from top. If Dt is in millimetres it is truncatedinto centimetres (class bottom). To get the diameter on the middle of the log (Dmid) you use the formula: Dmid = Dt + (L/2 * 0.1) + 0.5
   Example: L = 518 cm, Dt = 223 mm
   Dmid = 22 + (51/2 * 0.1) + 0.5 = 25.05 cm
   Volume: Volume is calculated in dm3.
   PI = 3.14
   V = ((Dmid/10)*(Dmid/10)) * PI/4 * L = ((25.05/10)*(25.05/10))*PI/4*51 = 251 dm3
4. Name of the French volume type must be: "NF B53.020 français" /"French NF B53.020"

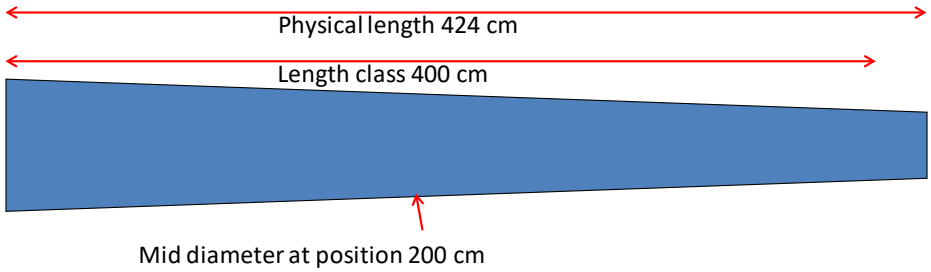
VolumeLengthCategory and diameters

The top part of a log is to be excluded from the price volume when "VolumeLengthCategory" has the value "LengthClass" and "VolumeDiameterCategory" has the value "All diameters (solid volume)" as illustrated below:



Volume categories "m3sub" or "m3sob" in hpr, mom and hqc must be based on the physical length of the log as specified in the annotation of LogVolumeCategory.

The value "Mid diameter" in VolumeDiameterCategory is measured at a position defined according to VolumeLengthCategory as illustrated below:



Cutting window and length margin

Background

The present StanForD variables are:

- Length margin (var135_t1)
  Additional length margin, can not be a negative number/length class/price matrix/tree species

- Cutting window (var135_t3)
  Lower length limit for "cutting window"/price matrix/tree species. Lower length class limit (var132) and variable 135, type 1 and 3 together, define the length class of a log if lower limit of the cutting window is below lower length class limit

- Cutting window (var135_t4)
  Upper length limit for "cutting window"/price matrix/tree species. It does not affect length classification of a log. It can not be above lower length class limit (132_t1) nor above lower length limit for the "cutting window" (135_t3), of the next length class.

Lower length class limit (var132) and variable 135, type 1 and 3 together, define the length class of a log if lower limit of the cutting window is below lower length class limit

Upper length limit does not affect length classification of a log. It cannot be above lower length class limit (132_t1) nor above lower length limit for the "cutting window" (135_t3), of the next length class

New standard

The purpose of the length margin and cutting window is:

- the additional length margin is used to avoid too short logs

- the cutting window is to give the harvester a suitable interval where the head can stop feeding the stem and thus to easily find a cutting position without having to feed the stem back and forth.

The following elements in table 6 are included in the pin-message (no significant changes when comparing to present standard).

Table 6.Elements defining cutting window

Group	Name	Description	Unit	OldVarNo
LengthDef	LengthClass-Margin	Additional length margin, cannot be a negative number /length class/price matrix/tree species The additional length margin is ONLY used to avoid too short logs.	Cm	v135_t1
Cutting-WindowDef	LowerLength-Limit	Lower length limit for "cutting window" per product. LengthClassLowerLimit (var132), LengthClassMargin (var135_t1) and LowerLengthLimit together define the length class of a log if lower limit of the cutting window is below lower length class limit. Purpose of cutting window is ONLY to give the harvester a suitable interval where the head can stop feeding the stem and thus to easily find a cutting position without having to feed the stem back and forth.	Cm	v135_t3
	UpperLength-Limit	Upper length limit for "cutting window" product. It does not affect length classification of a log. It can not be above LengthClassLowerLimit (132_t1) nor above LowerLengthLimit for the "cutting window" (135_t3) of the next length class. Purpose of cutting window is ONLY to give the harvester a suitable interval where the head can stop feeding the stem and	Cm	v135_t4

Group	Name	Description	Unit	OldVarNo
		thus to easily find a cutting position without having to feed the stem back and forth.		

Examples

Observe that these examples are basically identical with what was decided at the StanForD meeting 2002-10-28.

It is assumed that LengthClassAdjustment is "length class smaller or equal to log length" in figure 12-15 while it is assumed that LengthClassAdjustment is "closest length class, normal rounding" in figure 16-17

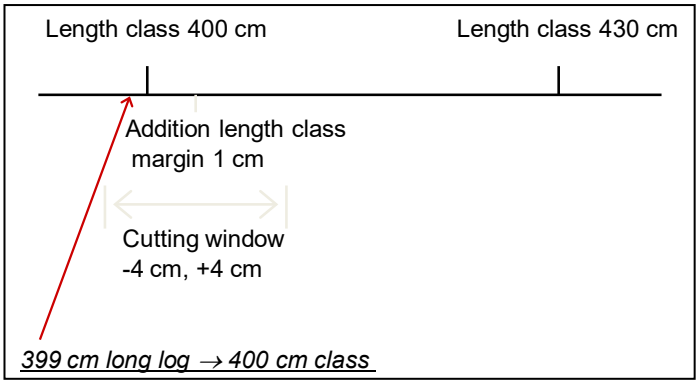


Figure 12. A log with a length of 399 cm will be registered in the 400 cm class (LengthClassAdjustment = "length class smaller or equal to log length")

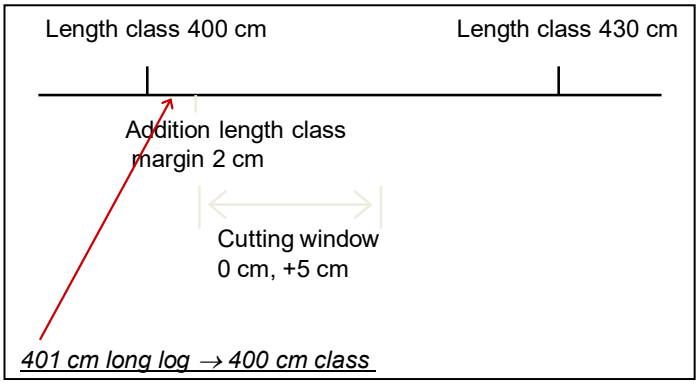


Figure 13. A log with a length of 401 cm will be registered in the 400 cm class while a log with a length of 399 cm will not be registered in the 400 cm class (LengthClassAdjustment = "length class smaller or equal to log length")

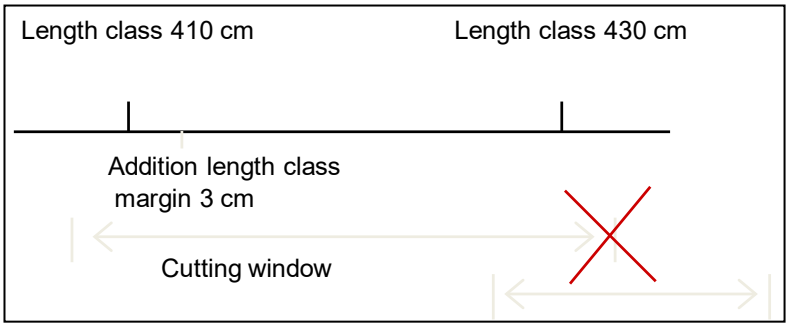


Figure 14.This is not a correctly defined "cutting window" since the upper length limit is above the lower length limit or above the lower length limit for the "cutting window", of the next length class (LengthClassAdjustment = "length class smaller or equal to log length").

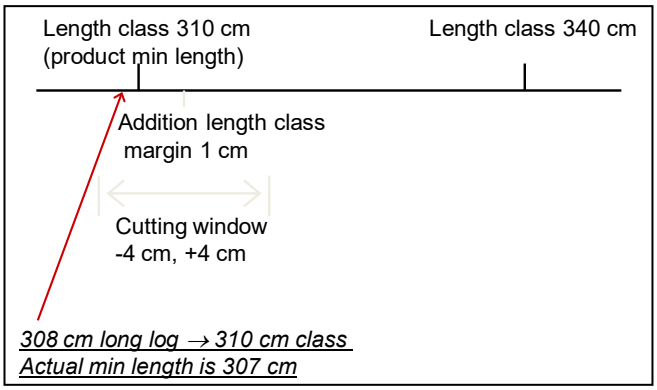


Figure 15.Length class 310 cm is the minimum length for the present product. This means that the "true" minimum length is actually 307 cm when using this kind of negative cutting window (LengthClassAdjustment = "length class smaller or equal to log length").

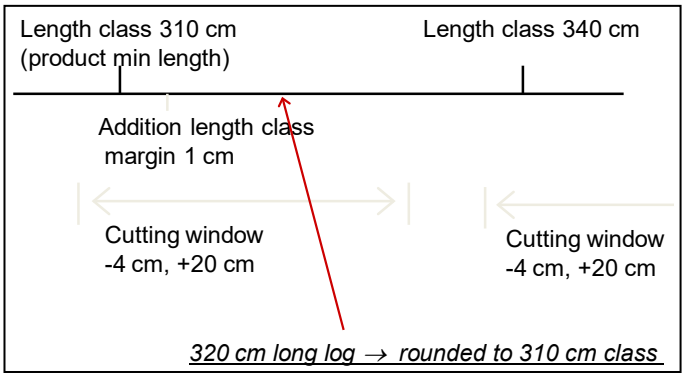


Figure 16.Rounding is done towards 307 and 337 cm. (LengthClassAdjustment = "closest length class, normal rounding")

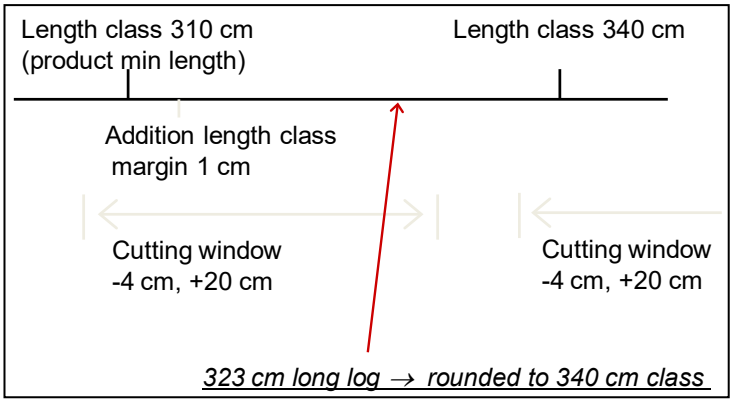


Figure 17. Rounding is done towards 307 and 337 cm (LengthClassAdjustment = "closest length class, normal rounding").

The following rules must be used:

1. Negative cutting window always decreases the class length, both when LengthClassAdjustment is equal to "length class smaller or equal to log length" and when equal to "closest length class, normal rounding".

2. Rounding always have a higher priority than cutting window (figure 17)

3. Upper limit of cutting window must not be above lower length class limit (132_t1) nor above lower length class limit for the "cutting window" (135_t3) of the next length class.

SPECIES GROUP INSTRUCTION

Within the old standard species are basically identified through the order which they are listed in the apt-file. The new species group definition is described in figure 18 below. Since the species in old standard files in some cases are individual botanical species (for example Norway spruce) while sometimes several botanical species are merged together (for example Other broadleaves) it is probably impossible to come up with one single standard solution (for example a list of all relevant species). It is therefore suggested that the user of StanForD decides on his own what groups of species he wants to use. It may however be a good idea to decide on a few default species groups per region or country that can be described within the standard. An example of how to manage species groups is illustrated in figure 19.

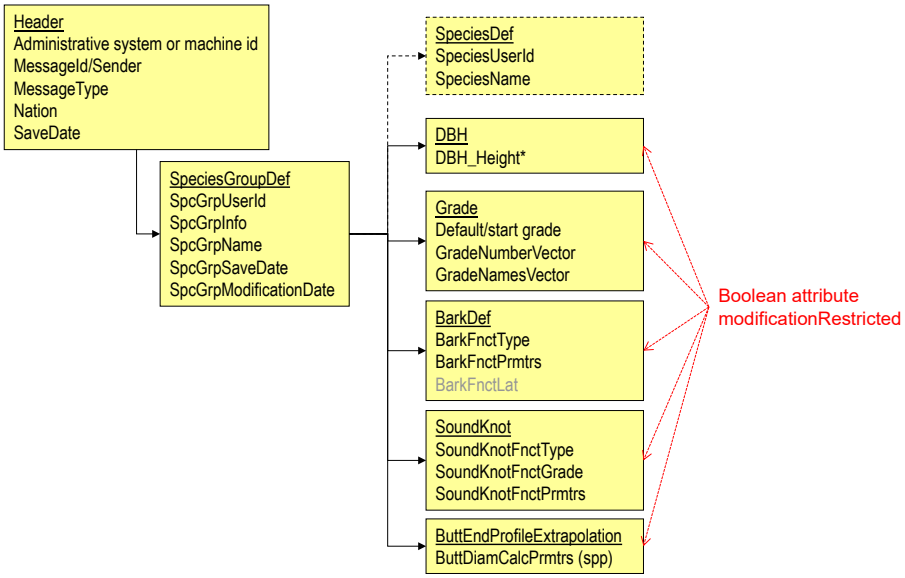


Figure 18. Diagram describing the definitions of a species group. Dashed arrows and borders indicate that a structure is optional.

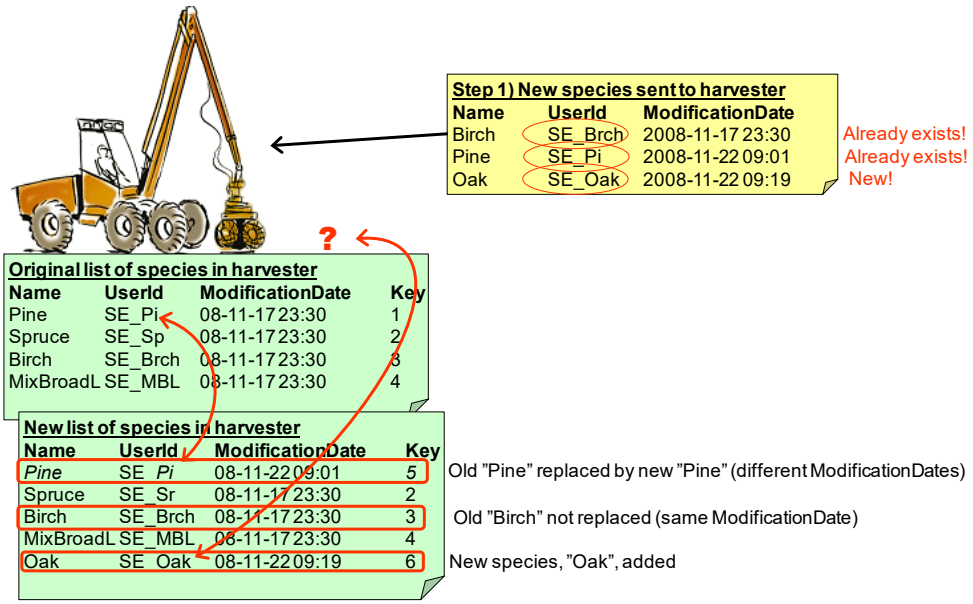


Figure 19. Two new species groups are sent to harvester. The old Pine is replaced by the new Pine since they have the same UserId, the old Birch is not replaced since it has the same modification date while the Oak group is added as a new group.
Two examples of how to handle calibration data and calibration history for Oak:
Case 1) Operator knows there is a lot of oak in mixed broadleaves. He therefore wants Oak to use same data as mixed broadleaves and to calibrate them together in the future.
Case 2) Operator knows there are no oaks in calibration database. He therefore wants Oak to be calibrated separately now and in the future.

Sound knot function

The model for automatic bucking of sound-knot saw logs is based on a relationship between *DBH* and the small-end diameter (*SED*) of logs in order to produce sound-knot sawn wood in the centre boards of the logs.

The sound knot cylinder (SKC) represents the *largest small-end diameter* of the lowest possible log meeting the grade requirements, assuming a cylindrical sound-knot core. Below this level, there will be too many loose (encased) knots on the sap-wood side of center boards. The following function is implemented to predict the *largest small-end diameter (SKC) for sound knot logs*:

SKQ = (constant A + factor B * DBH + factor C * DBH^2) * tolerance D

SKC = SKQ *DBH

SKQ = Sound knot quotient used to determine largest allowed small end diameter of sound knot log.
SKC = Sound knot cylinder diameter
A = ConstantA for determining the limit for sound knots
B = FactorB for determining the limit for sound knots
C = FactorC for determining the limit for sound knots
D = ToleranceD for dry knots within calculated limit for sound knot cylinder diameter (normally close to 1)

SoundKnotFunctionGrade = The grade given to logs when the small end diameter is smaller than SKC diameter.

GradeIncluded =Grades that can be replaced by the automatically calculated SoundKnotFunctionGrade. Means that if a section of stem does not have a grade included in this element the SoundKnotFunction will NOT affect the bucking optimization.

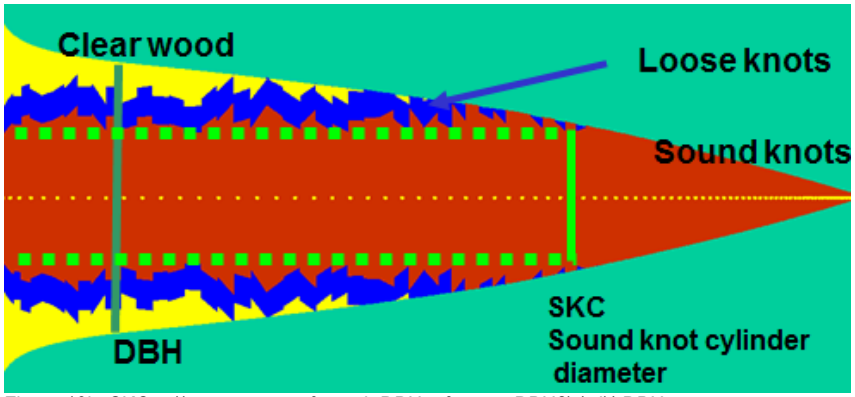


Figure 19b. SKC = ((constant a + factor b DBH + factor c DBH2) * d)* DBH.
All logs with a top diameter above SKC (smaller top end diameter) will be classified as sound knot logs (except if the operator manually selects a different quality for the log).

Butt end profile extrapolation

The butt end extrapolation can be defined in two separate ways in the spi-file, either the parameters of the functions or a table can be registered in the file. Some manufacturers require the parameters while some other manufacturers require the table.

Attribute buttEndProfileExtrapolationMethod in the spi-file includes the following enumeration list:

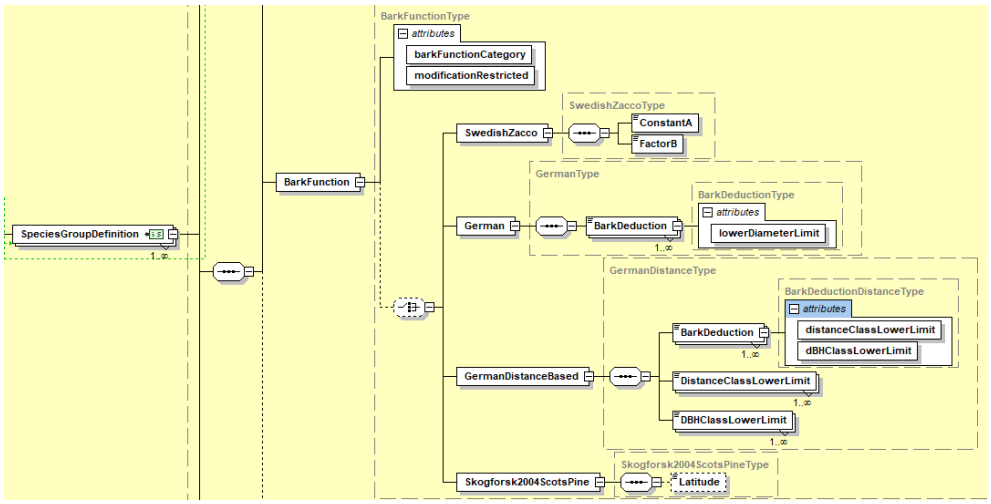
- ExtrapolationFunction (only element ButtEndProfileExtrapolation-Function is used)
- ExtrapolationTable (only element ButtEndProfileExtrapolationTable is used)
- Both(both ButtEndProfileExtrapolationFunction and ButtEndProfileExtrapolationTable are used)

Since different manufacturers use different methods it is strongly recommended that all spi-files sent out to harvesters include both methods. This means that attribute buttEndProfileExtrapolationMethod must have the value "Both". It is also required that all harvesters support the value "Both".

Bark functions

It should be possible to control what bark function to use through the spi-file and to register what function has been used in the hpr-file.

The following variables for bark functions exist today in StanForD:



Allowed values:"None","Swedish Zacco","German", “GermanDistanceBased”, “Skogforsk 2004, Scots pine" and "Skogforsk 2004, Norway spruce". If this attribute is "None" the BarkFunction element is empty

Description of bark function categories:

Swedish Zacco

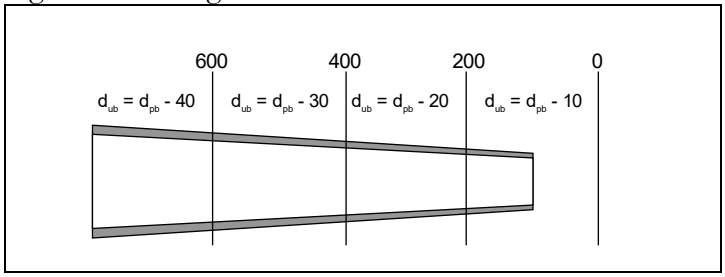
Bark function developed by Zacco (1974).
Linear function: *Double bark thickness = a + b * top diameter ob*
where *a* is stored as the first parameter in var113_t1 and *b* as the second parameter.
For example: ”Bark= 3,28+0,0370*diam” is stored as ”113 1 328 370~” in a StanForD file.

German

Bark function based on diameter classes with fixed bark deductions (double), based on German requirements. Example:

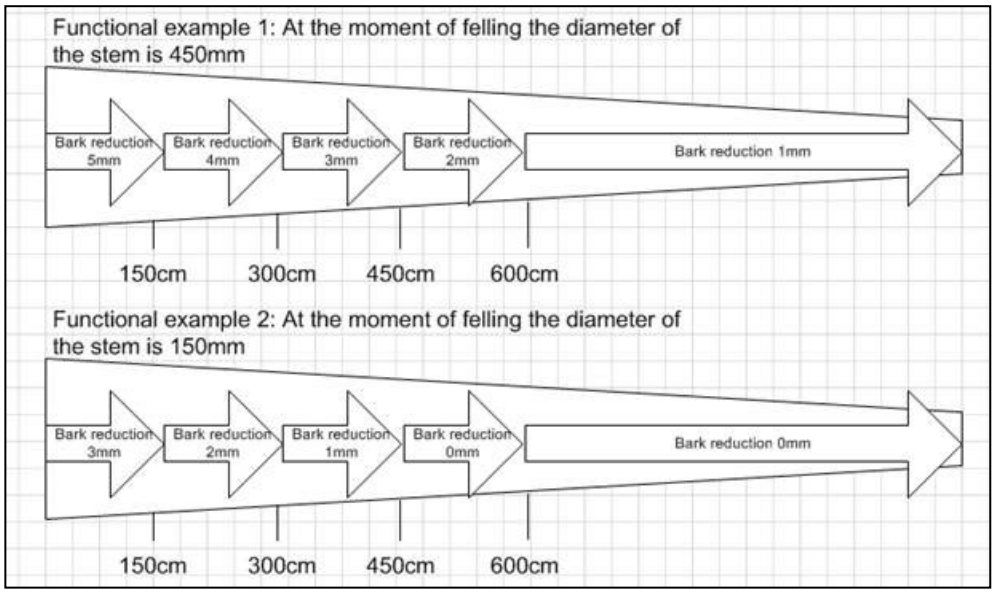
BarkDeduction (mm)	lowerDiameterLimit (mm)
40	>=600
30	<600 >=400
20	<400 >=200
10	<200 >=0

Figure illustrating how diameter under bark is calculated:



GermanDistanceBased

Bark function based on distance from stump and DBH with fixed bark deductions (double), based on German requirements.
The following figure illustrates how to implement this bark function (observe that the diameters 450 and 150 are DBH values):



Skogforsk 2004, Scots pine

Function developed by Skogforsk (2004) for Scots pine (Pinus Sylvestris)

```
dbh_b=min(dbh,480)      /* DBH maximum 590 mm. */
htg=-ln(0.12/(72.1814+0.0789*dbh_b-0.9868*lat))/(0.0078557-
  0.0000132*dbh_b)  /* Break point in cm calculated */
db=3.5808+0.0109*dbh_b+(72.1814+0.0789*dbh_b-0.9868*lat)*
  exp(-(0.0078557-0.0000132*dbh_b)*h)  /* Double bark thickness below
break point calculated, mm */
if h>htg then db=3.5808+0.0109*dbh_b+0.12-0.005*(h-htg)
/* Double  barkthickness above break point calculated, mm */
db=max(db, 2)    /* Bark thickness minimum 2 mm */
```

Skogforsk 2004, Norway spruce

Function developed by Skogforsk (2004) for Norway spruce (Picea abies)

```
db=0.46146+0.01386*dbh+0.03571*dia
/* Double bark thickness calculated, mm */
db=max(db, 2)    /* Bark thickness minimum 2 mm */
```

Db = double bark thickness (mm)

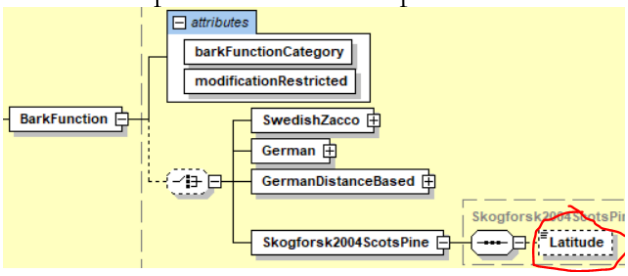
H = height from butt end of stem where bark thickness is to be calculated (cm)

Dbh = breast height diameter (mm)

Dia = diameter on bark where bark thickness is to be calculated (mm)

Lat = latitude (decimal degrees).

Observe that element BarkLatitude is NOT included in Swedish default spi. It must be simple to control and update BarkLatitude!



OBJECT INSTRUCTION

The structure for bucking control primarily covers the old file type oai.

The object instruction can include the same basic information as the old oai-file (figure 20). The definitions of products and species groups are sent in separate messages, *Product instruction* and *Species group instruction*, with no references to a harvesting object included.

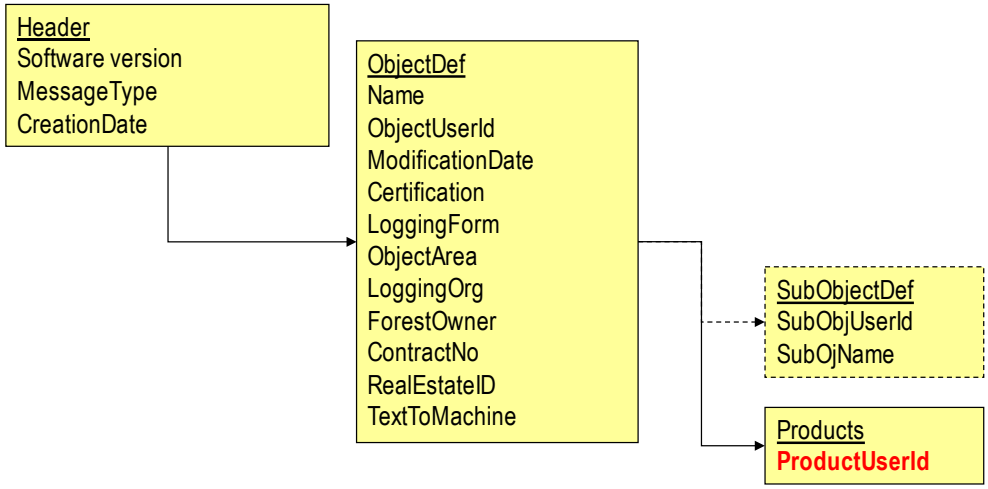


Figure 20. Detailed diagram describing the data per harvesting object sent out in an object instruction. Only references to different products are included in this structure (ProductUserId). Observe that this is how an object instruction will look like when sent to the machine, all Keys are set in the machine, only UserIds are sent to the machine.

OBJECT GEOGRAPHICAL INSTRUCTION

The structure for *object geographical instruction* is mainly a copy of the present ghd-file.

The structure of the *geographical instruction i*s similar to the *object instruction*. The *object geographical instruction* message includes references to what GIS-files (layers) are to be used instead of including references to what products are to be used at a certain harvesting object. This message also includes information about how the different layers are to be presented (formatting) and what information can be found in complementary data files (for example dbf-files).

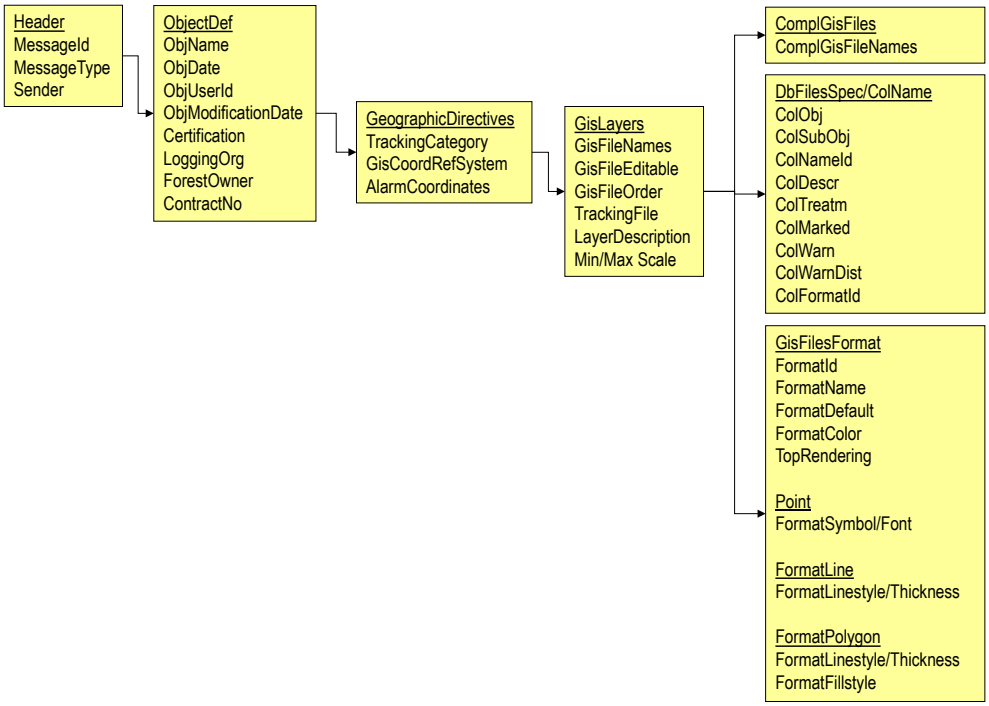


Figure 21. Diagram describing the GIS structures (basically copy of ghd-file). Observe that some of the variables in the diagram above are only set in machine after modification of a gis-layer.

The StanForD 2010 XML Envelope should be used to enclose additional files in case of ogi and ogr. Binary files must be embedded using Base64 encoding in the Stan-ForD2010 envelope. Observe that an "escape sequence" must be used for embedded but not encoded documents. This for example means that &gt is used instead of > and &lt is used instead of <. A file which is encoded means that it is included as a binary text string.

SUMMARY HARVESTER INSTRUCTIONS

Below is a diagram describing the three types of messages sent to a harvester. The different messages in diagram 26 below are simplified version that is intended to give the reader an idea of the main differences between these messages.

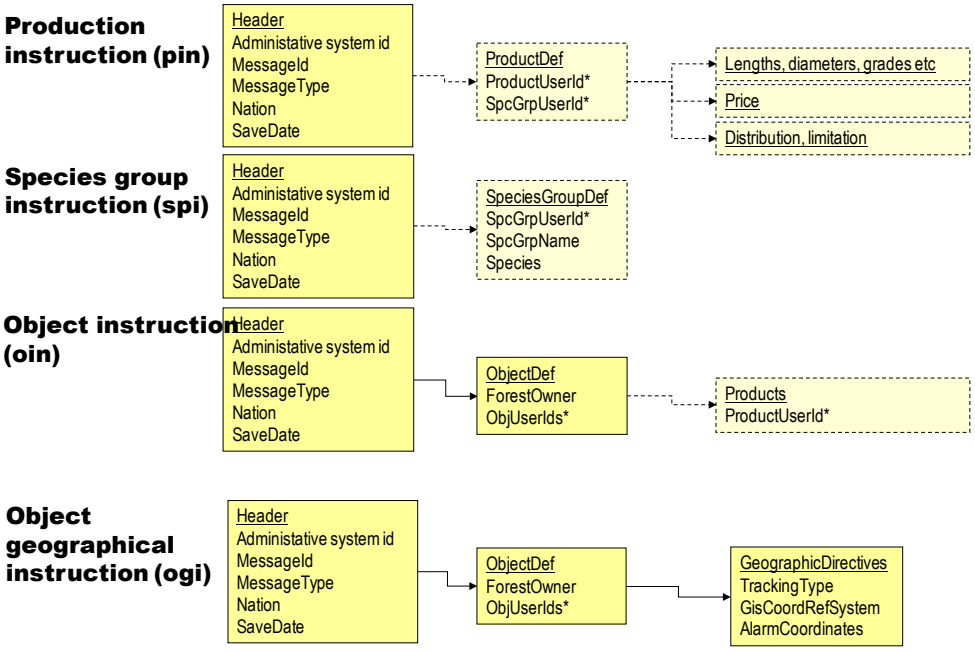


Figure 26. Diagram outlining the differences between the different messages that can be sent to a harvester. The product instruction is comparable with the present ap1-file, the object instruction may include information from the apt- or oai-files, the species group instruction includes information from ap1- and apt-file and the object geographical information is comparable with the ghd-file. Dashed arrows and borders indicate that a structure is optional.

FORWARDER INSTRUCTION

The structure for forwarding instructions is based on the forwarded production message as described in figure 22. Observe that machine specific information and production data are excluded.

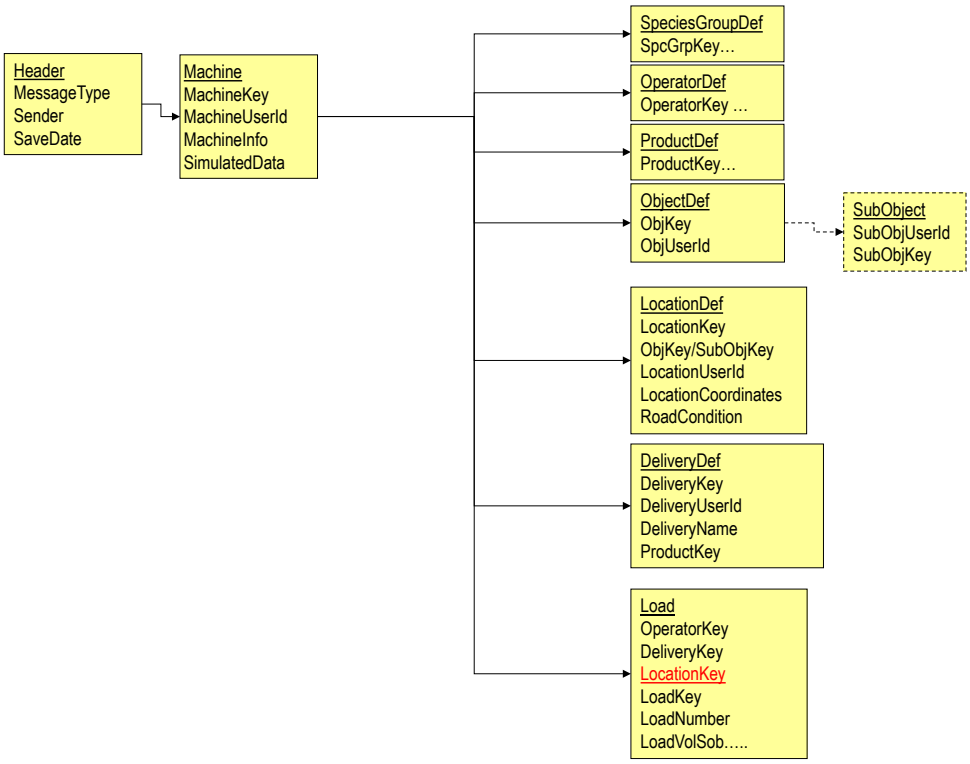


Figure 22. Diagram describing structures of forwarding production data.

Object specific information is included in Object- and LocationDefinition while information in SpeciesGroup-, Product- and DeliveryDefinition are in many situations constant for a long period of time covering several objects. The instruction sent to a forwarder is therefore divided into two separate messages like oin, spi and pin in the case of a harvester.

Below is one diagram illustrating the object specific information in the new ForwardingObjectInstruction message (figure 23) and one diagram illustrating the information used independently of object in the new ForwardingDeliveryInstruction message (figure 24).

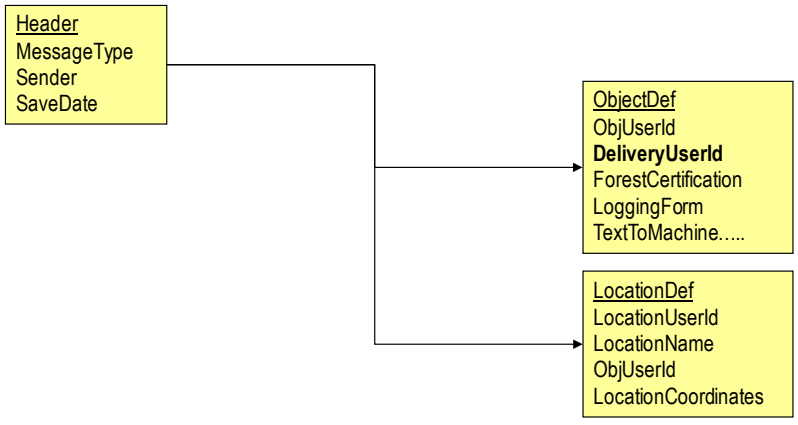


Figure 23. Overview of the ForwardingObjectInstruction.

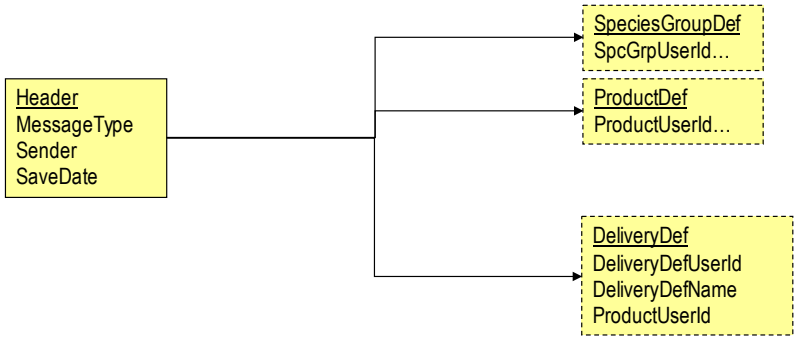


Figure 24.Overview of the ForwardingDeliveryInstruction.

The Forwarding DeliveryInstruction shall be handled in the same way as the pin-file. This means that DeliveryUserID in the foi-file defines what *Deliveries* to be used at a specific

A factor that makes the issue of forwarding instruction quite complex is the fact that the required information may come from several different sources as described in the figure below. An instruction might be sent from the forest company but it may also be manually included by the operator. A significant part of the relevant identities can also be found in an hpr-file from a harvester.

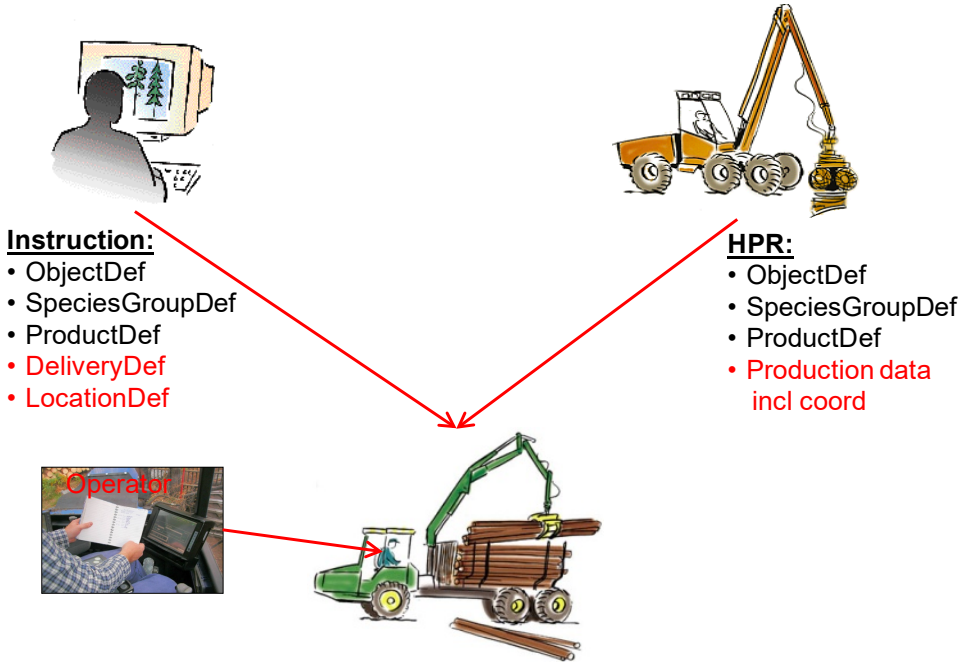


Figure 25.Example illustrating different data sources that may be used when starting forwarding on a new harvesting object.

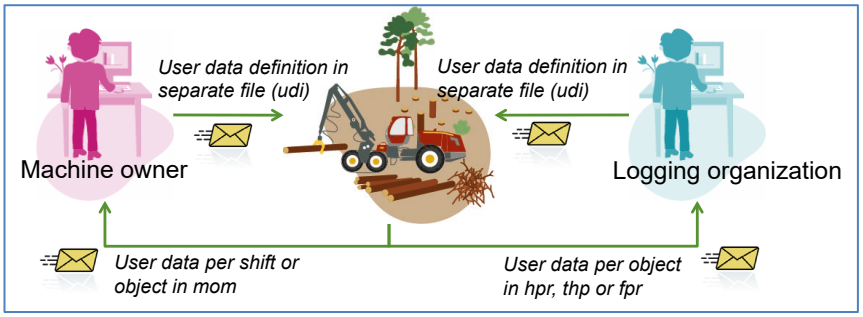
It is suggested that either a complete office instruction or an hpr-file is used when creating a new object in the forwarder. A single oin-file is probably of limited use since it does not contain a complete ProductDefinition or SpeciesGroupDefinition. The operator must manually define Delivery- and LocationDefinitions if an hpr-file is used since this information is not avalaible in the hpr-file. The operator must always complete missing parts in case of using an hpr-file.

The hpr-file must be sent to the forwarder if harvested production data is to be available in the forwarder. This means that harvester data can be used only for informing forwarder about harvested volumes if a forwarding instruction is sent from office to forwarder.

USER DEFINED DATA

StanForD 2010 includes a flexible solution for sending company-specific forms for e.g. follow-up in digital format (from version 2.1). The instruction defines user-specific tables and questionnaires to be manually filled in by the operator. The manually registered data are returned from the machine as part of messages for either production reporting or operational monitoring. User-specific data for follow-up could be cleaning of understory, oil consumption, number of high stumps for nature conservation, information regarding landings etc. This means that we hopefully can get rid of the operator writing additional company specific information on a piece of paper or in an excel-sheet.

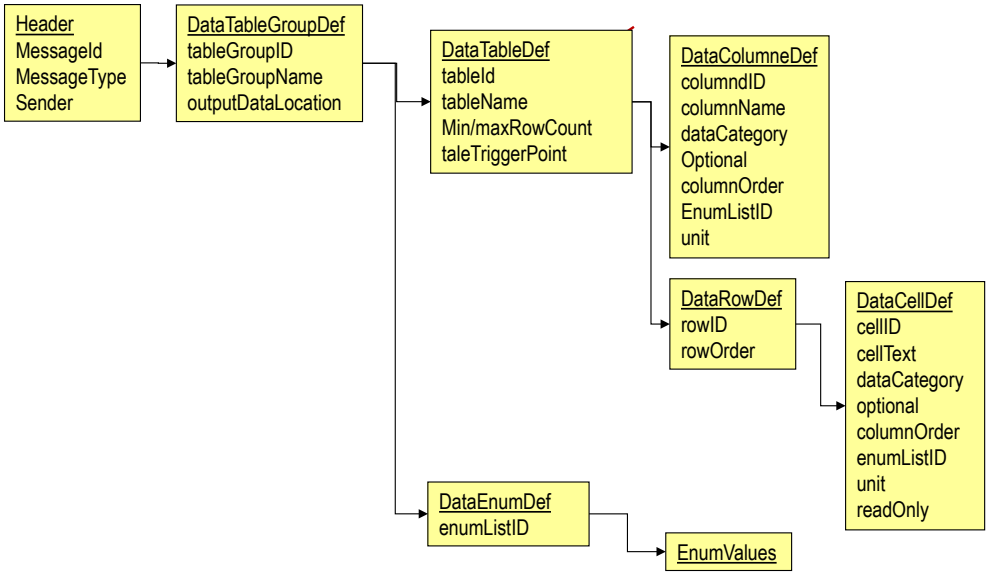
The figure below illustrates the solution for how instructions are sent to a machine from either forest machine owner or logging organization and then returned within a StanForD message.



The basic idea is to make it possible to define tables that the operator can use in the machine to manually register data relevant for logging organization or machine owner.

Input / Instruction

Below is an illustration describing the structure of the User defined data instruction (Udi) looks like:



Observe that the DataRowDefinition was added to make it possible to pre-define a number of questions with different enumeration lists for different rows.

DataTableGroupDefinition.

This structure is comparable with a database or an "excel work book" to be reported in a single way.

Attributes for DataTableGroupDefinition:

- tableGroupId (mandatory, string)
- tableGroupName (mandatory, string) = Name of user defined data table, to be used in the UI
- outputDataLocation (mandatory, enumeration list)

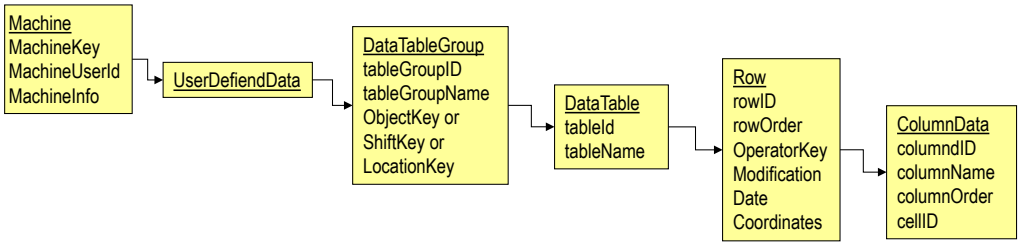
DataTableGroupDefinition is repeatable. The machine owner might need information both per shift and per harvesting object. This requires that there are possibilities to add many DataTableGroupDefinitions. The idea is that for every DataTableGroupDefinition there will be a new "tab" in the dialog.

Attribute ***outputDataLocation*** points to where the user defined data should be registered and then reported. The following enumerations are included:

- *ProductionObject*, information is stored per object in hpr, thp and fpr files (ObjectKey in illustration below is used).
- *MonitoringObject,* information is stored per object in mom files (ObjectKey in illustration below is used).
- *MonitoringShift*, information is stored per shift in mom files (ShiftKey in illustration below is used)). This means that the element ShiftKey must be used.

- *ProductionLocation*, information is stored per location in fpr files (LocationKey in illustration below is used). A recommendation could be to assume that the LocationKey references the last location where any volumes have been unloaded if several different locations are used at the same object.

The output data is stored using the following structure in hpr, thp, fpr and mom.



Tables with ProductionObject and MonitoringObject are object specific tables that are to be reset with a new object. If outputLocation is equal to MonitoringShift the table is shift specific, meaning it is to be reset when starting on new shift.

DataTableDefinition.

This structure is comparable with a database table or an "excel work sheet".

Attributes for DataTableDefinition:

- tableId (mandatory, string)
- tableName (mandatory, string) = Name of user defined data table, to be used in the UI
- maxRowCount (mandatory, "1" or "maxint") = Maximum number of rows in user defined data table.
- minRowCount (mandatory, "0" or "1") = Minimum number of rows in user defined data table.
- tableTriggerPoint introduced in version3.0 (mandatory, enumerations are "StartObject", "EndObject", "EndShift", "UnloadingComplete", "ChangeSubObject", "StartShift", "ChangeObject"). Attribute indicating when a user defined table must be presented to the operator (in the GUI of the machine) in order to make it simple to fill in the table at the right time.

DataColumnDefinition

This structure is comparable with fields in a database table or columns in an "excel-sheet".

Attributes for DataColumnDefinition:

- columnId (mandatory, string) = Identity of column

- columnName (mandatory, string) = Name of column. Presented in column header.
- dataCategory (mandatory, enumerations =”enum”, “string”, “integer”, “decimal”, “date”, “boolean” or “sequentialInteger”): Defines data type of column.
- optional (mandatory, boolean) = Indicates if column must be filled in by operator.
- columnOrder (mandatory, integer)= Presentation order of column in GUI.
- enumListId (optional, string)= Identity of enumeration list. E.g. "UnitList”, refers to a DataEnumDefinition (see also next section)
- unit (optional, string) = Defines unit of column, eg mm, kg, m3, cm etc. Can be used in GUI to inform operator about unit.

DataRowDefinition

This new suggested structure is used in order to define individual cells within a row if there e.g. is a need for defining different enumerations on different rows. All included setting must override settings in DataColumnDefinition.

Attributes for DataRowDefinition:

- rowId (mandatory, string) = Identity of row
- rowOrder (mandatory, integer) = Presentation order of row in GUI.

Attributes for DataCellDefinition:

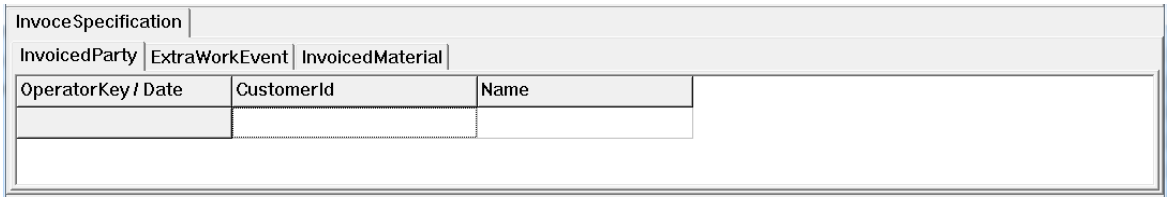
- cellId (mandatory, string) = Identity of cell
- cellText (optional, string) = Text in cell. E.g. a question to be answered by operator.
- dataCategory (optional, enumerations =”enum”, “string”, “integer”, “decimal”, “date”, “boolean” or “sequentialInteger”) = Defines data type of current cell, overrides DataColumnDefinition. Allowed values are: enum, integer, string, decimal, date, boolean, sequential no per row.
- optional (optiona, booleanl): Indicates if cell must be filled in by operator.
- columnOrder (mandatory, integer) = Column order for current cell in GUI.
- enumListId (optional, string) = Identity of enumeration list. E.g. "UnitList”, refers to a DataEnumDefinition (see also next section)
- unit (optional, string) = Defines unit of column, eg mm, kg, m3, cm etc. Can be used in GUI to inform operator about unit.
- readOnly (optional, Boolean) = Indicates if cell can be edited by operator.

DataEnumDefinition

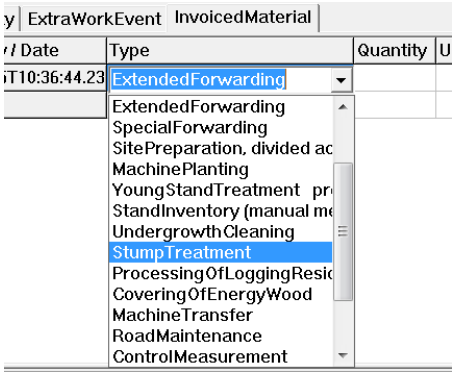
This structure includes enumeration lists to be used in the machine. Attribute enumListId identifies the enumeration list to be used.

User interface example of table

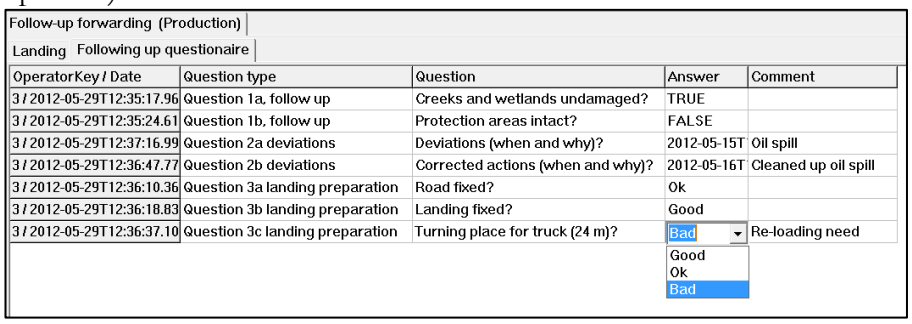
The instruction could render the following GUI.



Enumeration list for column:



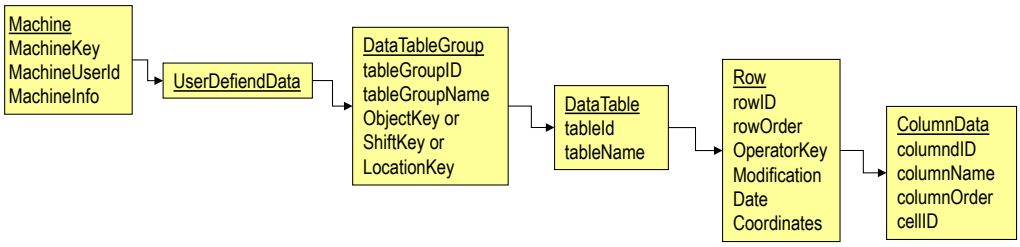
An instruction could also render the following GUI (example is filled in by operator):



Output / Report

The data is validated according to the definition before the file is created. This means that all mandatory fields have to be filled in.

Below is the structure for reporting user defined data from the machine, always included under Machine in mom, fpr, hpr or thp:



User interface example

The output data (XML) described above could be illustrated through the following GUI example.

InvoceSpecification

InvoicedParty | ExtraWorkEvent | InvoicedMaterial

OperatorKey / Date	CustomerId	Name
3 / 2001-12-17T09:30:47Z	35643	BigForest Lmtd

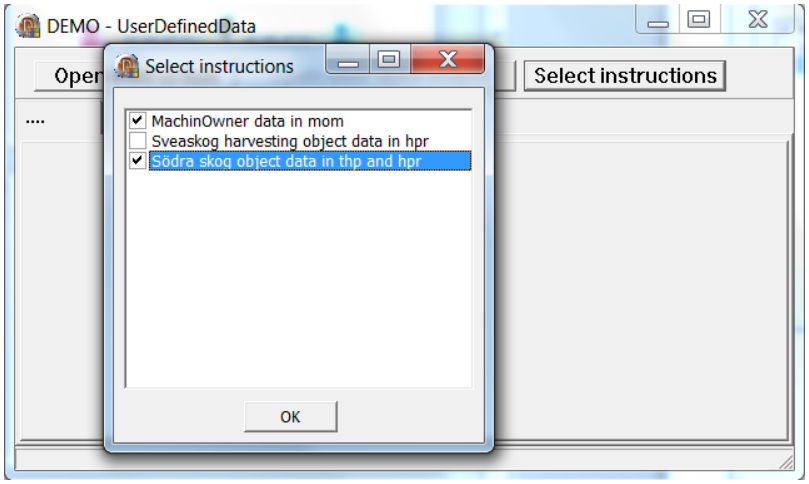
InvoceSpecification

InvoicedParty | ExtraWorkEvent | InvoicedMaterial

OperatorKey / Date	ExtraWorkCategory	BasicsForInvoicingTheWork	Quantity	Unit	Comment
3 / 2001-12-17T09:30:47Z	YoungStandTreatment	Ordered by forest owner	26	hours	This is a test of invoice extrawork
3 / 2001-12-17T09:30:47Z	StandInventory	Ordered by logging company	4	hours	
3 / 2001-12-17T09:30:47Z	StandInventory	Ordered by logging company	4.5	ha	

Managing different instructions in machines

A machine may very well receive instructions from several different organizations. I simple solution for handling different instructions could be that the operator manually selects what instructions to use when starting at a harvesting object:



Operator manually deletes old instructions that are not to be used anymore.

Messages sent from forest machine

Below follows a number of messages covering production data, quality control data and operational monitoring data sent from forest machines.

There are no rules concerning partial reporting (data reported only once). However a recommendation has been written by Skogforsk concerning reporting of hpr-messages. This recommendation is included in appendix 1.

It must be possible to generate complete production messages including all harvested and forwarded production from one specific harvesting object (fpr and hpr).

The only file that always must include all data from an object, in other words must be an aggregated file, is the thp message.

All messages are normally object oriented with the exception of the mom message which always must be time oriented. This means that all times within the report interval must be included independently of harvesting object.

It is strongly recommended that all manufacturers implement a static folder (location) where files are saved by default, it is acceptable to have separate folders for different file types. It is for example important that hpr messages are not saved in different folders depending on harvesting object or harvesting date.

HARVESTED PRODUCTION

The message structure for *harvested production* covers the present stm- and pri-files.

A new data structure for production reporting must make it possible to:

- Identify multiple occurrences of identical stems (using combination of *MachineKey* and *StmKey*)
- Identify missing stems (*StemNumber* a running stem number per object needed)
- Identify location (object & sub-object) of a stem
- Identify objects for a specific machine if data from several machines are included in one message.
  Example:
  Data from harvesters A and B harvesting object XX is merged into one message. It must be possible to separate Object-data (for example start and end date) for the two harvesters.

It must be possible to generate an hpr-file including all data from HarvestingStartDate of a harvesting object until the time the message was created (HarvestingSaveDate). This means in other words that it must be possible to generate a message including accumulated data.

A structure similar to the present pri-file will be used (figure 27). Using this structure will make it possible to register data from several machines and harvesting objects in one single message. The abbreviation GUID means Globally Unique Identity. Observe that the combination of *MachineKey*, *StmKey* and *LogKey* must be totally unique for each and every log.

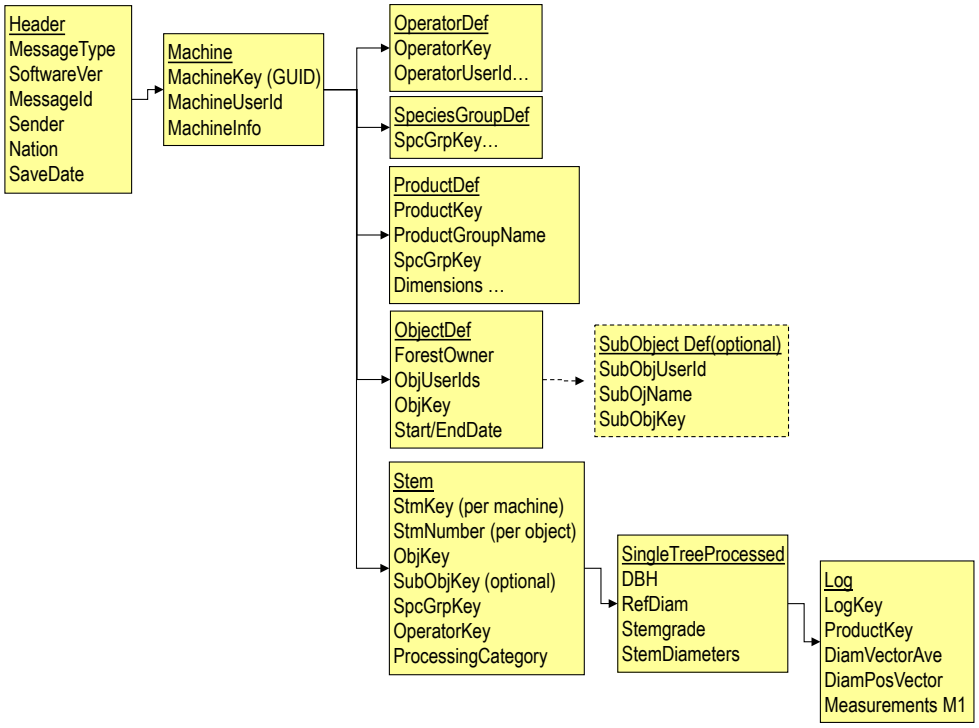


Figure 27. General layout of message including harvester production data. Diagram also describes how sub-objects may be used. Observe that MultiTreeProcessing is not included.

A more detailed description of harvested production data is included in figure 28. Optional structures for diameter and grade vectors, today found in the stm-file, are included. Observe that this kind of very detailed information should only be included if there is a clearly expressed need for this information, for example when doing analyses of bucking optimization results.

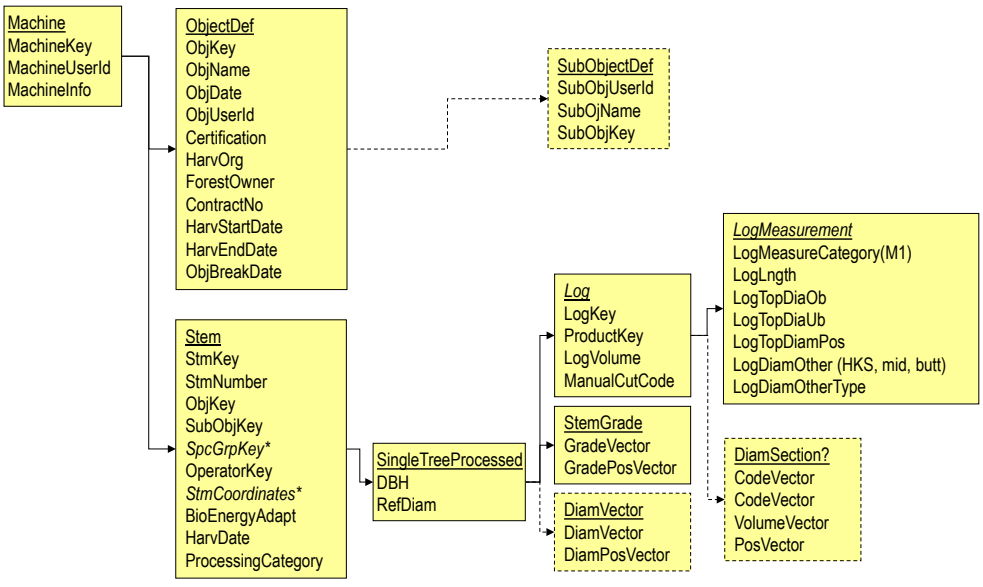


Figure 28. Diagram describing detailed harvesting data.

We have not included the possibility to register individual measurement directions for diameters (cross measurement) in diagram 28. Only absolute filtered diameter values are to be registered.

Observe that HarvestingDate for each stem is optional. It should always be possible to set in the machine whether or not you want to include the HarvestingDate in hpr. This is sensitive information to send to a forest company that does not own the harvester. However this may be very useful information not only when machine owner wants to do detailed productivity studies but also if information is sent from harvester to forwarder. It would make it very easy for the operator of a forwarder to see what the harvester has cut during for example the last 12 hours. What assortments have been cut and where is it located.

MultiTreeHandling

The hpr structure is possible to use for both single tree harvested and multi tree harvested (MTH) stems as illustrated in figure below.

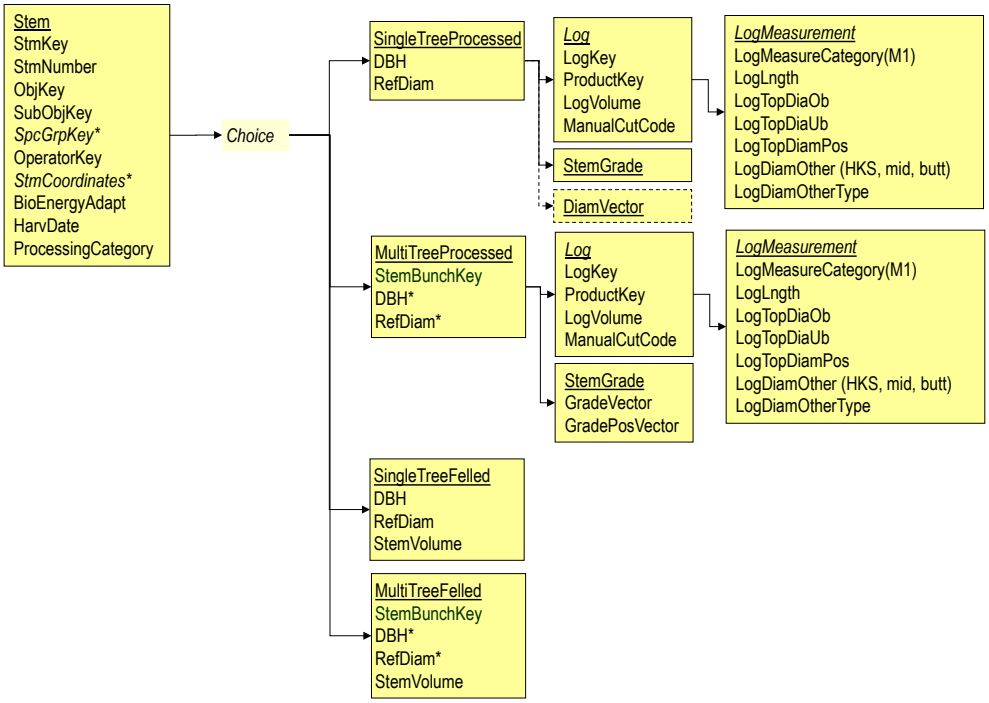


Figure 29. Ilustration of the different processing categories in the hpr-file-

The advantage with having the MTH trees in the same structure is that we get one structure for all stems. Each individual stem harvested together (stems in a stem bunch) is registered separately under element Stem. The total number of Stem elements is to be equal to the total number of harvested stems.

The following data is included for each stem in order to use the same structure for single and multi tree harvested stems:

- Mandatory element *StemBunchKey* which is a running number used for all MTH stems. All stems in a stem bunch are given the same *StemBunchKey*, this means that it is possible to analyse a bunch of stems.
- Indication of whether logs have been measured (volume) or if one or more logs have been estimated using new log volume types: *m3subEstimated* and *m3sobEstimated*
- DBH is normally an extrapolated diameter since the harvester head normally measures the first diameter some dm below 120 cm (height from the stump). The element *ReferenceDiameter* has therefore been included in order to register the actually measured diameter of MTH stems.
- Element ProcessingCategory indicates what type of harvesting was done. This element is especially useful in case of SingleTreeFelling with no measurement data available to be registered under structure *SingleTreeFelledStems*.

Observe that all MTH data below (excluding LogVolume) normally will represent the first stem if not measured for each individual stem: HarvestDate, DBH, Coordinate, StemGrade, StemDiameter, LogDiameter and LogLength.

Do not mix single tree processing, multi tree processing, felling (multi and single tree felling) when calculating means stem volumes.

Rules

The same DBH, ReferenceDiameter and species are registered for all stems in a stem bunch if only DBH and Species is measured/registered for the first stem. The estimated log volumes shall also be identical and represent an individual log and NOT the log bunch. Element must *StemBunchKey* included. Only LogVolume enumerations m3sobEstimated and m3subEstimated are to be used for MultiTreeHarvested stems.

Example

Only possible to measure and register the DBH and species of the first stem. Example of data today illustrated in table 7 below.

Table 7.Three stembunches where DBH and species is only measured on the first stem

Bunch	Species, 1st stem	DBH, 1st stem	No of stems
1	Pine	121	3
2	Spruce	106	2
3	Pine	131	2

The table below illustrates how the MTH stems in table 7 will be registered in the hpr.

Table 8.

StemNo-PerObj	StemBunch-Key	Species-Group	DBH	referenceDiameter	NoOfStems
1	1	Pine	121	129	3
2	1	Pine	121	129	3
3	1	Pine	121	129	3
4	2	Spruce	106	109	2
5	2	Spruce	106	109	2
6	3	Pine	131	138	2
7	3	Pine	131	138	2

Unclassified logs

Unclassified logs are registered in the same structure as normal logs. However the ProductKey for these logs must refer to a hard coded ProductDefinition of the type MeasuredUnclassifiedProduct. This ProductDefinition is never to be sent to machine from logging organization.

The following table illustrates the differences between the ProductDefinitions for classified vs unclassified logs.

Table 9.

MeasuredClassified-ProductDefinition	MeasuredUnclassified-ProductDefinition
ProductKey	ProductKey
ProductName	ProductName
SpeciesGroupUserId	ModificationDate
ModificationDate	ProductInfo
ProductInfo	ProductIdInfo
ProductIdInfo	LoggingOrganization
ProductBuyer	ProductGroup
LoggingOrganization	
ProductGroup	
ProductUserId	
SpeciesGroupKey	
DiameterDefinition	
LengthDefinition	
PriceDefinition	

TOTAL HARVESTED PRODUCTION

Several manufacturers have seen a need for having the possibility to register total volumes per object, sub-object, product and operator. A structure as described in figure 20 has thus been included in a separate *total harvested production* message. This message will be of interest in cases when no detailed information about individual logs is needed and when communication capacity is low. This is a "Pandora's box", very strict rules about not adding new data to this structure must be agreed on.

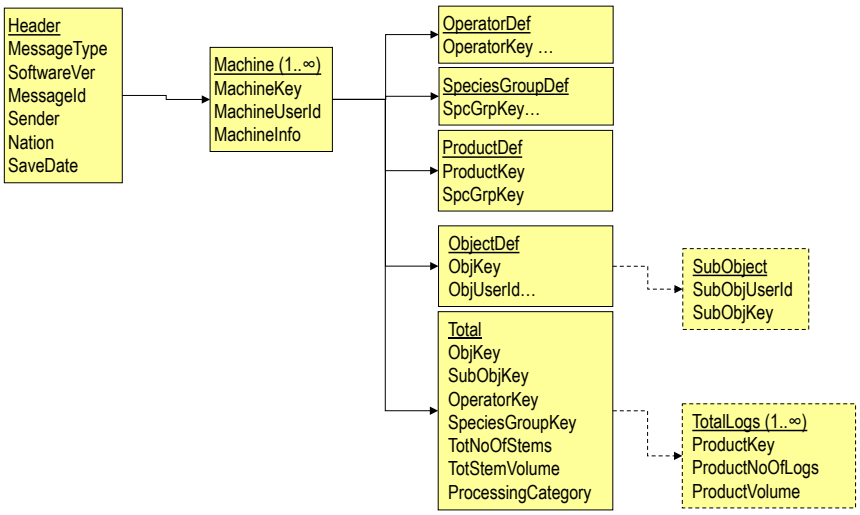


Figure 30. Diagram describing suggested variables for making it possible to register total production (prd-light).

The total harvested production message is always to include all data from HarvestingStartDate of a harvesting object until the time the message was created (SaveDate). This means in other words that the message only includes accumulated data.

HARVESTING QUALITY CONTROL

The message structure for quality control covers the old ktr- and stm-files.

The *harvesting quality control* message is defined as a separate message but has the same structures as the *harvested production* message. Only stems used specifically for control or calibration are to be included in this message.

The quality control message is described in figure 21 below. The possibility to also include other data than measurement quality control data has been discussed but it will be kept in a separate message.

All data communication connected to measuring quality control will be using the hqc message. This includes both sending data to and from the caliper. Element *DiameterVector* must be included with diameters per each dm module when sending an hqc to the caliper. The hqc sent to caliper must include historical log regarding rejected stems and calibration. This means that StanForD 2010 harvesters must ”offer” a complete hqc (including M1 data and calibration/reject log) to the control and calibration system (for example a caliper together with a com driver).

The old Kermit protocol will excluded from StanForD 2010. This means that no increase in Kermit transmission speed is needed. It is in other word up to the machine and caliper manufacturers to decide how to set up the communication which could be based on some kind of com-driver. A com driver is a communication software located either in sending or receiving part of the system (harvester / caliper / measuring sensor). When using com driver it will be machine manufacturer’s and caliper manufacturer’s joint decision which communication protocol and what type of files are to be used.

It was decided, in order to avoid problems, that the hqc files sent from control and calibration system is not to be modified in any way by the harvester.

Whether standardized folders for in- and output in external usb is needed is in other words an issue between harvester and caliper manufacturers.

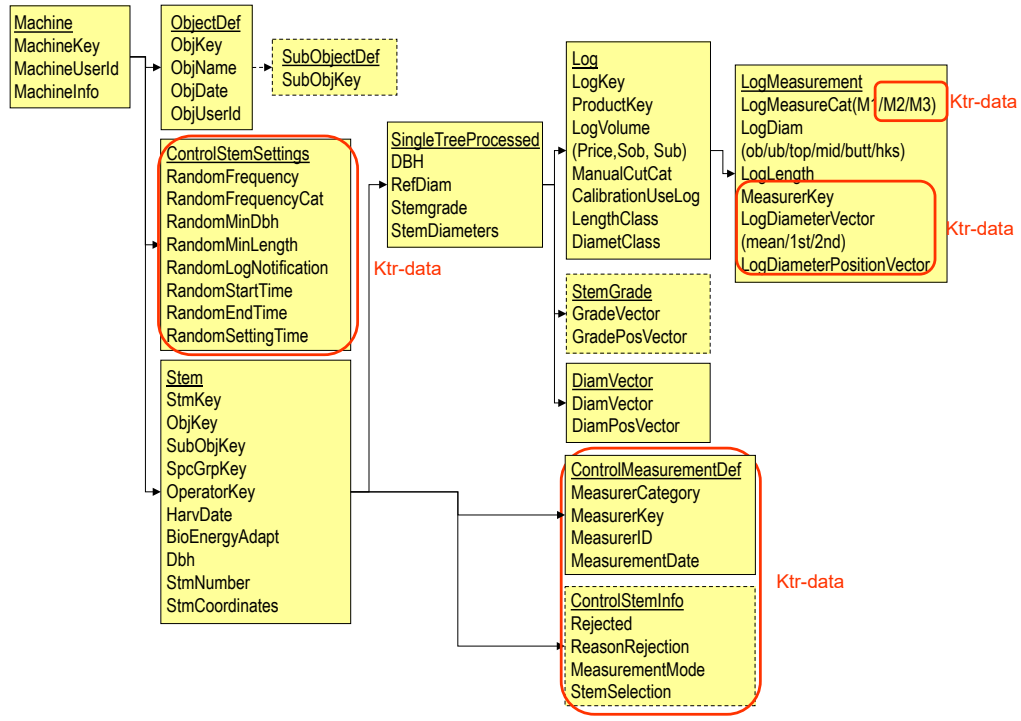


Figure 31. General layout of quality control message. "Ktr-data" indicates the data normally measured manually and registered in a computer caliper. MTH stems are presently not included in hqc messages.

Data regarding calibration history (figure 22) will also be included in the *harvesting quality control* message.

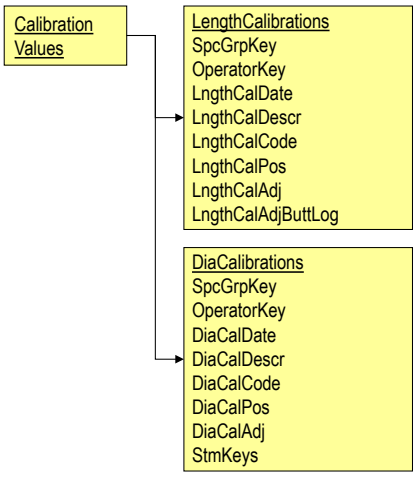


Figure 32. Diagram describing calibration log.

FORWARDED PRODUCTION

The structure for forwarder production covers the old prl-file.

New structures *LocationDefinition* and *DeliveryDefinition* has been added to the *forwarded production* message when comparing with the old prl-file (figure 25). The old TranspObject is basically divided up into these two new structures.

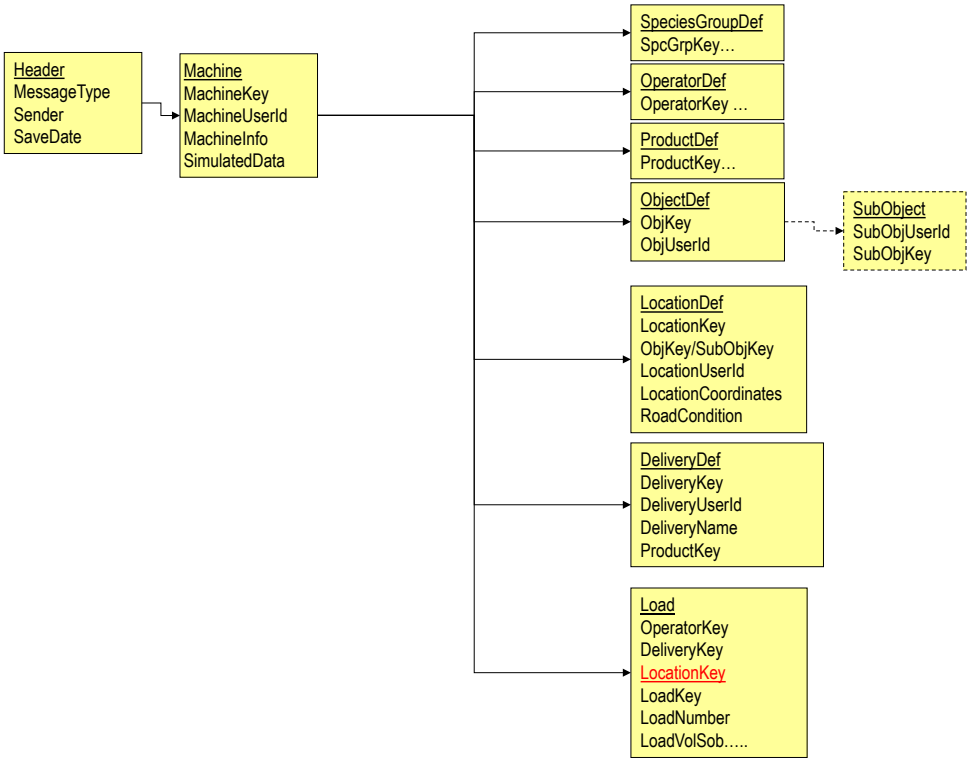


Figure 33. General layout of forwarder production message.

The following structures are specific for the fpr-message:

- *Load* and *PartialLoad* includes forwarded production (no of units, volume, mass), operator identity and unloading time. *Delivery* and *Location* shall be registered for each unloading in element structure PartialLoad.
- *DeliveryDefinition* includes a description (content) of forwarded production, references to harvested products and time stamps (start/end). It may also include a reference to the final destination (industry)
- *LocationDefinition* includes a specification of where the forwarded production has been unloaded in the form of coordinates and harvesting object. A description of the road condition is also possible

This means that if the transportation from forest to industry is to be planned relevant volumes can be found in the *LoadDefinition* structure, the content and delivery destination in the *DeliveryDefinition* (in combination with *ProductDefinition*) and the position of the available volumes can be found in *LocationDefinition*.

It is possible to identify the following for each unloading using the structure illustrated above:

- A delivery specification (content)
- A location
- A harvesting object
- An operator
- One or several merged products (assortments)

Observe that a *DeliveryDefinition* can be connected to more than one location and that several locations can exist at one harvesting object.

The fpr structure means that we clearly separate description and location. As a consequence of this a new structure *ForwardingStatus* is included with elements for registering when a certain combination of delivery and location was first used (element *ForwardStartDate*) and when it was completed (element *ForwardEndDate*). This modification also means a start and end date per location and delivery is irrelevant.

The diagram below gives a detailed description of the structures specific for the fpr-message.

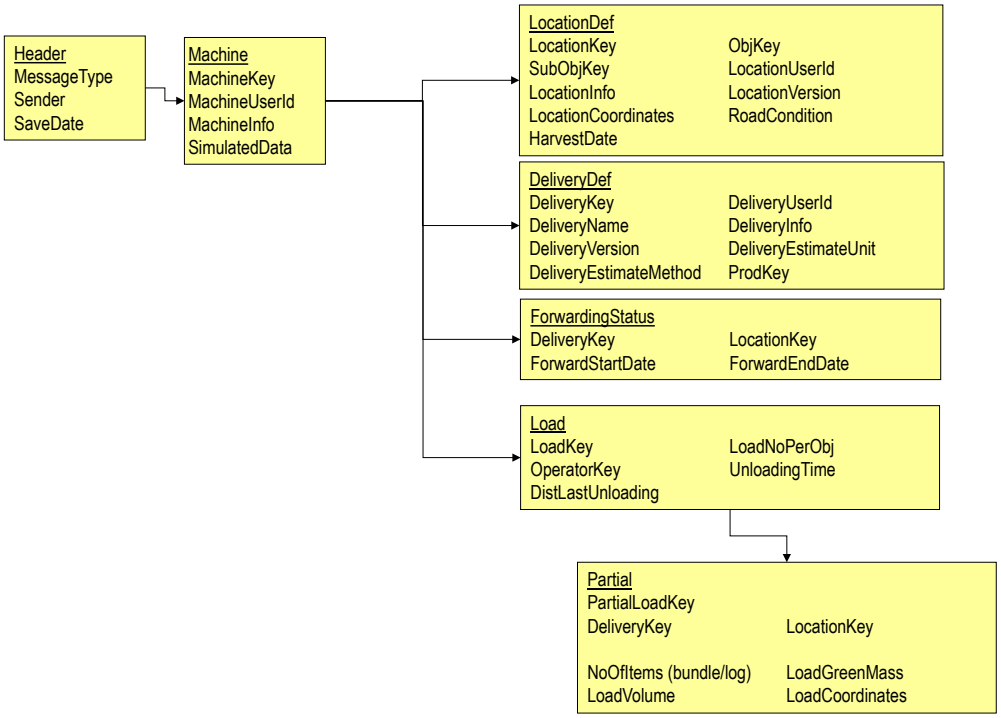


Figure 34. Detailed view of fpr-structure including all production data

The following keys are critical for identifying a specific load:

- One *DeliveryKey* registered for each unloading (load dataset)
- One or several *ProductKeys* registered per *DeliveryDefinition*
- One *LocationKey* registered for each unloading (load dataset)
- One *ObjectKey* registered per *LocationDefinition*

Below is a diagram illustrating these primary references between load data, delivery, product, location and harvesting object.

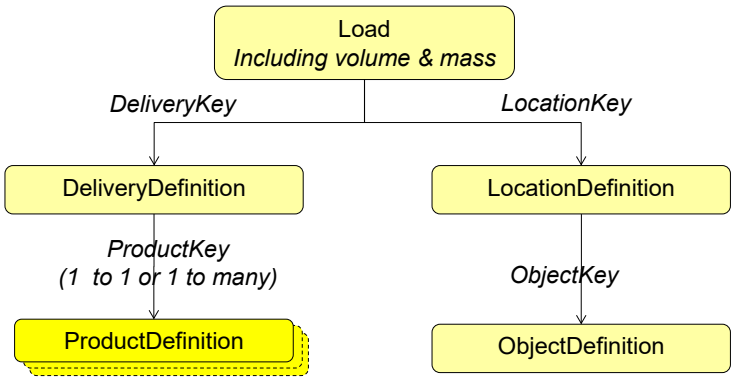


Figure 35. References between load data, delivery, product, location and harvesting object in fpr-file.

The following figure illustrates the fact that the total driven distance between point 1 and 4 (red arrows) are to be included in DistanceFromLastUnloading when unloading the pulpwood and small sized saw logs.

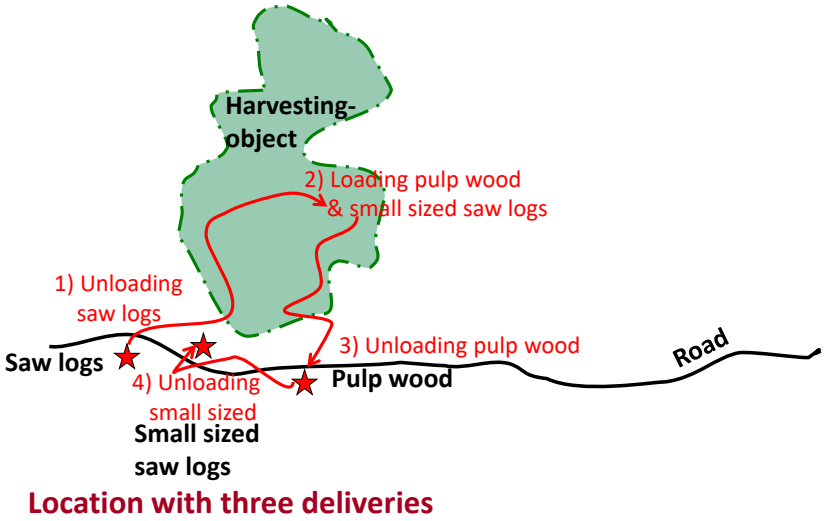


Figure 36 Illustration of DistanceFromLastUnloading.

FPR EXAMPLES

Two different examples are included in this document in order to illustrate how the fpr-message might be used when product specific volumes are to be registered as well as when several products are merged into a single *TransportObject*.

Production per product (assortment)

This is an example of how to use the suggested structure when the objective is to register the forwarded volumes per harvested product and harvesting object.

A number of harvested products at an object are described in table 1.

Table 1.Harvested products in forwarding example

ProdKey	ProdInfo	ProdName
1	0110	PineSaw
2	1010	PinePulp
3	5090	PineEnergy
4	0120	SpruceSaw
5	1020	SprucePulp
6	5090	SpruceEnergy

Only one single location defined since no specification per landing is assumed to be needed (table 2).

Table 2. A single location at a harvesting object

LocationKey	LocationUserId	LocationName	Coord.	ObjKey
1	101	XY	60.01,16	1

One delivery per product is defined which means that we have a one-to-one relationship between *ProductDefinition* and *DeliveryDefinition*:as illustrated in table 3.

Table 3. A Delivery per Product at a harvesting object.

Delivery-Key	ProdKey	Delivery-UserId	DeliveryName
11	1	0110	PineSaw
12	2	1010	PinePulp
13	3	5010	PineEnergy
14	4	0120	SpuceSaw
15	5	1020	SprucePulp
16	6	5020	SpruceEnergy

The deliveries above could be considered to be copies of the product definitions with some additional information like start and end dates.

One dataset per unloading (unique combination of load, DeliveryDefinition and Location) is registered in *Load* and *PartialLoad* elements as illustrated in table 4.

Table 4.Forwarded amounts are registered per Location and Delivery in PartialLoad element

LoadKey	LoadNo	Partial-LaodKey	Delivery-Key	Location-Key	Operator	GreenMass	UnloadingTime
73	1	1	11	1	1	7 900 kg	10-02-14, 12:10
73	1	2	12	1	1	2 850 kg	10-02-14, 12:10
74	2	1	13	1	1	10 350 kg	10-02-14, 12:45
75	3	1	16	1	1	6 250 kg	10-02-14, 13:11
75	3	2	13	1	1	4 500 kg	10-02-14, 13:11
76	4	1	14	1	1	9 700 kg	10-02-14, 13:51
77	5	1	15	1	1	10 920 kg	10-02-14, 14:27

Data concerning status of forwarding to specific combination of location and delivery is registered in *ForwardStatus* as illustrated in table 5.

Table 5.Example of ForwardingStatus data

DeliveryKey	LocationKey	ForwardStartDate	ForwardEndDate
11	1	10-02-14, 12:10	-
12	1	10-02-14, 12:18	-
13	1	10-02-14, 12:45	10-02-14, 13:21
14	1	10-02-14, 13:51	-
15	1	10-02-14, 14:27	-
16	1	10-02-14, 13:11	-

Production per industry and location

The *DeliveryDefinition* and *ProductDefinition* are basically identical in the previous example. So why include the *DeliveryDefinition* in the fpr-message?

The harvested products are often merged together by the forwarder and unloaded in the same piles since these products are often delivered to one single industry.

A very common case in Sweden is that pulp logs from both spruce and pine are of mixed and transported to one industry, the same thing with bio energy assortments. Another example might be if you transport all assortments to a terminal where they are sorted and measured and the only relevant volume when transporting is the total volume.

The following products were harvested at the harvesting object:

A number of harvested products at an object are described in table 6.

Table 6. Harvested products in forwarding example

ProdKey	ProdInfo	ProdName
1	0110	PineSaw
2	1000	PinePulp
3	5090	PineEnergy
4	0120	SpruceSaw
5	1000	SprucePulp
6	5090	SpruceEnergy

Two separate locations are defined in this case since a specification per landing is assumed to be needed (table 7).

Table 7. Two locations at a harvesting object

LocationKey	LocationUserId	LocationName	Coord.	ObjKey
1	101	South	60.01,16	1
2	202	North	60.02,16	1

All pulp wood products are to be transported to one industry and the same for the energy assortments. *DeliveryDefinitions* 13 and 14 therefore includes two products which mean that we have a one-to-two relationship between *DeliveryDefinition* and *ProductDefinition* for *Delivery* Pulp and Energy as illustrated in table 8.

Table 8.Four deliveries at a harvesting object where there are references to several products in two cases

Delivery-Key	ProdKey	Delivery-UserId	Delivery-Name
11	1	0110	PineSaw
12	4	0120	SpuceSaw
13	2, 5	1000	Pulp
14	3, 6	5090	Energy

One dataset per unloading (unique combination of load and delivery definition) is registered in the *Load* and *PartialLoad* element (table 9).

Table 9. Forwarded amounts are registered per Location and Delivery in PartialLoad element

LoadKey	LoadNo	Partial-Loadkey	Delivery-Key	Location-Key	Operator	GreenMass	UnloadingTime
73	1	1	11	1	1	7 900 kg	10-02-14, 12:10
73	1	2	12	1	1	2 0850 kg	10-02-14, 12:10
74	2	1	14	1	1	10 350 kg	10-02-14, 12:45
75	3	1	13	1	1	6 250 kg	10-02-14, 13:11
75	3	2	14	1	1	4 500 kg	10-02-14, 13:11
76	4	1	12	1	1	9 700 kg	10-02-14, 13:51
77	5	1	13	2	1	10 920 kg	10-02-14, 14:27

LoadKey 2 in the table above, has a reference to delivery *Energy* which also has references to products *PineEnergy* and *SpruceEnergy* as well as a reference to *Location South*.

Data concerning status of forwarding to specific combination of location and delivery is registered in *ForwardStatus* as illustrated in table 5.

Table 10.Example of ForwardingStatus data

DeliveryKey	LocationKey	ForwardStartDate	ForwardEndTime
11	1	10-02-14, 12:10	-
12	1	10-02-14, 12:18	-
13	1	10-02-14, 13:11	-
14	1	10-02-14, 12:45	10-02-14, 13:21
13	2	10-02-14, 14:27	-

FORWARDING QUALITY CONTROL

It was suggested that in January 2010 that a new message for quality control of forwarders are to be included in StanForD 2010.

This message is primarily based on the old var61 that was implemented 2009-04-01. The focus is on control and calibration of the forwarder scale.

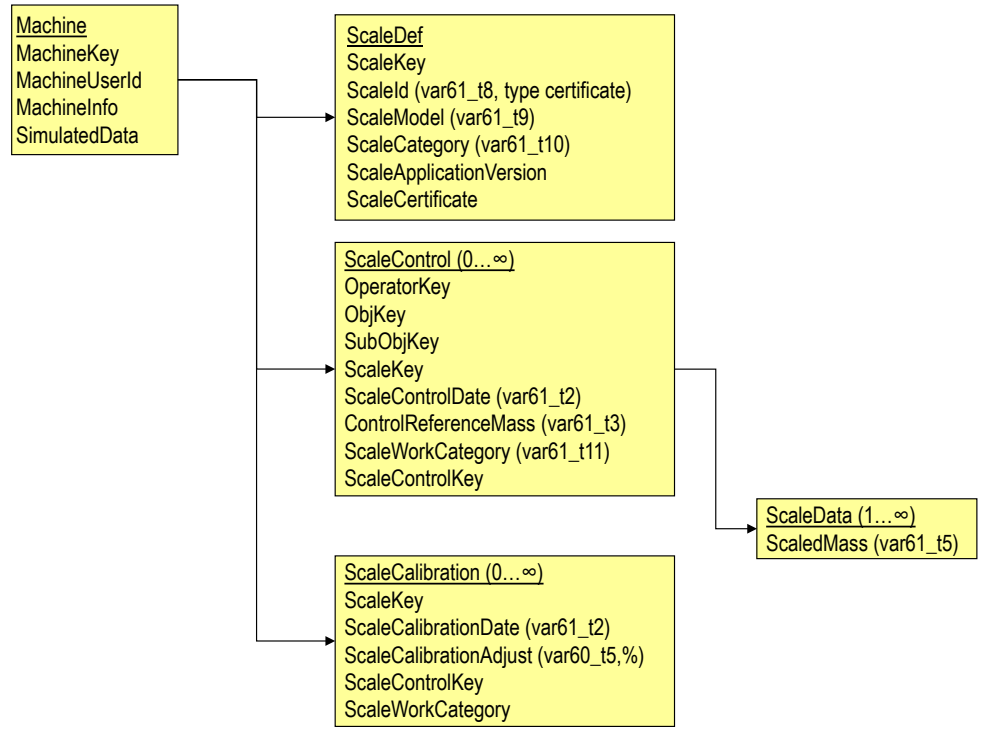


Figure 36. Diagram structure of forwarding quality control

Table 10. Elements specific for the forwarding quality control

Name	Description
ScaleControlDate	Date and time of weight scale control measurement:
ControlReferenceMass	Reference weight, weight of the control object e.g. test object or weight bridge. (kg)
ScaledMass	Registered mass of control scaling in forwarder per scaling occasion. Weight values of the forwarder's scale. (kg)
scaleWorkCategory	Forwarder's working type when weighting with weight scale is done:loading or unloading (enumeration list). Variable only used for weight scale control measurements.
scaleControlCategory	Category of scale control: "Random dynamic mass" or "Manual fixed mass"
Orientation	Oriantation of crane during scale control: "Right side of load space" or "Left side of load space"
ScaleCalibrationDate	Date and time of weight scale calibration
ScaleCalibrationAdjust	Scale adjustment, % (Old text: Factor used when weighing.)
scaleWorkCategory	Forwarder's working type when weighting with weight scale is done:loading or unloading (enumeration list). Variable only used for weight scale control measurements.
ScaleKey	Name and identity of certificate of type examination for scale (free text)
ScaleID	Identity of scale
ScaleModel	Scale model and manufacturer (free text)
ScaleCategory	Scale type: grapple scale or load bearer scale (enumeration list)
ScaleApplicationVersion	Version of scale software.
ScaleCertificate	Name and identity of certificate of type examination for scale

OBJECT GEOGRAPHICAL REPORT

The structure for *object geographical report* is to a large extent a copy of the present ghd-file.

The structure of the *geographical report i*s similar to the *geographical instruction*. The *object geographical report* message includes references to what GIS-files (layers) have been used in the machine. This message also includes information about how the different layers are to be presented (formatting) and what information can be found in complementary data files (for example dbf-files).

A number of variables are included for identifying what modifications and updates have been done as well as by whom.

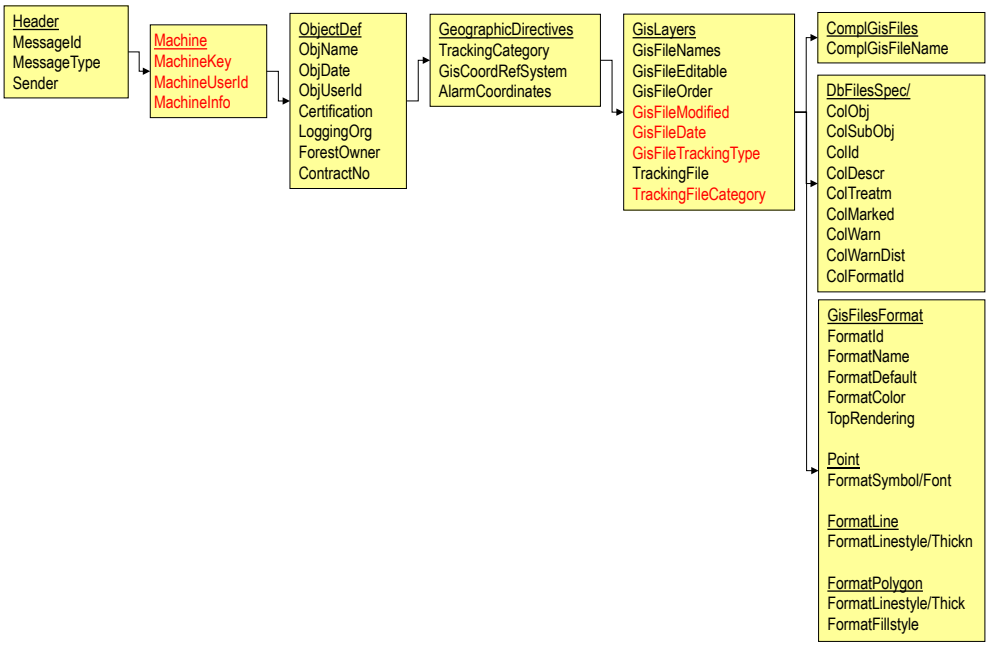


Figure 37. Diagram describing the GIS structures (basically copy of ghd-file). Observe that some of the variables in the diagram above are not included in the geographical instruction (red text).

OPERATIONAL MONITORING

The structure for operational monitoring covers the old drf- and rep-files.

The new monitoring message shall always be time oriented, meaning that we register all times within a certain time interval. The main reason for this is that if we have used the same machine at different objects, for example going from object A to B to A and then only register data for object A the machine will appear to be inactive while it has actually done work at object B. It is of course possible to extract only data covering a certain harvesting object if the start date and end date is equal to the chosen time interval.

Time is registered as different time types that can be displayed separately and compared for follow up purposes. The main types to be used are:

- machine work times
    - other machine data (production , fuel, distance).
- operator work times

These different types of time will be possible to draw as separate but comparable timelines (figure 25-27). It will also be possible to compare monitoring data with production data if harvesting and unloading times are registered in production messages.

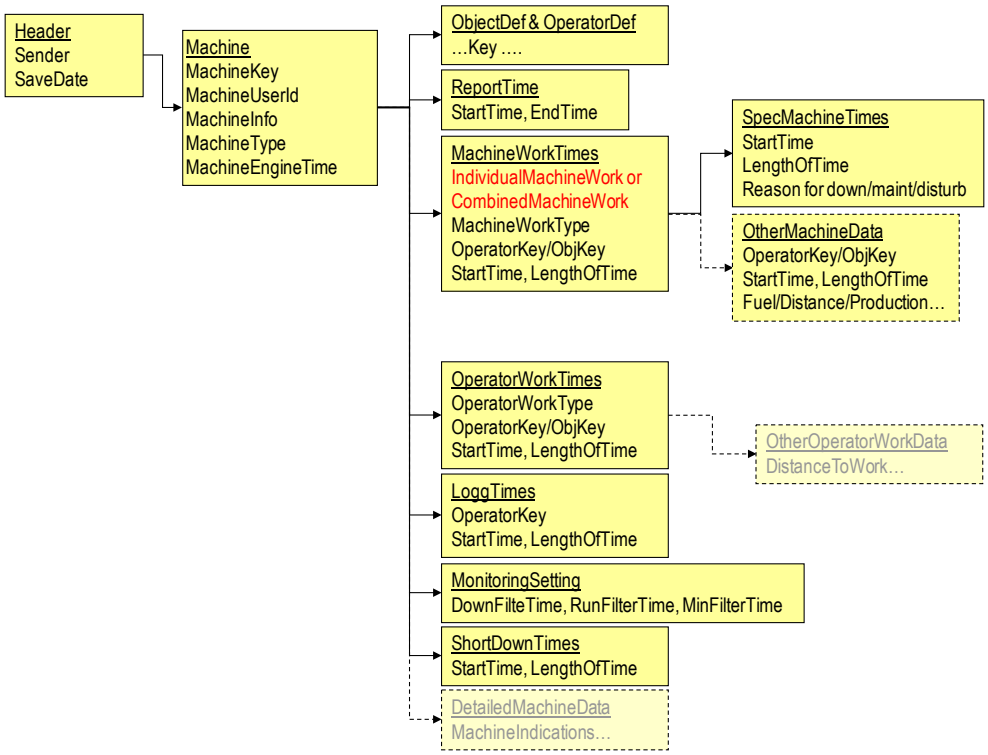


Figure 38. General layout of message including operational monitoring data.

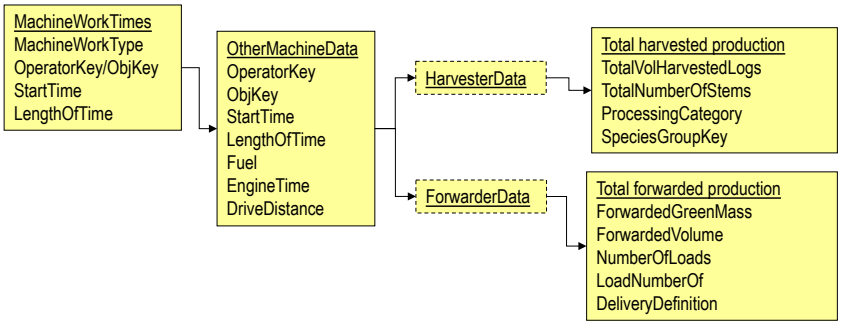


Figure 39. Diagram describing variables for other productivity related data in monitoring messages. ** Production data from harvesters and forwarders may instead be registered in production files.

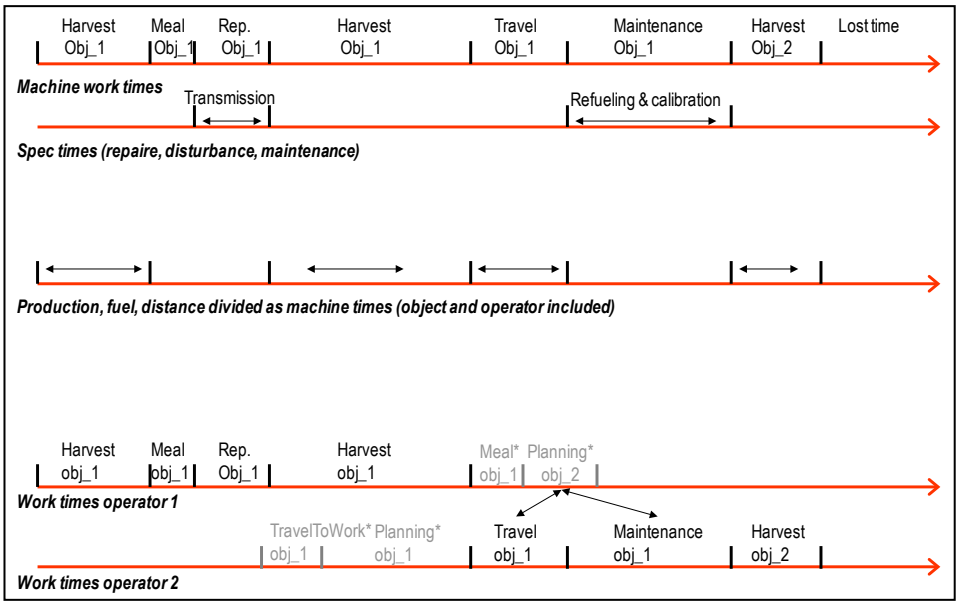


Figure 40. Example illustrating operational monitoring data as six separate but comparable timelines. * Operator not working on or with machine.

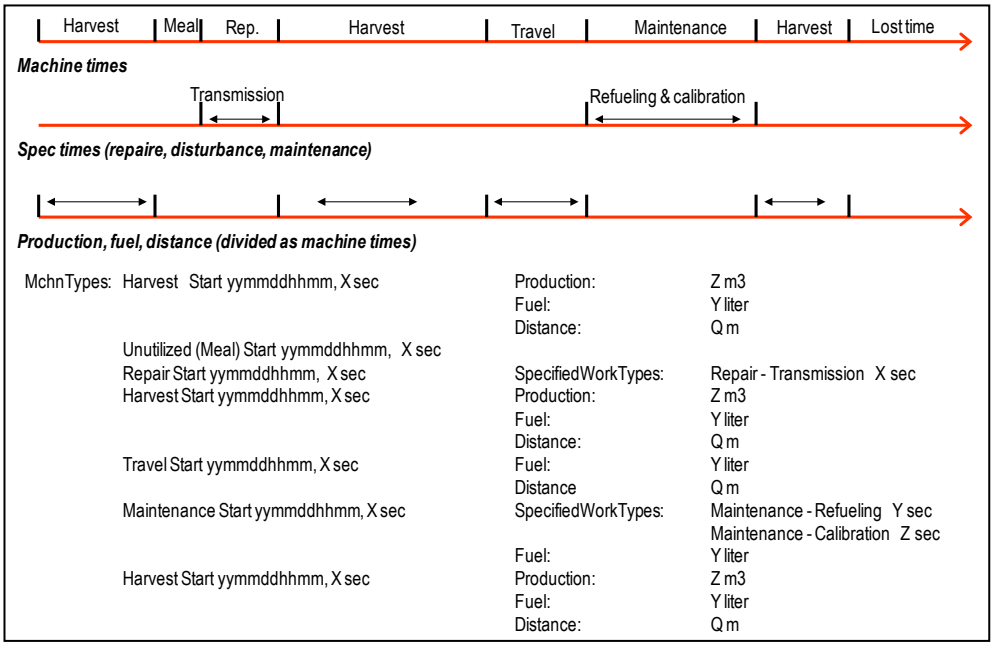


Figure 41. Example individual time registration in operational monitoring message (date and time format shortened)

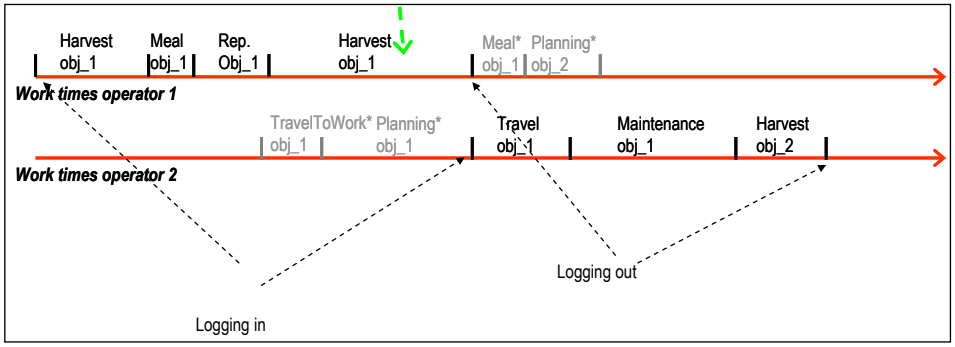


Figure 42. Example registration of logging in and out.

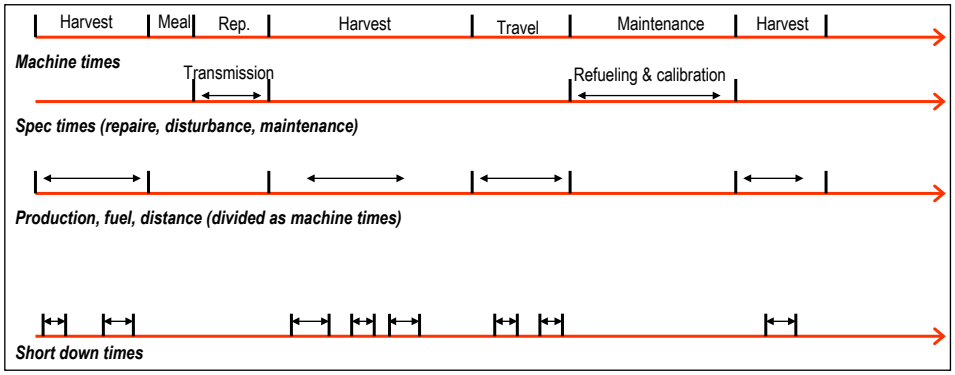


Figure 43. Example registration of short down times, observe that it will also be possible to register short down times in a frequency table as in the present standard.

It may also be of interest to define structures for registering:

- Basic machine indications (cutting, felling, feeding, machine movement etc)

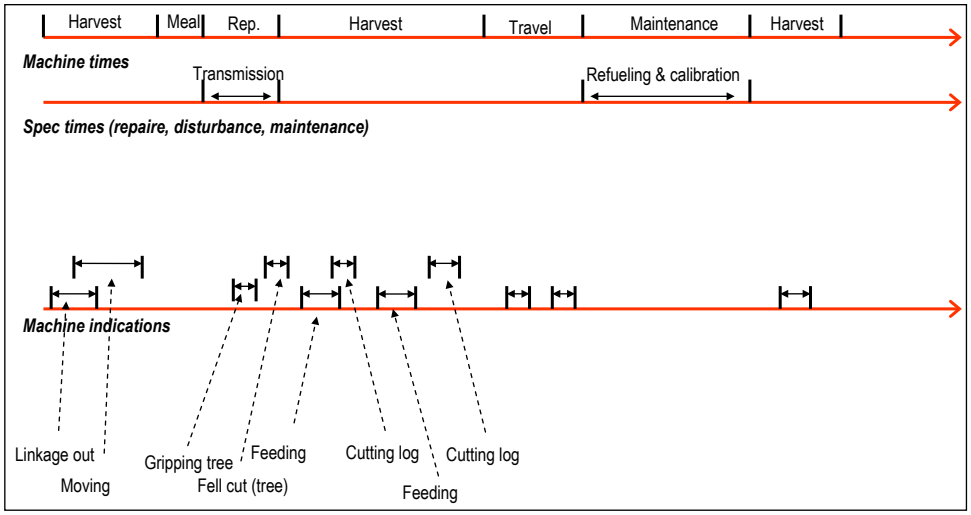


Figure 44. Example time registration in operational monitoring message (date and time format shortened)

Individual vs combined machine work times

The machine work times must include lost times (unutilized times). This means that the sum of all times registered as machine work times (element MachineWorkTime) must be equal to the total report interval as registered in element ReportInterval. The sum of the operator work times (element OperatorWorkTime) will normally not be the same as the length of the time interval in ReportInterval.

It was assumed in the first draft versions of StanForD 2010 that only individual times are to be registered in the new operational monitoring messages (mom). However this solution was not acceptable to all manufacturers. A need for also being able to register different kinds of combined or aggregated times was identified. Two separate structures for registering individual and combined machine work times are therefore included in the mom-file. Individual instances of "Processing" can be registered in the IndividualMachineWorkTime or the sum of "Processing" times can be registered in CombinedMachineWorkTime. The structures are described in the figure 33. Observe that short down times may also

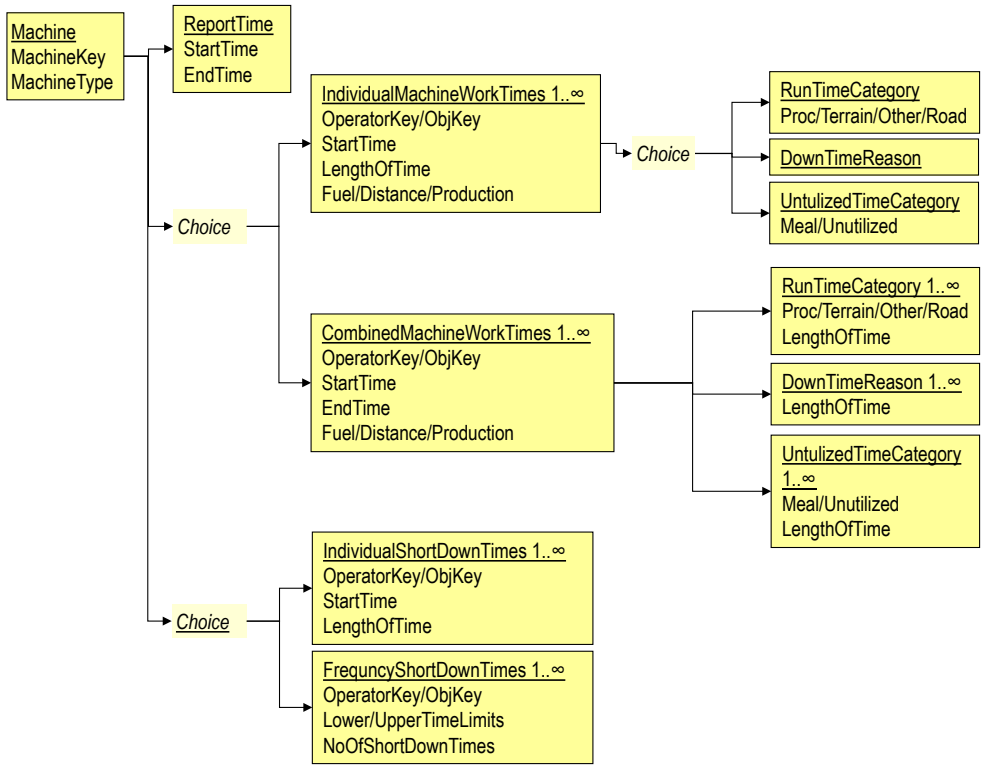


Figure 45. Diagram describing differences between combined and individual times in operational monitoring message.

The most visible difference between these structures are the only one single individual time (run time, down time or unutilized time) per element MachineWorkTime can be registered, while several different times can be included per element MachineWorkTime when registering combined times.

Combined times makes it:

- More complex to compare mom-data with hpr and fpr (including time stamps).
- More complex to calculate key figures for any time interval.

Operator work times

The operator work times are always to be registered as individual times in the following structure. The following enumerations exist for OtherOperatorTimeCategory:

- MachineRunTime
- Waiting for repair
- Trailer transportation
- Meal break
- Planning outside machine
- Travel to work outside machine
- Other work outside machine

Observe that the element MachineRunTimeCategory must be analyzed in order extract information regarding exactly what machine work an operator has been carrying out, for example Processing, Terrain travel, Other work or Road travel.

A significant difference between the operator work times and the machine work times is that the operator work times can be overlapping.

Examples

Example 1

Below is an example with a time line with a number of different work and down times occurring in a harvester:

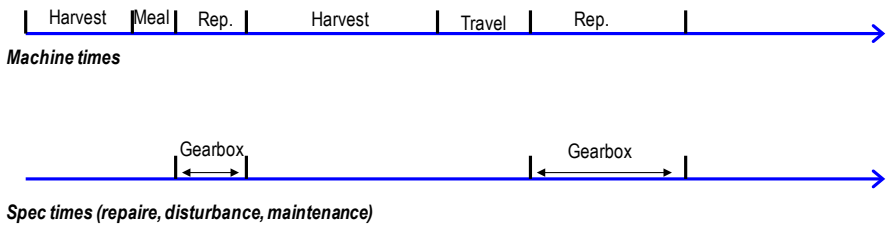


Figure 46.

Two xml-examples, describing how this time line may be registered, is included below.

Only individual times reported in mom (observe ObjKey is excluded):

```
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <MonitoringStartTime>2001-12-17T09:30:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>10</MonitoringTimeLength>
        <IndividualMachineRunTimeCategory>Processing</IndividualMachineRunTimeCategory>
</IndividualMachineWorkTime>
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <MonitoringStartTime>2001-12-17T09:40:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>15</MonitoringTimeLength>
        <IndividualUnutilizedTimeCategory>Meal break</IndividualUnutilizedTimeCategory>
</IndividualMachineWorkTime>
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <MonitoringStartTime>2001-12-17T09:55:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>19</MonitoringTimeLength>
        <IndividualMachineDownTime>
                <Repair>
                        <CarrierRepairReason>
                                <Mechanical>Gearbox</Mechanical>
                        </CarrierRepairReason>
                </Repair>
        </IndividualMachineDownTime>
</IndividualMachineWorkTime>
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>0
        <MonitoringStartTime>2001-12-17T10:14:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>51</MonitoringTimeLength>
        <IndividualMachineRunTimeCategory>Processing</IndividualMachineRunTimeCategory>
</IndividualMachineWorkTime>
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <MonitoringStartTime>2001-12-17T11:05:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>5</MonitoringTimeLength>
        <IndividualMachineRunTimeCategory>Terrain travel</IndividualMachineRunTimeCategor>
</IndividualMachineWorkTime>
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <MonitoringStartTime>2001-12-17T11:10:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>25</MonitoringTimeLength>
        <IndividualMachineDownTime>
                <Repair>
                        <CarrierRepairReason>
                                <Mechanical>Gearbox</Mechanical>
                        </CarrierRepairReason>
                </Repair>
        </IndividualMachineDownTime>
</IndividualMachineWorkTime>
<IndividualMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <MonitoringStartTime>2001-12-17T11:35:47.0Z</MonitoringStartTime>
        <MonitoringTimeLength>22</MonitoringTimeLength>
        <IndividualUnutilizedTimeCategory>Meal break</IndividualUnutilizedTimeCategory>
</IndividualMachineWorkTime>
```

Combined (aggregated) times reported in mom:

The significant differences (aggregated data) between combined and individual times are indicated using bold text. Observe that the down times are registered individually:

```
<CombinedMachineWorkTime>
        <OperatorKey>2</OperatorKey>
        <StartTime>2001-12-17T09:30:47.0Z</StartTime>
        <ObjectKey>2</ObjectKey>
        <CombinedMachineRunTime>
                <MachineRunTimeCategory>Processing</MachineRunTimeCategory>
                <TimeLength>61</TimeLength>
        </CombinedMachineRunTime>
        <CombinedMachineRunTime>
                <MachineRunTimeCategory>Terrain travel</MachineRunTimeCategory>
                <TimeLength>5</TimeLength>
        </CombinedMachineRunTime>
        <CombinedMachineDownTime>
                <Repair>
                        <CarrierRepairReason>
                                <Mechanical>Gearbox</Mechanical>
                        </CarrierRepairReason>
                </Repair>
                <TimeLength>19</TimeLength>
        </CombinedMachineDownTime>
        <CombinedMachineDownTime>
                <Repair>
                        <CarrierRepairReason>
                                <Mechanical>Gearbox</Mechanical>
                        </CarrierRepairReason>
                </Repair>
                <TimeLength>25</TimeLength>
        </CombinedMachineDownTime>
        <CombinedUnutilizedTime>
                <UnutilizedTimeCategory>Meal break</UnutilizedTimeCategory>
                <TimeLength>37</TimeLength>
        </CombinedUnutilizedTime>
        <CombinedEndTime>2001-12-17T11:57:47.0Z </CombinedEndTime>
</CombineMachineWorkTime>
```

Example 2

How to handle operator work times when merging data from the same operator working on different machines?

The following example illustrates the question above:

1. An operator starts working on a machine in the morning.
2. The machine breaks down during the day and operator is forced to use a reserve machine.

An important consideration is whether this is primarily a StanForD issue or an issue that should be considered in the administrative system receiving data from the machine systems.

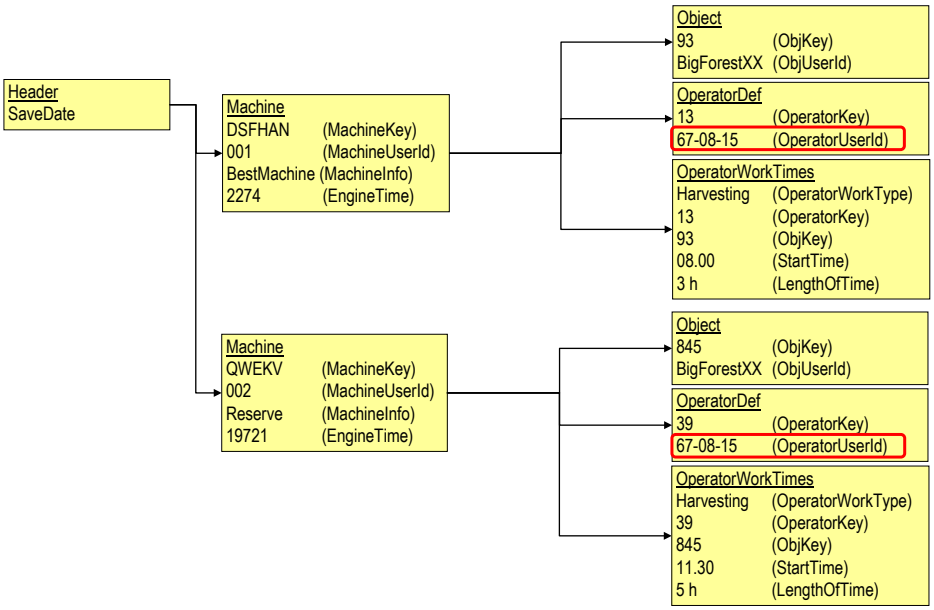


Figure 47. Simplified data structure when merging operational monitoring data from two separate machines into one message. Observe that the way to identify the operator is to use the *OperatorUserId*, the operator must use exactly the same *OperatorUserId* in both machines in order to make it possible to get the total no of work hours for an individual operator.

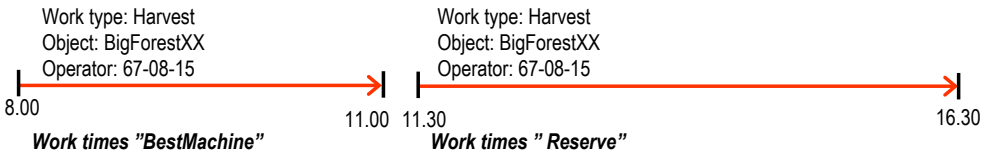


Figure 48. The following timeline can be calculated using the data in the example above.

Other messages

STANFORD 2010 ENVELOPE

Included in the standard is also a message for communicating several files together. This message is called an envelope (.env). It is supposed to be used for object geoagraphical packes. It is also recommended that the manufacturers are able to use envelopes including a complete set of messages for harvesting instructions (oin, pin, spi).

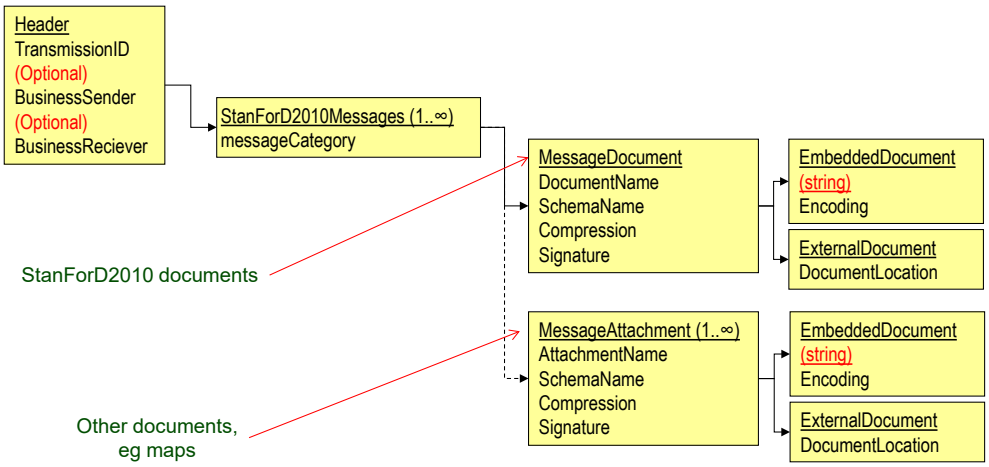


Figure 48b. Basic structure of the StanForD 2010 envelope

A document may be either embedded into the envelope or it may be stored as a separate file (Extrnal document). An embedded document may be encoded (for example a compressed file).

It should be possible to use StanForD 2010 envelope as a StanForD 2010 message container for communication but it must also be possible to generate the basic individual StanForD 2010 messages without envelope wrapper in the forest machines.

The StanForD 2010 messages are not totally well-formed because of the general structure of XML-documents. So all characters in embedded documents "<" and ">" have to be changed to escape sequences < and >.

All binary files (for example compressed files and pictures) that are embedded in StanForD 2010 envelope message must be encoded using Base64 encoding.

StanForD 2010 Envelope should be used to enclose additional files in case of ogi and ogr. Management of additional files in messages is easier if they are wrapped with envelope.

Extension elements

An element called *Extension* exists in all different messages at many different locations in order to make it possible to add manufacturer or user specific information outside the standard.

The Extension-element was designed to be as flexible as possible. The detailed documentation of the extension element can be found from the chapter 10 of "StanForD 2010 – Naming and design rules" –document.

However there is small possibility that element and attribute names inside extension elements may clash between StanForD 2010 messages of different organizations. Therefore we should use organization specific namespaces to distinguish names between them. Namespace should be chosen so that we can be sure that other organizations don't use that name. Good candidate for namespace is derived from the domain name of organization.

EXTENSION EXAMPLE INCLUDING NAMESPACE

The root element is as usual:

```
<?xml version="1.0" encoding="utf-8"?>
<HarvestedProduction areaUnit="ha" diameterUnit="mm" lengthUnit="cm"
volumeUnit="m3" weightUnit="kg" version="3.0" messageType="hpr"
versionDate="2014-12-15+02:00" xsi:schemaLocation="urn:skogforsk:stanford2010
HarvestedProduction_V3p0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="urn:skogforsk:stanford2010">
        <HarvestedProductionHeader>
```

There is no additional namespace defined above. Only the xsi prefix and the default namespace (i.e. the one that is assumed in this element and all of its children without a prefix) are defined, like in any ordinary SF2010 file. In the <Extension> block of a product/assortment that needs to have some custom data we could have:

```
<ProductDefinition>
  <ProductKey>313</ProductKey>
  <ClassifiedProductDefinition>
    <ProductName>PS 1</ProductName>
…
    <Extension>
      <PreSelectionProduct xmlns="http://www.deere.fi/xml/forestry">
        <TopSaw>0</TopSaw>
        <FeedingSpeed>0</FeedingSpeed>
      </PreSelectionProduct>
    </Extension>
```

Here some manufacturer sepcific elements <PreSelectionProduct> with its children have been added. Beginning from and including this element, the default namespace, "urn:skogforsk:stanford2010" is overridden with our own. This makes the <PreSelectionProduct> along with all of its children to belong in our "http://www.deere.fi/xml/forestry" namespace. When the block ends, </Extension> again belongs to the namespace "urn:skogforsk:stanford2010".

Regarding the namespace used above; the URL does not have to lead anywhere and this one does not. It is not checked by any parser or validator or anything, it is only supposed to be rather unique. However, some companies may place some relevant data available on the page indicated by the namespace URL.

Use cases harvesting

As a basis for the continued development of StanForD 2010 use cases is a good tool. It is also a good way to illustrate how the standard should be used and implemented in the machines.

Three use cases for harvesting are illustrated below. The first one illustrates how to use the StanForD 2010 in a way similar to the way StanForD normally is used in Sweden today. The second example is to a certain degree based on how the Finnish Apteri-system is constructed while the third alternative is flexible alternative where new updated products can be sent and used at any time.

APT-FILE (SIMPLE HARVESTING)

Step 1.

Complete instructions sent out before harvesting at an object is started. All products that are to be used are included in the pin-message. Basically the pin and the oin messages together function as an old apt-file. Four separate products are included in the example as is illustrated in the diagram below.

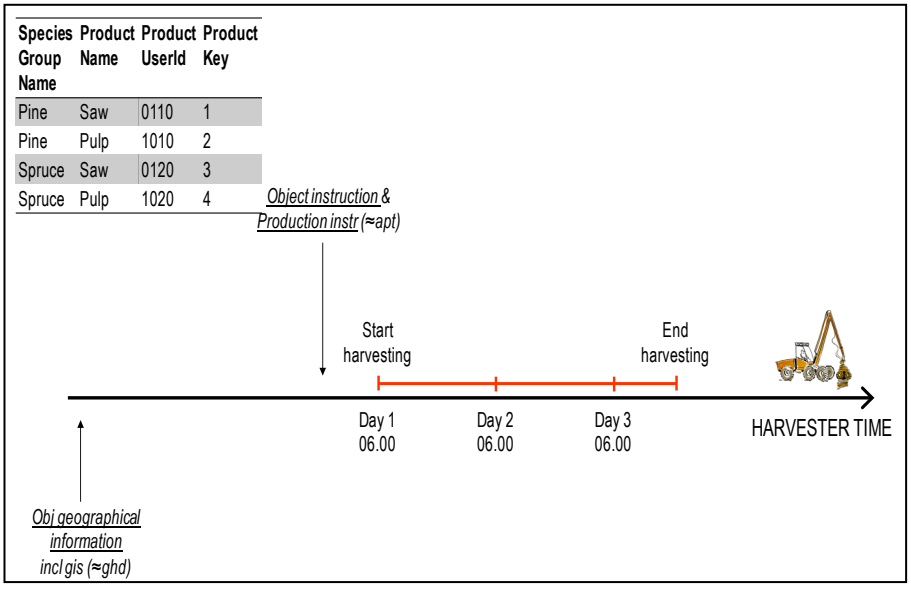


Figure 49.

Step 2.

Production, operational monitoring and quality control data are reported once a day.

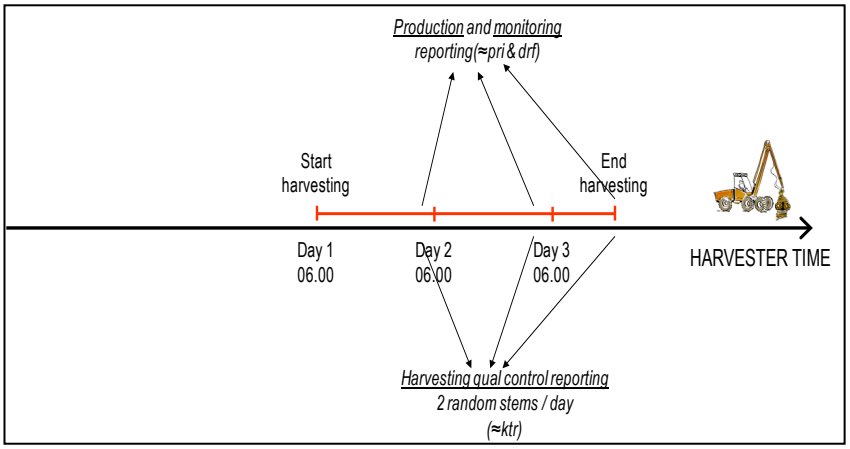


Figure 50.

Step 3.

A new set of instruction are sent out to the machine before starting on a new object. The prices of the products have been changed and the products have consequently been given new modification dates. The old product definitions sent in step 1 are replaced by the new ones.

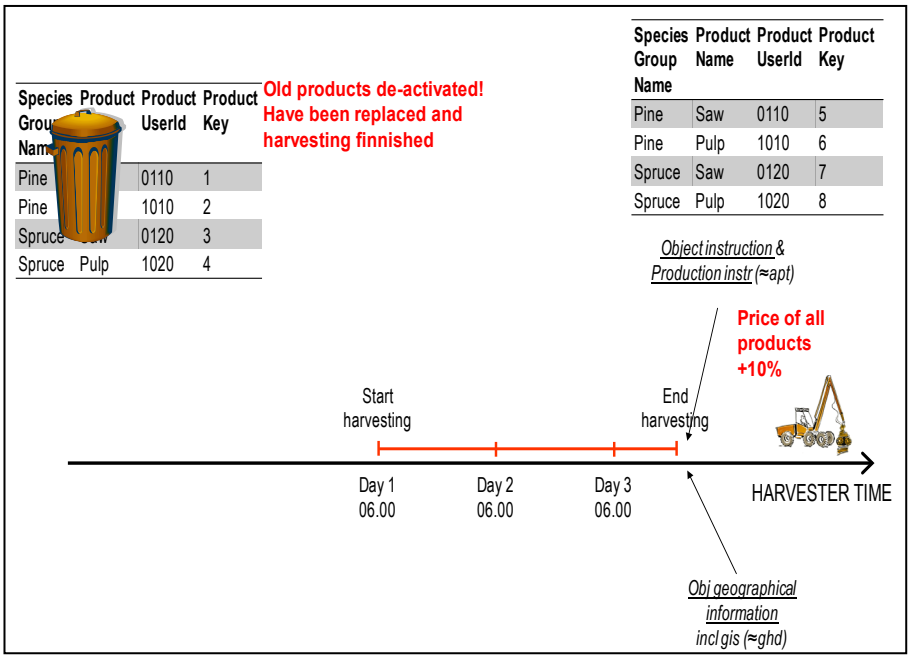


Figure 51.

The complete old product definitions may in theory but deleted once they have been replaced by the new. However it is probably a good idea to only "de-activate" them and save them in some kind of archive if someone wants to be able to go back and analyse the production of a specific object at later time.

APTERI HARVESTING

Step 1.

Product instructions (pin) are sent continuously when updated to the machine independently of harvesting object.

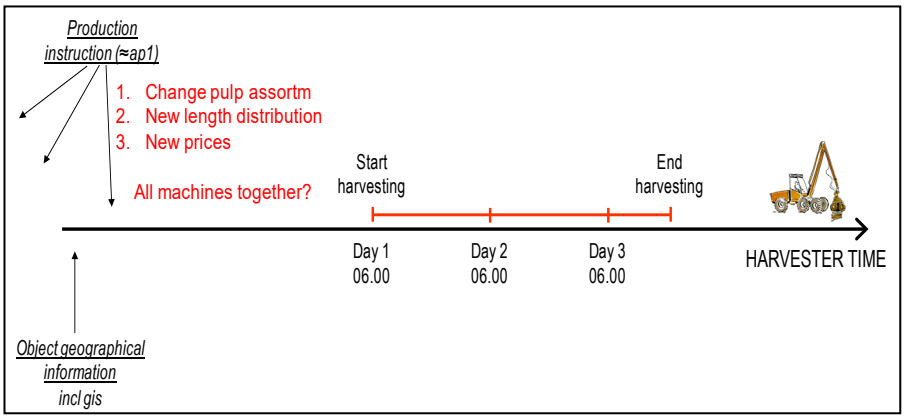


Figure 52.

Step 2.

Object instruction (oin) sent to harvester just before starting at an object. This instruction includes references (ProductUserId) to the product that are supposed to be harvested at the object.

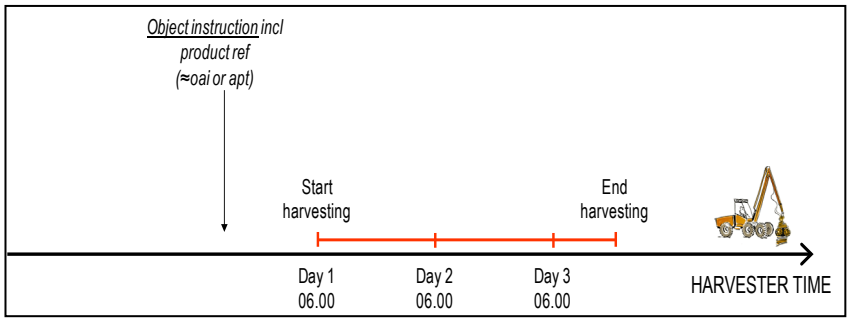


Figure 53.

Step 3.

Production, operational monitoring and quality control data are reported continuously sent from machine. Updated geographical information is sent when object is finished. It is suggested that an automatic routine is implemented in the harvesters whereby it is possible to set that these reports are to be generated for example twice a day as well as when an object is completed.

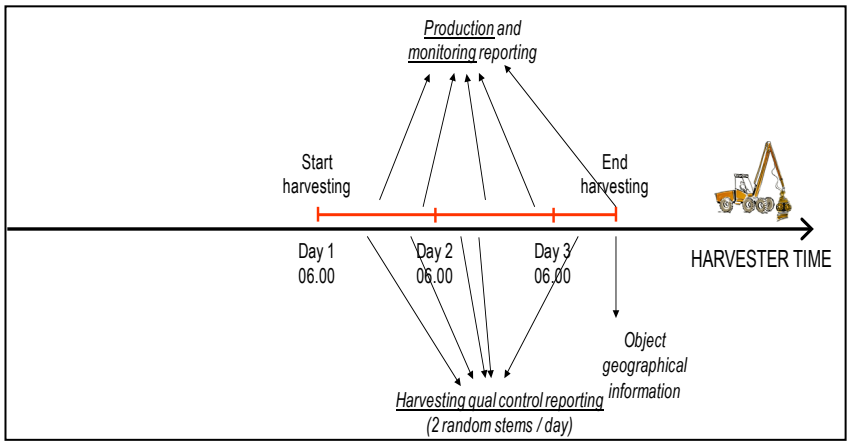


Figure 54.

FLEXIBLE HARVESTING

The third *Flexible* example is very similar to the *Apteri* example. The significant difference is that it should be possible to send updated object (oin) and product (pin) instructions to the harvesters at any time before or during production at a specific harvesting object in order to for example activate a certain product or change the distribution matrix of an already existing product.

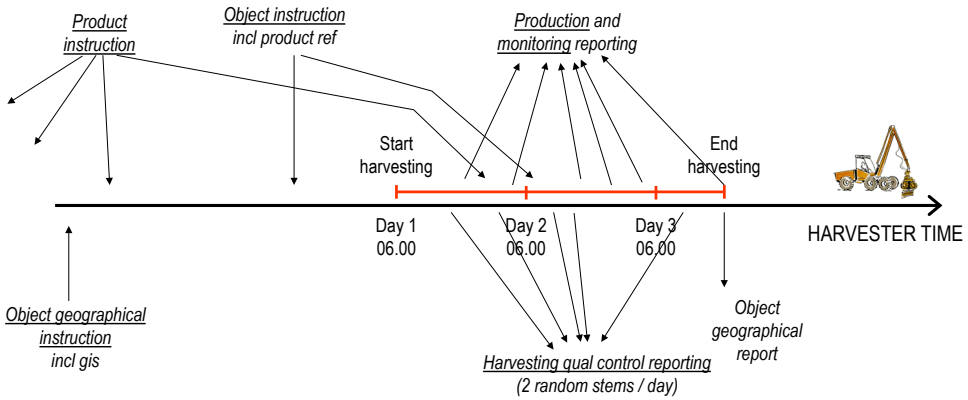


Figure 55.

Below follows a slightly more concrete example of how to administrate product and object instructions sent to the harvester.

Step 1.

Product instruction (pin) including four products is sent to harvester

ProductName	PineSaw	PinePulp	SpruceSaw	SprucePulp
ProdUserId	0110	1010	0120	1020
ModificationDate	10-01-18	10-01-18	10-01-18	10-01-18
LengthClasses	34, 40, 46, 49	38, 40, 45	37, 43, 46, 49	37, 41, 44
DiamClasses	18,22,24,30	5,10,15,18	18,22,24,30	5,10,15,18
PriceMatrix	45 €/m3 ...	20 €/m3 ...	42 €/m3 ...	25 €/m3 ...

Step 2.

New object instruction (oin) sent to harvester

ObjectUserId	ModificationDate	Product references
PinkForest_1380	10-01-19	PineSaw (0110), PinePulp (1010), SprucePulp (1020)

Step 3.
New harvesting object started with the following products according to oin message

ObjectKey	ObjectUserId	Products used
38	PinkForest_1380	PineSaw (0110), PinePulp (1010), SprucePulp (1020)

Step 4.
New product instruction for "PineSaw" (new diameter classes and increased price) sent to harvester during harvesting at "PinkForest_1380"

ProductName	PineSaw
ProdUserId	0110
ModificationDate	10-02-01
LengthClasses	34, 40, 46, 49, 55
DiamClasses	18,22,24,26,30,32
PriceMatrix	51 €/m3 …

Operator is asked: Replace old PineSaw with new?

Answer: Yes!

New object instruction (oin) including product "SpruceSaw" sent to harvester during harvesting at "PinkForest_1380"

ObjectUserId	ModificationDate	Product references
PinkForest_1380	10-02-01	PineSaw (0110), PinePulp (1010), SpruceSaw (0120), SprucePulp (1020)

Operator is asked: Update present object or start a new object?

Answer: Update present object!

Step 5.
Harvesting at object "PinkForest_1380" is continued with updated product "PineSaw" and new product "SpruceSaw"

ObjectKey	ObjectUserId	Products used
38	PinkForest_1380	PineSaw (0110), PinePulp (1010), SpruceSaw (0120), SprucePulp (1020)

HARVESTER SYSTEM LAYOUT

Depending on whether we want to be able to handle all examples described above or not we have to design our receiving systems in the harvester in different ways.

Apt-file harvesting

It would probably be enough to have a system identical with the present with the exception that it can also receive new files that are internally converted into an apt-structure in the *apt-file* use case. However not much is gained. Most of the advantages with StanForD 2010 will not be used.

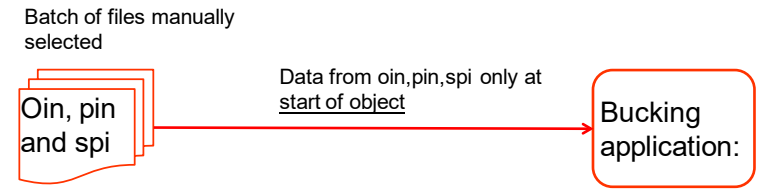


Figure 56.

Apteri harvesting

In the case of an *Apteri* scenario a database for administrating the instructions is needed in the harvester.

In the *Apteri* case data is sent from the database only when starting at a new harvesting object.

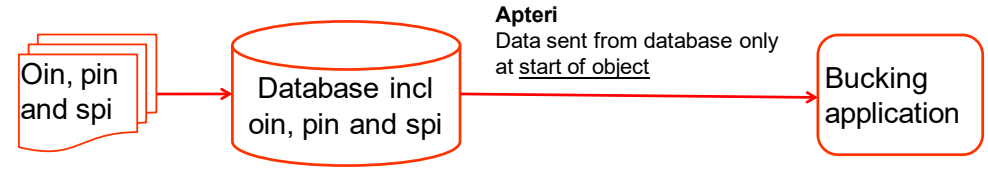


Figure 57.

Flexible harvesting

As in the case of an *Apteri* scenario the *Flexible* scenario also needs a database for administrating the instructions.

In the *Flexible* case data can be sent from the database at any time, also during harvesting at the object.

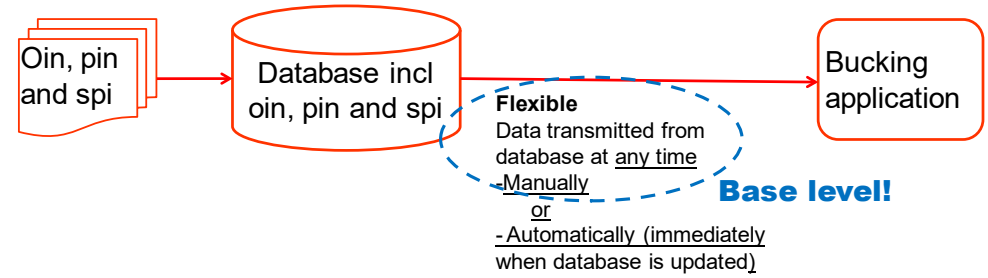


Figure 58.

The *Flexible* system could be manual and/or automatic. The first step is to make the system manual which means that the operator manually decides when to import a new or updated product (pin) from the database to the bucking application. The same applies to SpeciesGroups (spi) and Objects (oin). The manual flexible system is the first base level that is strongly recommended to implement. In an automatic system an updated or new product will be

automatically imported into the bucking application. The table below describes four different events where pin and oin files are received by the harvester and added to the database.

File recieved	Flexible - manual	Flexible - automatic
New oin	Used when starting at new object.	Used when starting at new object.
New pin	Used instantly if manually imported by operator.	Used instantly if ProductUserId is included in active oin, automatically imported by bucking application.
Updated oin	Used instantly if ObjUserId is same as in active oin, manually imported by operator.	Used instantly if ObjUserId is same as in active oin, automatically imported by bucking application.
Updated pin	Used instantly, manually imported by operator.	Used instantly if ProductUserId is included in active oin, automatically imported by bucking application.

Observe that it is strongly recommended that the harvester system is a manual flexible system. To also include automatic functionality is optional.

Also observe that it should always be possible for operator to edit products and species groups during production if attribute modificationRestricted is false. ProductKey and SpeciesGroupKey must be updated if product or species group is modified.

Other use cases

QUALITY CONTROL HARVESTING

The data flows regarding harvesting quality control in Sweden can in most cases be illustrated as in the diagram below. The harvester sends randomly or operator selected stems to the calliper. The operator measures the stem and sends the data back to the harvester. The data may be used for calibrating the machine or for only checking the machine. Some additional data (calibration log, rejected stems etc) is added to the hqc-file and subsequently sent to the logging organization.

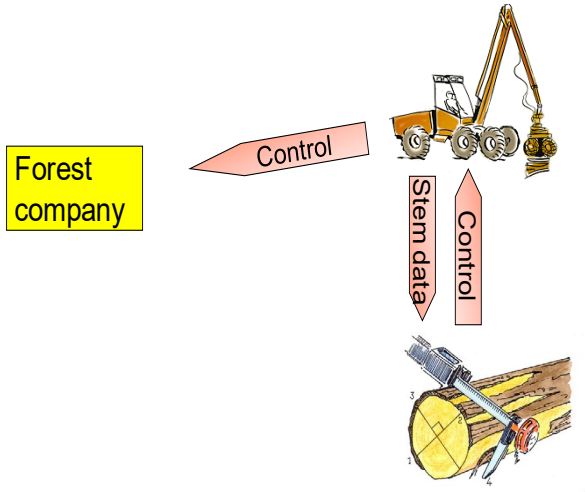


Figure 59.

However there also exist systems for a third party control (auditor) which makes the data flows more complex. The data may in this case be routed in several different ways as illustrated below (green arrows).

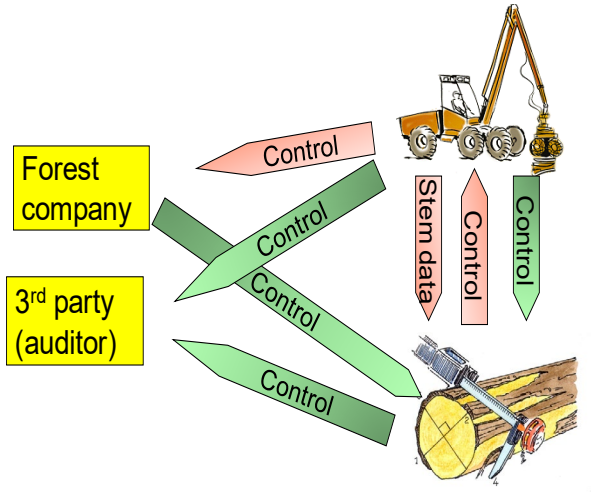


Figure 60.

Based on the diagrams above it can be concluded that the following issues must be dealt with:

- Direct communication from calliper to logging organization/auditor needed
- Quality control data must be possible to send from harvester to calliper for auditor
- Production data might be confidential which means that it must not be given to auditor

All data communication connected to harvesting quality control will be using the hqc message.

REPORTING OF HARVESTED PRODUCTION (HPR)

It is of great importance for the implementation of StanForD 2010 that reporting of production data:

- is simple for operators.
- gives no significant differences between manufacturers.
- is robust (trustworthy, simple to check for missing or duplicate data)

It was first assumed that no production data is to be sent more than once. This could be described as forbidding aggregated reporting. However finding a suitable solution for partial reporting was not possible. All manufacturers expressed a very clear desire to keep rules concerning partial reporting outside the standard including any elements for FileOrder or BatchNumber.

Since StanForD 2010 does not include any rules concerning partial reporting it must be possible to generate a complete hpr-message including all stems harvested at a harvesting object with the first StemNumber being 1 and the last

StemNumber being equal to the total number of stems in the whole file. Until there is some commonly used solution for partial reporting this will probably be the recommended solution. It is a robust and simple reporting solution generating large files. An advantage with this way of reporting is that it does not really matter if one report is lost during harvesting at an object.

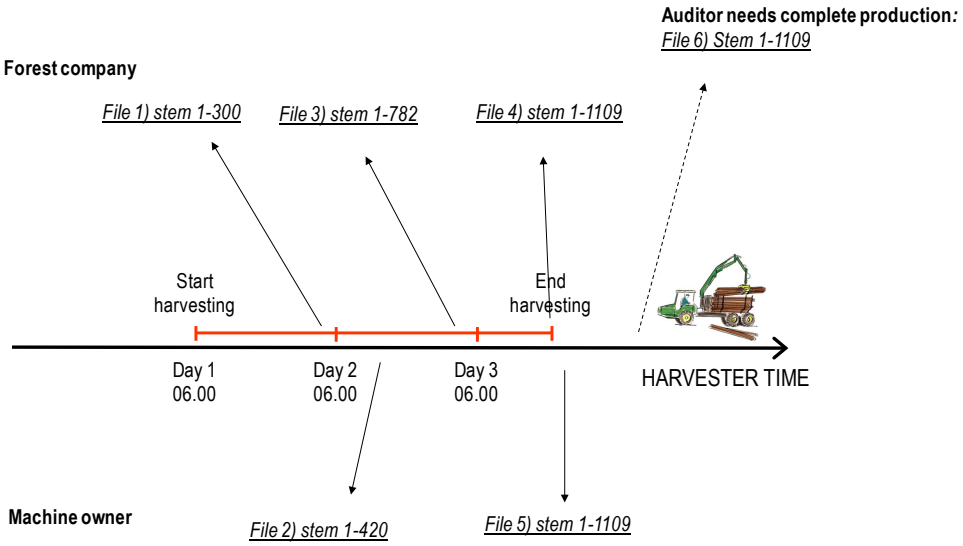


Figure 61. Illustration of alternative where the only requirement is that it must be possible to report total production.

Example of five hpr-files sent from one harvesting object:

Machine-Key	Obj-Key	Stem-Count	1st stem	Last stem	HarvestDates	Creation-Date
3XE2_Pink	126	119	1	119	2010-02-03 – 04	2010-02-04
3XE2_Pink	126	343	1	343	2010-02-03 – 06	2010-02-06
3XE2_Pink	126	519	1	519	2010-02-03 – 08	2010-02-08
3XE2_Pink	126	1021	1	1021	2010-02-03 – 11	2010-02-11
3XE2_Pink	126	1411	1	1411	2010-02-03 – 14	2010-02-14

It is simple to check the total number of stems as well as the first and the last stem number in each file. The first StemNumber must be equal to 1 and the last StemNumber must be equal to the total number of stems in the whole complete hpr-file. Based on this information we can easily identify that this is a complete hpr-file including all harvested stems.

Reporting of production and monitoring data must be simple:

- Always possible to create a complete hpr-report including all previously reported stems (default alternative)
- Partial reporting possible to implement (no rules exist)

Figure 62 illustrates two buttons for generating hpr-messages and a possible setting for automatic generation of hpr-messages.

Figure 62. Example gui for generating hpr-files

It is possible to report only stems processed since last reporting when generating a new partial hpr-message.

Figure 63. Example illustrating partial reporting of hpr-files.

It is possible to create partial reports per stem number, per time, per operator, per file size etc.

It must always be possible to generate a complete total hpr-report.

Figure 64. Example illustrating total reporting of hpr-files.

A crucial issue is how to retrieve data from a specific harvester if the receiving system has found that data is missing.

If we allow aggregated files and require that all manufacturers can generate a complete hpr-file as described above we only need to tell the operator to create a new complete hpr-file (illustrated in previous section).

If needed by a receiving system it is possible to create a unique message identity by combining MachineKey and CreationDate. This may for example be used when there is a need to refer to certain hpr-files in a papiNet MeasuringTicket.

OPERATIONAL MONITORING

Daily reporting from harvester, all times covered since last reporting.

New standard makes it possible break report in many ways, eg day, week, month, quarter, year

Calculate key figures eg per week

Week	22	23	24	25	26	27
TU	84	83	85	73	85	84
m3/G15	30	31	27	30	32	20

Week 25? Have a look at Tuesday!

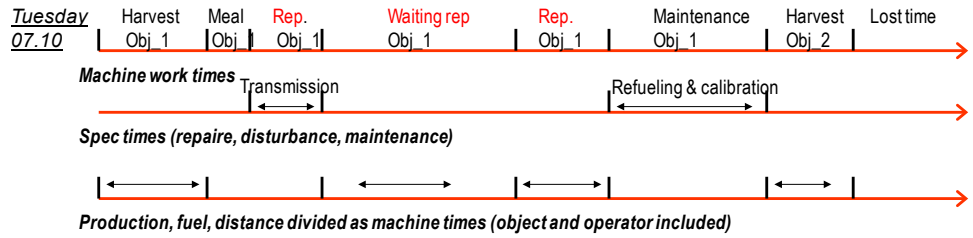


Figure 64.Illustration of time lines included in mom-file.

Week 27? Average stem volume very low!

FORWARDING

Reporting of forwarded production is done in the same way as in the harvester. Creating a new object in the forwarder can be done based on several different data sources:

- instruction sent from forest company
- harvester production data
- manual operator input

Instruction from forest company

The following figure illustrates possible information flows in case of a complete instruction being created by the forest company (office). Observe that harvester data is only used for informing forwarder about harvested volumes and not for creating new object.

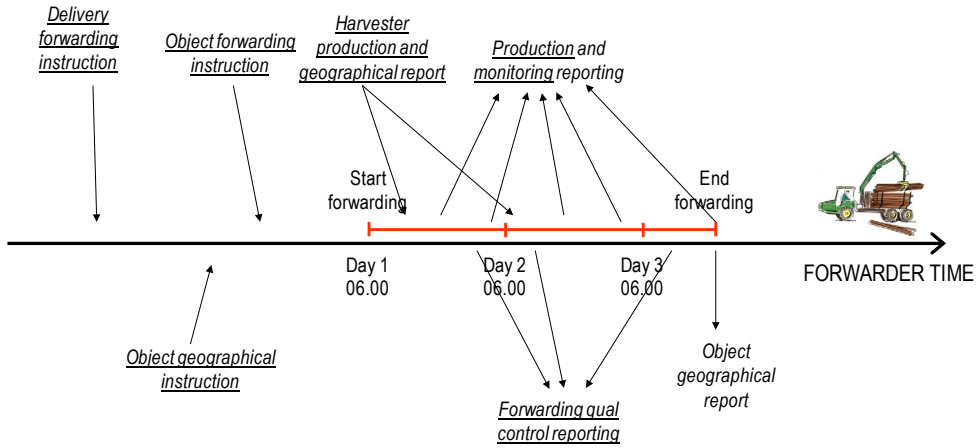


Figure 65. Example illustrating forwarder information flows where object is started using foi- and fdi-files.

Harvester production data

The following figure illustrates possible information flows in case of only having harvester data available. Observe that the operator manually must define/add Delivery- and LocationDefinition.

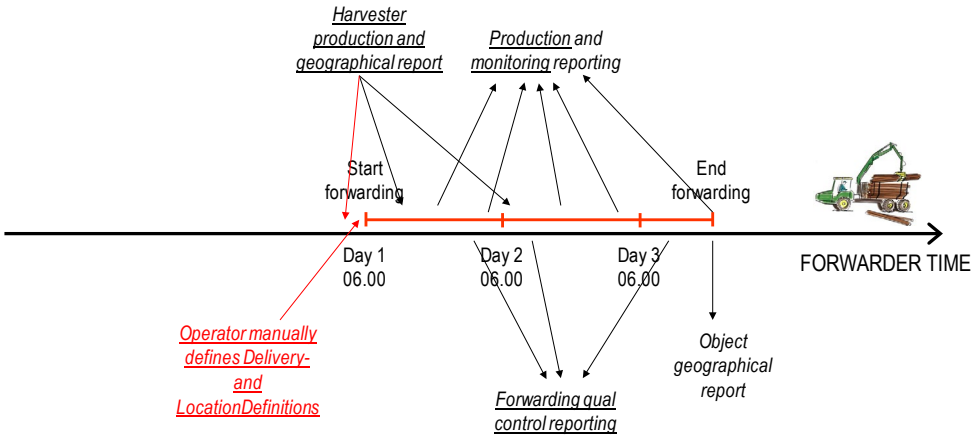


Figure 65. Example illustrating forwarder information flows where object is started using harvester production data (hpr-file).

Manual operator input

The following figure illustrates possible information flows in case of having no data from harvester or forest company (office) available. Observe that the operator manually must add all required information manually.

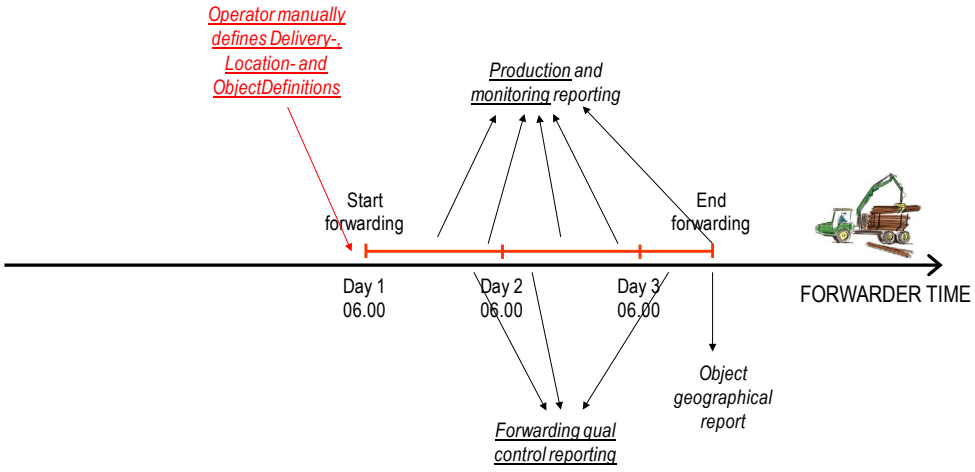


Figure 66. Example illustrating forwarder information flows where object is started using information printed on paper.

Operator interaction during harvesting

A fully automatic system for administrating harvesting objects and product definitions is probably not possible today. The operator therefore has to interact with the harvester applications. Below is a list of situations where the operators must interact with these applications.

- Notified when new product is to replace old product:
  - Operator can accept or deny.
- Notified when new Object instruction is to replace old
  - Operator can select replacement or new harvesting object.
- Notified if product in Object instruction is missing in harvester

- Operator may request correct product definition from forest company

- Should be possible to:
  - Update product and species group at any time
  - Manually open/activate any instruction at any time
  - Set whether all product definitions (in pin message) are reported parallel to normal production

Questions

Connecting Products and SpeciesGroups to buttons

-- How do we connect a product to a specific button?

Today the order of the assortments and species are often used when connecting a certain assortment to a certain button in the harvester. With the new standard no such an order exists. We can for example mix products from different species in any way we want in StanForD 2010. This issue was solved by including the elements:

- SpeciesGroupPresentationOrder: Indicates the order of SpeciesGroups. May be used in presentation tools in order to show the SpeciesGroups in a certain order, for example if pine is always to be presented before spruce and birch in a printed report. The element should be considered as an optional hint given by forest company.
- Product PresentationOrder: Indicates the order of products per SpeciesGroup. May be used in presentation tools in order to show the products in a certain order, for example if sawlog product are always to be presented before pulp wood and fuel wood in a printed report. The element should be considered as an optional hint given by the forest company.

Missing ProductDefinitions and SpeciesGroupDefinitions

-- Use case examples of sending out oin, pin, spi files. What if some are missing? What if some are replaced?

This is a problem that perhaps might occur also in the old Finnish Apteri system. The idea is that the operator should be notified if some definition is missing. The operator must be able to start working at a site even if a specific product is missing. The normal procedure will probably be that the operator immediately contacts the person responsible for the harvesting in order to receive further instructions.

Replacing ProductDefinitions and SpeciesGroupDefinitions

-- Use case examples of sending out oin, pin, spi files. What if some are replaced?

The idea is been that the operator should always be notified when an old product is about to be replaced by a new one. It should also be possible for the operator to deny the replacement.

Definitions in single or multiple messages

-- Do the products come in to machine as one single file, or multiple files, or in an XML-envelope?

ProductDefinitions, SpeciesGroupDefinitions or DeliveryDefinitions can be sent individually or severeal merged into one file. There is therefore no significant advantage in using the envelope for example when sending severel ProductDefinitions together. The envelope will primarelly be used for merging all relevant files for a geographical logging instruction, including both ogi, shp, shx, mif, jpg, dbf, pdf etc.

Updating definitions

-- When should updating of prodcuts be allowed?

Basically at any time. A new *ProductKey* and *ModificationDate* must be set if an old product is replaced by an old or if it is modified in any way.

History of old/replaced definitions

-- What kind of "history" should be kept of changes made in the machine? Should this be reflected in production files?

Very good question! A minimum to save the data included in the ProductDefinition in the hpr-message (ID, LengthDef, DiameterDef, PriceDaf) as long a the production data is saved. The complete ProductDefinition as defined in pin-message must be saved in the machine as long as the product is not replaced with new product with the same UserId but updated ProductKey.

Products in GUI

-- What is the role of user-ids to the operator. Should we e.g. have the IDs visible in GUI?

I have not really thought about this but I guess that the most important thing to include in GUI is the ProductName but tha ProductInfo, ProductVersion and ProductUserId are also of interest in many cases. The same is true also for the species groups.

Restrict modification of product

-- Should it be possible for forest companies to restrict modifications of ProductDefinitions in harvesters?

Yes, this is possible by using the attribute modificationRestricted.

Defining important concepts and key figures

StanForD 2010 should include recommended definitions of key figures and basic concepts, for example:

- Mean stem volume should include also unclassified volumes. Should be possible to keep different processing categories separate.

- Stem: There are no conditions for registering a stem in a StanForD2010 files as there were in the old standard version. This means that a stem may only include small unclassified logs and still be registered in an hpr-file.

- Volume (solid): solid volume should always be based on filtered diameter values and physical length. The minimum modul length for calculating solid volume is 1 dm, it is however suggested that 1 cm modules are to be used. The diameters must always represent the lowest measured diameter up to that point (for example no average per dm module). The systematic difference between using the formula for a cylinder (average diameter or mid diameter) or a cut cone is not significant (<0.1%), it is thus possible to use either of these formulas.

  Registered diameters incl top diameters: filtered values (no increasing values) at the position. No average values allowed.
  Annotation for diameter vector is:
  Diameter at heights defined by diameterPosition attribute (representing the actual point of measuring). Refers to filtered values on bark. Same values as used in bucking optimisation and calculations of solid volumes. No systematic error when comparing log volumes and volume calculations based on DiamValue.
  Diameter values must start at heighet 0 cm from stump. Extrapolated diameters at butt end are to be registered. Height of first and last measured diameter registered in DiameterMeasuredStart and DiameterMeasuredLas.t

- Sound knot function

- Distribution bucking

- Multi tree handling

Other issues

- Security needs? Checksum?

Appendix 1) Reporting of hpr-files

REQUIREMENTS AND RECOMMENDATIONS

INTRODUCTION

This document deals with some issues that are very close to the limits of StanForD. It is highly relevant to question whether the ideas and proposals in this document should be included in StanForD 2010. However, if it is not easy to send and receive an hpr-file we risk that the new standard will fail.

It is of great importance for the implementation of StanForD 2010 that reporting of production data:

- Is simple for operators.
- Gives no significant differences between manufacturers. A receiving system should be able to handle hpr-reporting from all harvesters.

When developing systems in Sweden for using pri-files it has been noted that there are some significant differences between different manufacturers when it comes to how pri-files are generated. One of the main reasons for this is that the present standard allows both aggregated and non-aggregated (partial) pri-files. It is therefore important that we do try to decide on a set of rules for how to report normal production data from forest machines.

StanForD data will always, as a first step, be sent as one message from one single machine. StanForD messages can be used for merging data from several machines, but this is a secondary objective. This means that production data from several machines and several objects/sub-objects can be included in the presented structure. However, normal production reporting will be per machine and harvesting object.

The following preliminary reporting rules were the starting point for long discussions in the StanForD 2010 working group:

- The primary alternative should be to report data report data only once!
- Partial reporting shall be used for fpr, hpr, hqc, mom
- Final partial reporting must be possible also when object is finished
- Only exception is thp which always must include all data from an object = an aggregated file.

The idea of implementing some kind of BatchNumber element, similar to the old FileOrder (var22_t1) was also discussed but rejected as described under section *Other issues*.

KEEPING TRACK OF HPR-FILES

The receiver of data must make certain that no stems are missing and that duplicate stems are not included. There is presently only one way to do this and that is through the existing StemNumber which is always reset when starting on a new object together.

Observe that the idea of having a BatchNumber-element has been discussed but dismissed.

REQUIREMENTS AND RECOMMENDATIONS

Since no agreement regarding partial reporting could be reached it was decided 2010-06-15 that the only definite requirement included in the standard is that it must be possible to generate a total hpr-file including all harvested stems at an object. Only one object per report shall be included. The first StemNumber must be equal to 1 and the last StemNumber must be equal to the total number of stems in the whole complete hpr-file.

It has been noted after the meeting in at Åland (2010-06-15) that reporting of total hpr-files might cause problems on large objects due to the fact that the size of hpr-files is significantly large. The generation of an hpr-file in the forest machine application may take a significant amount of time and consume a large part of the CPU capacity. Skogforsk has therefore written a recommendation concerning how partial reporting could be carried out in the harvester. Whether or not the manufacturers follow these recommendations is not a StanForD 2010 issue.

Total reporting (StanForD requirement)

The StanForD requirement is that all manufacturers must implement a function for reporting the total (aggregated) production for a harvesting object. This approach could be illustrated using the following figure where only total production is reported.

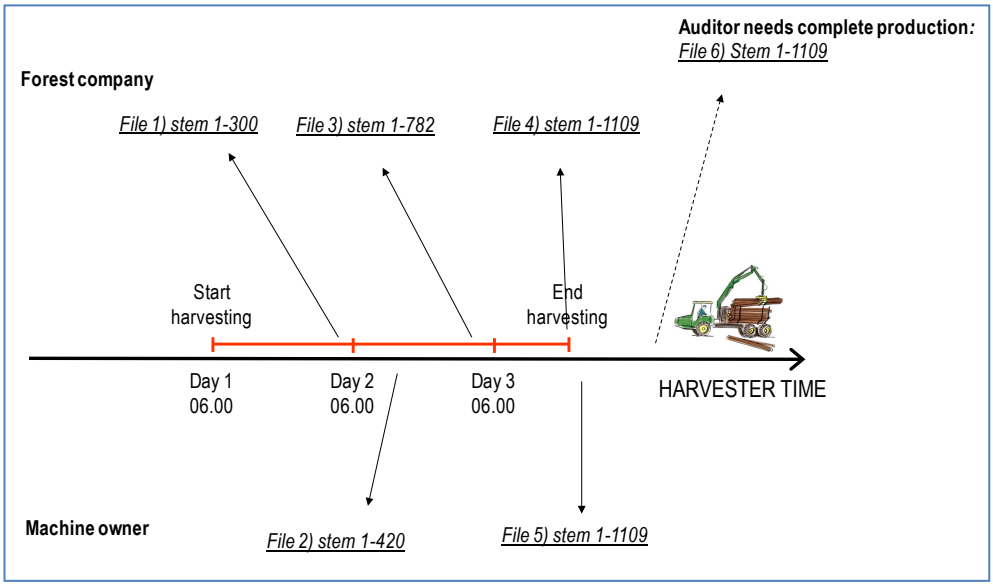


Figure 1. Illustration of StanForD requirement that it must be possible to report total production. In this case sending data to three separate organisations.

It was decided at the meeting in June 2010 that a total reporting alternative must be implemented by all manufacturers. The requirement is that it must be possible to generate a complete hpr-file including all stems harvested at the object since the start of harvesting. Partial reporting can be used if manufacturer finds it suitable (see also section concerning Skogforsk recommendation).

Example of four hpr-files sent from one harvesting object:

Machine-Key	Obj-Key	Stem-Count	1st stem	Last stem	HarvestDates	Creation-Date
3XE2_Pink	126	119	1	119	2010-02-03 – 04	2010-02-04
3XE2_Pink	126	343	1	343	2010-02-03 – 06	2010-02-06
3XE2_Pink	126	519	1	519	2010-02-03 – 08	2010-02-08
3XE2_Pink	126	1021	1	1021	2010-02-03 – 11	2010-02-11
3XE2_Pink	126	1411	1	1411	2010-02-03 – 14	2010-02-14

It is simple to check the total number of stems as well as the first and the last stem number in each file. The first StemNumber must be equal to 1 and the last StemNumber must be equal to the total number of stems in the whole complete hpr-file. Based on this information we can easily identify that this is a complete hpr-file including all harvested stems.

Observe that this control can be done by third party reporting software (for example SDC Sender) in the harvester. It is possible to develop a reporting software that controls whether stems has already been sent and if that is the case excludes them from the hpr-file that is sent to the administrative system. It would also make it possible to have softwares that works similar to an existing SCA software that extracts some limited information that is sent.

Note that processing of large xml-files requires significant computing resources. Processing (eg excluding duplicate stems) in harvester can be problematic since the hpr-file might be sent from a different location if communication from harvester does not work (SDC Sender).

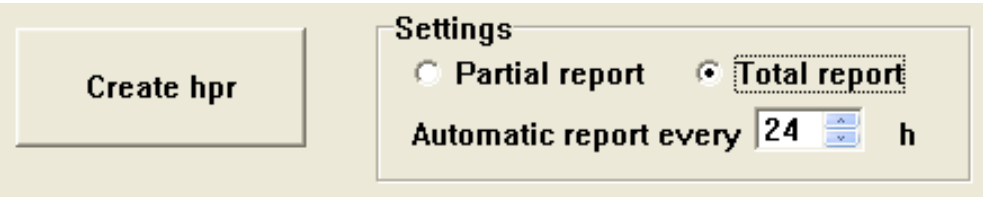
An advantage with this way of reporting is that it does not really matter if one report is lost during harvesting at an object!

Operator interaction during harvesting

Reporting of production and monitoring data must be simple:

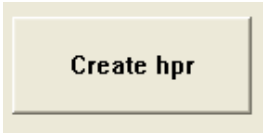
- Always possible to create a complete hpr-report including all previously reported stems (default alternative)
- Partial reporting possible to implement (no strict rules exist)

The figure below illustrates two buttons for generating hpr-messages and possible setting for automatic generation of hpr-messages:



No BatchNumber is registered.

It must always be possible to generate a complete total hpr-report:



Click 1 → Stem 1-300 (1st hpr)
Click 2 → Stem 1-421 (2nd hpr)
Click 3 → Stem 1-783 (3rd hpr)

It could be possible to report only stems processed since last reporting when generating a new partial hpr-message:

Click 1 → Stem 1-300 (1st hpr)
Click 2 → Stem 301-421 (2nd hpr)
Click 3 → Stem 422-783 (3rd hpr)

It is possible to create partial reports per stem number, per time, per operator, per file size etc.

No specific solution for retrieving missing data is needed if only complete hpr-files are reported.

Retrieving missing data from harvester

A crucial issue is how to retrieve data from a specific harvester if the receiving system has found that data is missing.

If aggregated files are allowed and all manufacturers are required to make it possible to generate a complete hpr-file as described above we only need to tell the operator to create a new complete hpr-file (illustrated in previous section).

Partial reporting (Skogforsk recommendation)

As stated above the standard only mandates that it must be possible to generate total hpr. However this may end up being a problem in the harvesters due primarily to the size of the hpr-files. The following recommendation has therefore been written by Skogforsk in order to describe a solution for partial reporting that will probably be acceptable to the Swedish market.

The Skogforsk recommendation is based on the preliminary reporting rules, as defined in section *Introduction*, which mean that stems sent more than once always are to be sent in identical messages (batches/intervals). An example could be that four daily reports are sent to a forest company during harvesting of an object. It is then discovered by an auditor that a complete set of production data is needed, the strict interpretation of only using partial reporting would mean that the same four hpr-files are to be generated again. A slightly more complex example of a strict interpretation is illustrated in figure 2 below where data is sent to both forest company, machine owner as well as to an auditor.

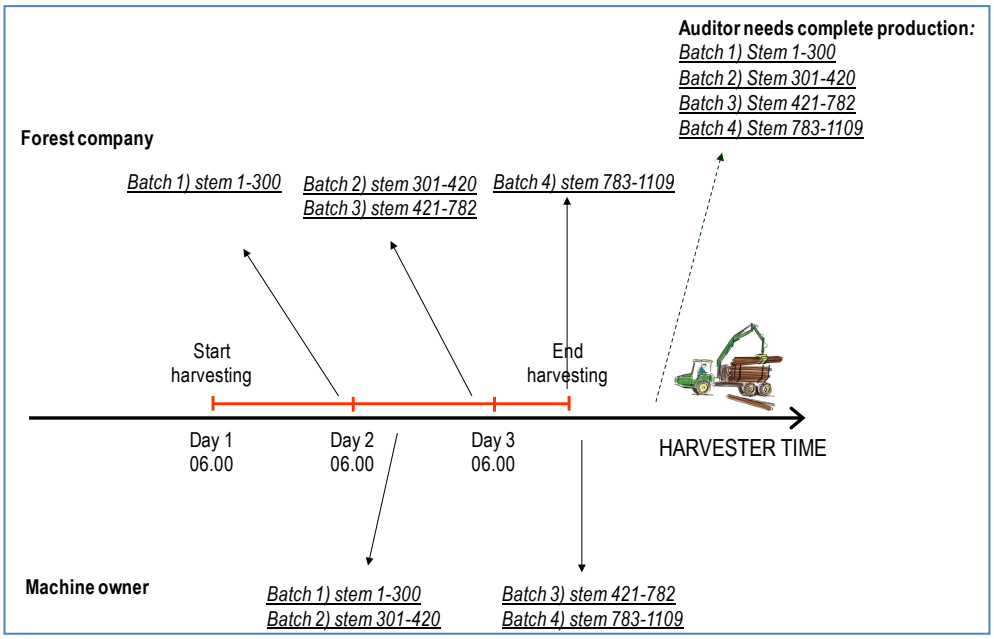


Figure 2. Illustration of a strict interpretation where production data is sent to three different organisations.

As described earlier it will probably always be necessary to check the individual stem numbers to find out if some stems are missing. It is in other words logical to implement a function checking of missing individual in all systems receiving hpr files.

Example of four hpr-files sent from one harvesting object:

Machine-Key	Obj-Key	Stem-Count	1st stem	Last stem	HarvestDates	Creation-Date
3XE2_Pink	126	119	1	119	2010-02-03 – 04	2010-02-04
3XE2_Pink	126	223	120	343	2010-02-05 – 06	2010-02-06
3XE2_Pink	126	175	344	519	2010-02-07 – 08	2010-02-08
3XE2_Pink	126	301	720	1021	2010-02-09 – 11	2010-02-11
3XE2_Pink	126	238	1173	1411	2010-02-12 – 14	2010-02-14

It is simple to check the total number of stems as well as the first and the last stem number in each file. Based on this information we can easily identify which stems are missing, in this example 520-719 and 1022-1175.

Operator interaction during harvesting

Reporting of production and monitoring data must be simple:

- Possible to re-create old hpr-reports
- An idea might be a setting for daily automatic reporting

The figure below illustrates functions for generating hpr-messages and a setting for a possible automatic generation of hpr-messages:

No BatchNumber is registered, only stems processed since last reporting are included when generating a new hpr-message:

Click 1 → Stem 1-300 (1st hpr)
Click 2 → Stem 301-421 (2nd hpr)
Click 3 → Stem 422-783 (3rd hpr)

It is suggested that the operator easily can view a log of hpr-reports alreade sent. A possible solution when re-creating old hpr-messages is to give the operator the possibility select whether he wants to generate all previously sent reports or only some specific reports including the missing stems as illustrated in the figure below:

Retrieving missing data from harvester

A crucial issue is how to retrieve data from a specific harvester if the receiving system has found that data is missing.

If we decide to exclude BatchNumber we have to tell the operator what stem numbers are missing and when these where probably sent. This could be done by phone or some free text message like e-mail or sms.

Note that it must be possible to re-generate the same partial reports that were previously sent.

Example recommended partial reporting:

1. *Stems 1-300 sent from harvester on 2010-02-04*
2. *Stems 301-421 sent from harvester on 2010-02-06*
3. *Stems 422-783 sent from harvester on 2010-02-08*
4. *Object finished and stems 784-1201 sent from harvester on 2010-02-11*
5. *Forest company (receiver of data) discovers that stems 422-783 are missing.*
6. *An sms or e-mail is sent to harvester indicating that stems 422-783 are missing and that this production was sent between 2010-02-06 and 2010-02-11.*

7. *Operator easily views a log of all partial reports, including stem numbers and date for each report.*

8. *Operator can either select only the one single file missing or if he is a bit uncertain select all partial reports and resend.*

Other issues

Batch number

Including an element BatchNumber has been discussed at several meetings during 2010. An important argument against "BatchNumber" is that they do not guarantee that the information of every stem is sent and received. It will probably be necessary to check the individual stem numbers to find out if some stems are missing. It is in other words logical to implement a function checking of missing individual in all systems receiving hpr files. This means that BatchNumber will not give any additional value, just making the whole standard slightly more complex. It is for example easier to make certain that the element StemNumber is implemented according to the standard than both StemNumber and BatchNumber.

Another problem with BatchNumber is to define how to use it in cases where you merge several hpr-files from one object into one single file in an administrative system or if data from separate objects are included in one hpr-file. StemNumber must always be constant for a specific stem which means that the cases described above does not matter at all.

It is not unreasonable to expect on-line reporting within a decade where each stem is sent individually (directly after harvesting). BatchNumber will then be identical with StemNumber and thus of no use in such a scenario.

An advantage with using BatchNumber is that is a bit simpler to use than stem numbers. It would also make it very simple to give feedback to operators. It would be quite simple to tell the operator by phone to tell him that a certain BatchNumber is missing.

LOCATION OF MESSAGES GENERATED IN MACHINE

The possibility of including rules specifying fixed folders for messages generated in the machine has been discussed and rejected. This proposal meant that a specific folder location and name should be used by all manufacturers.

Since the possible ways of communicating with forest machines varies dramatically between different markets there is no way StanForD can include how the forest machine is to communicate with the outside world. This means that the issue of where files are located in the machine is still an important issue for all those organizations wanting to develop a transmission tool of their own.

Another solution for this issue is that:

- All StanForD 2010 files always located in one single folder ("outbox")
- This folder can be located anywhere and have any name

This means that a third party application only needs to "find" one single location that is constant throughout the lifetime of the computer.

Another related issue is whether we need some kind of "inbox". An example might be if we create "automatic" systems for receiving and updating ProductDefinitionsand SpeciesGroupDefinitions using *pin*- and *spi*-messages, another example is a new *MissingHarvestedProduction* file.

It was decided that we have a recommendation that all manufacturers should implement a static folder (location) where as default StanForD 2010 files are to be saved. It is acceptable to have separate static folders for different types of messages.

John Arlinger
Lars Henriksson

Appendix 2) Mapping of StanForD Id elements

It was decided during the development of StanForD 2010 (2010-11-10) that John Arlinger was to write a proposal for how to map and convert between the two standard versions with a focus on relevant identities.

Conversions may according to the manufacturers occur in two directions:

- StanForD 2010 →StanForD
  For example, new StanForD 2010 instructions (oin/pin/spi) are sent to harvester and old drf-file is reported from machine.

- StanForD →StanForD 2010
  For example, old StanForD apt-file is sent to harvester and new hpr-file reported from harvester.

BACKGROUND

Three major issues concerning mapping was discussed at the telephone meeting 2010-12-10:

- Is a one-to-one relationship really needed? Do we need a couple of new elements?

- Why is not UserId elements included in the mapping table?

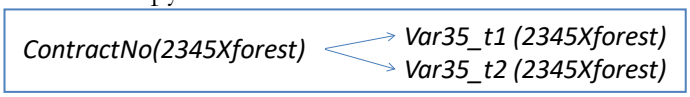
- Is apt-file to be converted into oin/pin/spi before being used in a harvester adopted to StanForD 2010?

ONE-TO-ONE RELATIONSHIP

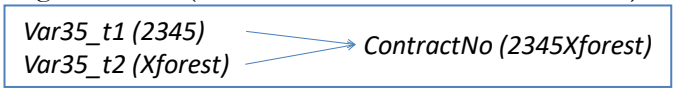
Conversions will become significantly more complex if we do not have a one-to-one relationship between StanForD variables and StanForD 2010 elements.

A simple example of this problem (having a one-to-many relationship) can be illustrated by comparing variable 35 (contract number) and element ContractNo. Var35 consists of two types with no clear difference which means that we may have a on-to-two relationship.

When StanForD 2010 structure is converted to StanForD (for ex. oin → drf) we could copy the same data to both variables:

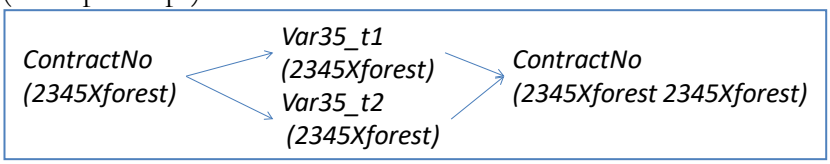


When a StanForD structure is converted to StanForD 2010 (for ex. apt → hpr) a solution could be that we merge the data for these two variables into a single element ("2345"+"Xforest"="2345Xforest"):

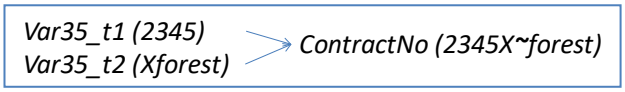


One problem with this solution is that we do not know whether the whole string came from one or both (var35_t1 and var35_t2) StanForD variables.

Another problem with one-to-many relationships is that we may end up with a mess if the conversion is done more than once as illustrated below (oin→pri→hpr):



One way to handle this kind of one-to-many relationship could be to use a separation character like ~ between the different StanForD variables included in one element as illustrated below:



However this would mean that we must have a separate parser for certain elements when they are based on converted structures.

Consequently Skogforsk do prefer staying with some kind of strict one-to-one relationship.

USERIDS

The UserIDs have a dual purpose in StanForD 2010 which does not exist in StanForD:

- Forest companies or other users of data use *UserId* for identification purposes. *UserIds* may be used for harvesting objects, products, tree species, operators, locations, deliveries and machines.
- UserIds are also used to link for example products in oin and pin together which means that they in some cases (for example oin) must be unique. This purpose is further described below.

The basic idea is that only one instance of a specific *UserId* should be available to use in a machine at a time. An old product or species group is for example to be replaced if a new product or species group with the same *UserId* is received by a machine. This means that for example only one instance of *ProductUserId* can be selected and used at a given time

The following figures (1-2) illustrates an example of how *UserIds*, *ModificationDates* and *Keys* are handled when updating and using object instructions. Observe that both *UserId* and *ModificationDate* are to be considered when updating the list of objects. An existing object is not to be replaced by a new object sent to the harvester if they have identical *UserId* and *ModificationDate*. The operator always must have the possibility to deny the replacement of an existing object as well as an update of the active object presently in use.

Step 1) New objects sent to harvester

Name	UserId	ModificationDate	
BlueForest	456	2008-11-22 23:30	Exists!
BlackForest	567	2008-11-22 09:01	Exists!
YellowForest	678	2008-11-22 09:19	New!
BlueForest	111	2008-11-22 23:30	New!

Original list of objects in harvester

Name	UserId	ModificationDate	Key
RedForest	234	08-11-17 23:30	1
GreenForest	345	08-11-17 23:30	2
BlueForest	456	08-11-17 23:30	3
BlackForest	567	08-11-17 23:30	4

Figure 1. Five new objects (oin) are sent to a harvester. The UserId of the first (456) and the second product (567) already exists among the object while the third and forth objects are new.

Step 1) New objects sent to harvester

Name	UserId	ModificationDate
BlueForest	456	2008-11-22 23:30
BlackForest	567	2008-11-22 09:01
YellowForest	678	2008-11-22 09:19
BlueForest	111	2008-11-22 23:30

New list of objects in harvester

Name	UserId	ModificationDate	Key	
RedForest	234	08-11-17 23:30	1	
GreenForest	345	08-11-17 23:30	2	
BlueForest	456	08-11-22 23:30	3	Update active object definition in use
BlackForest	567	08-11-22 09:19	4	Replaced by new "BlackForest"
YellowForest	678	08-11-22 09:19	5	
BlueForest	111	08-11-22 23:30	6	New objects, added

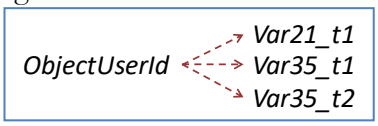
Figure 2. The list of objects is updated. Observe that a second object named BlueForest now exist since the ObjectUserIds are different .

Unique identities in the old StanForD are today registered in different variables depending on market and companies involved. Sometimes a single variable is used sometimes several variables are used together. The following examples illustrate the fact that this is quite complex today:

- The unique id of a harvesting object (same as ObjectUserId) is often stored in var21_t1, var35_t1 or var35_t2.
  Södra skogsägarna use a combination of var21_t1 and var35_t1
  Sveaskog use var35_t2 (most common way in Sweden)
  Finland (as far as I have understood) use var35_t1

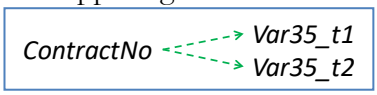
- ProductUserId stored in var32_t2, var121_t2, var121_t3 or var121_t6 in present StanForD

The following figure illustrates that ObjectUserId would have to be mapped against a number of different StanForD variables (example oin → drf):



Observe that we would have to have different rules for mapping ObjectUserId on different markets and for different forest companies.

To make the situation even more complex there are other elements that would be mapped against the same variables as for example ContractNo:



A rule would be needed to define whether ObjectUserId should over ride ContractNo or not.

Suggested solution

After some discussions (phone meeting 2011-01-21 and 2012-01-31) it was decided that we make the exact mapping settable in the machine according to the following table:

Elements	Alternative mapping variables set in machine
MachineOwnerID	Var3_t1
MachineUserId	Var3_t2
ObjectUserId	Var21_t1 OR var35_t1 OR var35_t2
ContractNumber	Var35_t1 OR var35_t2 Or no mapping
SubObjectUserId	Var21_t1 Or Var21_t2 Or no mapping
RealEstateIDObject	Var21_t1 OR var35_t1 OR var35_t2 Or no mapping
ProductUserId	Var120_t1 AND Var121_t1 AND Var121_t2 AND var121_t3 AND var121_t5
ProductVersion	Var121_t2 OR var121_t3 OR var121_t4 OR var121_t5 OR no mapping
ProductInfo	Var121_t2 OR var121_t3 OR var121_t4 OR var121_t5 OR no mapping
ProductDestination.BusinessID	var32_t2
SpeciesGroupUserId	var120_t1 AND var120_t3
SpeciesGroupName	var120_t1 (var120_t2 in case of stm/ktr)
SpeciesGroupInfo	Order of species in file (eg stem code 1, var266 in pri-file)
SpeciesGroupVersion	var120_t3

Red and underlined means default mapping in Sweden. Necessary to follow if file is sent to SDC

Observe that ProductUserId and SpeciesGroupUserId are never to be converted from StanForD 2010 to StanForD (for example from hpr to prd).

In this case it is not necessary to add any new id elements.

This solution may in theory give as a problem if for example var21_t1, var35_t1 or var35_t2 are empty (no ObjectUserId created) or if duplicate UserIds (for example ProductUserId) are created within a message. The following rule must be implemented when doing conversions:

- Conversion of apt to oin/pin/spi not allowed if UserIds in file are not unique (for example ProducUserId).

Converting apt-file to StanForD 2010

Is an apt-file to be converted into oin/pin/spi before being used in a StanForD 2010-harvester or not? How is it going to be used if we generate StanForD 2010 files?

It is suggested that an apt-file is normally not converted to oin/pin/spi before being imported to "product database"/"bucking application" in harvesters adapted to StanForD 2010. All "Keys" are then set when apt is imported and products created in product database.

Below is an example based on the suggestions above (mapping set in machine):

Apt →

Var	Data
v21_t1	XForest
v35_t1	123417
v35_t2	xY
v120_t1	Pine
V121_t1	P_Saw
v121_t2	11_Q1
v121_t3	Mill_1Y_ Special
v32_t2	Mill_1Y

Hpr →

Element	Data
ObjUserId	XForest
RealEstateIdObject	123417
ContractNo	xY
SpeciesGroupName	Pine
ProductName	P_Saw
ProductInfo	11_Q1
ProductVersion	Mill_1Y_special
ProdDest. BusinessID	Mill_1Y
ProdUserId	PineP_Saw 11_Q1 Mill_1Y_Special
ObjectKey	1
ProdKey	1

Pri

Var	Data
v21_t1	XForest
v35_t1	123417
v35_t2	xY
v120_t1	Pine
V121_t1	P_Saw
v121_t2	11_Q1
v121_t3	Mill_1Y_ Special
v32_t2	Mill_1Y

This alternative means that a pin-product is not to be used for updating apt-products!

Mapping table

The following table describes a suggested mapping for important id variables:

Structure (Def)	New element	Old variable
MachineDefinition	MachineIdOwner	Var003_t1
MachineDefinition	MachineUserId	var003_t2
MachineDefinition	MachineCategory	Var003_t3
MachineDefinition	MachienBaseManufacturer	Var003_t5
MachineDefinition	MachienBaseModel	Var003_t6
MachineDefinition	MachienHeadManufacturer	Var003_t7
MachineDefinition	MachienHeadModel	Var003_t8
MessageHeader	ApplicationVersionCreation	var005_t1
MessageHeader	ApplicationVersionCreation	var005_t2
MessageHeader	ApplicationVersionModification	var005_t4 (caliper id in ktr)
MessageHeader	CountryCode	Var006_t1
MessageHeader	CreationDate	var012_t4
ProductDefinition	ProductCreationDate	Var013_t4
ObjectDefinition	StartDate	Var016_t4

Structure (Def)	New element	Old variable
ObjectDefinition	EndDate	Var017_t4
ObjectDefinition	ObjectUserId	Var021_t1 OR var35_t1 OR var35_t2
ObjectDefinition	RealEstateIdObject	Var021_t1 OR var35_t1 OR var35_t2 Or no mapping
SubObjectDef	SubObjectName	Var21_t2 OR Var21_t3 OR Var21_t4 Or no mapping
SubObjectDef	SubObjectUserId	Var21_t2 OR Var21_t3 OR Var21_t4 Or no mapping
SubObjectDef	RealEstateIdSubObject	Var21_t2 OR Var21_t3 OR Var21_t4 Or no mapping
ObjectDefinition	ForestCertification	Var021_t5
ObjectDefinition	LoggingFormCode	Var023_t1
ObjectDefinition	LoggingFormDescription	Var023_t2
ObjectDefinition	ObjectArea	Var023_t3
ObjectDefinition	LoggingOrganisation.BusinessId	var031_t1
ObjectDefinition	LoggingOrganisation.BusinessName	var031_t2
ObjectDefinition	LoggingOrganisation.**District**	var031_t3
ObjectDefinition	LoggingOrganisation.**Team**	var031_t4
ObjectDefinition	LoggingOrganization.LastName	var031_t6
ObjectDefinition	LoggingOrganization.Address	var031_t7
ObjectDefinition	LoggingOrganization.Email	var031_t8
ObjectDefinition	LoggingOrganization.Phone	var031_t9
ProductDefinition	ProductBuyer.BusinessID	var032_t1
ProductDefinition	ProductDestination.BusinessID	var032_t2
ObjectDefinition	ForestOwner.BusinessName	Var033_t1
ObjectDefinition	ForestOwner.BusinessID	Var033_t2
ObjectDefinition	ForestOwner.Name	Var033_t3
ObjectDefinition	ForestOwner.Adress	Var033_t4
ObjectDefinition	ForestOwner.Email	Var033_t5
ObjectDefinition	ForestOwner.Phone	Var033_t6
MachineDefinition	MachineOwner.BussinesName	Var034_t1
MachineDefinition	LoggingContractor.BussinesName	Var034_t1
MachineDefinition	MachineOwner.BussinesID	Var034_t2
MachineDefinition	LoggingContractor.BussinesID	Var034_t2
MachineDefinition	MachineOwner.LastName	Var034_t3
MachineDefinition	LoggingContractor.LastName	Var034_t3
MachineDefinition	LoggingContractor.Adress	Var034_t4
MachineDefinition	LoggingContractor.Email	Var034_t5
MachineDefinition	LoggingContractor.Phone	Var034_t6
ObjectDefinition	ContractNo	Var35_t1 OR var35_t2 Or no mapping
Stem	SpeciesGroupKey	var110_t0 (stm/ktr)
SpeciesGroupDefinition	SpeciesGroupUserId	var120_t1 AND var120_t3
SpeciesGroupDefinition	SpeciesGroupName	var120_t1
SpeciesGroupDefinition	SpeciesGroupName	var120_t2 (stm/ktr)
SpeciesGroupDefinition	SpeciesGroupVersion	var120_t3
SpeciesGroupDefinition	SpeciesGroupInfo	Order of species in file (eg stem code 1, var266 in pri-file))
ProductDefinition	ProductName	Var121_t1
ProductDefinition	ProductUserId	Var120_t1 AND Var121_t1 AND Var121_t2 AND var121_t3 AND var121_t5
ProductDefinition	ProductVersion	Var121_t2 OR var121_t3 OR var121_t4 OR var121_t5 OR no mapping

Structure (Def)	New element	Old variable
ProductDefinition	ProductInfo	Var121_t2 OR var121_t3 OR var121_t4 OR var121_t5 OR no mapping
ProductDefinition	ProductCreationDate.	Var12_t4
ProductDefinition	ProductGroupName	var127_t1
DiameterDef	DiameterClassLowerLimit	var131_t1
DiameterDef	DiameterClassName	var131_t2
SpeciesGroupDefinition	GradeName	var143_t1
ObjectDefinition	ObjectTextToMachine	Var200_t2
OperatorDefinition	OperatorUserId	Var212_t1
DeliveryDefinition	DeliveryInfo	Var441_t2

Red and underlined means default mapping in Sweden. Necessary to follow if file is sent to SDC

Swedish Translations

Below is a table with Skogforsk and SDC recommendations regarding Swedish translations of certain important elements.

Structure (Def)	New element	Swedish GUI name	Comments
MachineDefinition	MachineKey	Maskinnyckel	
MachineDefinition	MachineOwnerID	Ägarens maskin id	Id defined by machine owner
MachineDefinition	MachineUserId	Maskin id enl	Id defined by logging organisation
MachineDefinition	MachineCategory	Maskintyp	
MachineDefinition	MachienBase-Manufacturer	Basmaskin, märke	
MachineDefinition	MachienBase-Model	Basmaskin, modell	
MachineDefinition	MachienHead-Manufacturer	Aggregat, märke	
MachineDefinition	MachienHeadModel	Aggregat, modell	
MessageHeader	ApplicationVersion-Creation	Version när skapad???	
MessageHeader	ApplicationVersion-Modification	Version när ändrad	
MessageHeader	CountryCode	Landskod	
MessageHeader	CreationDate	Skapad:	
ProductDefinition	ProductCreationDate	Ändrad:	
ObjectDefinition	StartDate	Startdatum	
ObjectDefinition	EndDate	Slutdatum	
ObjectDefinition	ObjectName	Objektnamn	
ObjectDefinition	ObjectUserId	Objekt id	Id defined by logging organization
ObjectDefnition	RealEstateIdObject	Objektgrupp-ID	
SubObjectDef	SubObjectName	Del-objektnamn	
SubObjectDef	SubObjectUserId	Del-objekt id	Id defined by logging organisation
SubObjectDef	RealEstateIdSubObject	Del-objekt, Fastighets nr	
ObjectDefinition	ForestCertification	Skogs-certifiering	
ObjectDefinition	LoggingFormCode	Avverkningstyp	
ObjectDefinition	LoggingFormDescr.	Avverkningstyp, beskrivning	
ObjectDefinition	ObjectArea	Areal objekt	
ObjectDefinition	LoggingOrganisation. BusinessId	Organisation/skogsföretag, Id företag	
ObjectDefinition	LoggingOrganisation. BusinessName	Organisation/skogsföretag, Företagsnamn	
ObjectDefinition	LoggingOrganisation. **District**	Organisation/skogsföretag, Region	
ObjectDefinition	LoggingOrganisation. **Team**	Organisation/skogsföretag, Distrikt	
ObjectDefinition	LoggingOrganization. LastName	Organisation/skogsföretag, Efternamn	
ObjectDefinition	LoggingOrganization. Address	Organisation/skogsföretag, Adress	
ObjectDefinition	LoggingOrganization. Email	Organisation/skogsföretag, E-post /Mäjl	
ObjectDefinition	LoggingOrganization. Phone	Organisation/skogsföretag, Tele	

Structure (Def)	New element	Swedish GUI name	Comments
ProductDefinition	ProductBuyer. BusinessID	Köpare, Kod	Used by SDC
ProductDefinition	ProductDestination. BusinessID	Mottagningsplats, Kod	Used by SDC
ObjectDefinition	ForestOwner. BusinessName	Skogsägare, Företagsnamn	
ObjectDefinition	ForestOwner. BusinessID	Skogsägare, Kod	
MachineDefinition	MachineOwner. BussinesName	Maskinägare, företagsnamn	
MachineDefinition	LoggingContractor. BussinesName	Skogsentreprenör, företagsnamn	
MachineDefinition	MachineOwner. BussinesID	Maskinägare, Företagskod	
MachineDefinition	LoggingContractor. BussinesID	Skogsentreprenör, Företagskod	
OperatorDefinition	BusinessName	Arbetsgivare	
OperatorDefinition	BusinessID	Anställnings ID/Nr	
ObjectDefinition	ContractNo	~~Kontraktsnr~~ Virkesordernr	Today normally virkesordernummer in Sweden, used by SDC.
ObjectDefinition	ContractCategory	Kontraktstyp	
SpeciesGroupDefinition	SpeciesGroupUserId	Trädslag id	Id defined by logging organisation
SpeciesGroupDefinition	SpeciesGroupName	Trädslag, namn	
SpeciesGroupDefinition	SpeciesGroupVersion	Trädslag, version	
SpeciesGroupDefinition	SpeciesGroupInfo	Trädslagskod	
ProductDefinition	ProductName	Produktnamn	
ProductDefinition	ProductUserId	Produkt id	Id defined by logging organisation
ProductDefinition	ProductVersion	Produktversion	Used by SDC.
ProductDefinition	ProductInfo	Produktkod	Used by SDC. Today normally assortment code (SSTE)
ProductDefinition	ProductCreationDate.		
ProductDefinition	ProductGroupName	Produktgruppsnamn	
OperatorDefinition	OperatorUserId	Förar id	Id defined by machine owner
DeliveryDefinition	Delivery	Leveransenhet or Leverans	
DeliveryDefinition	DeliveryInfo	Leveranskod	Used by SDC. Today normally assortment code (SSTE)
DeliveryDefinition	DeliveryDestination	Destinering	Used by SDC.
LocationDefinition	Location	Avlastningsläge or Läge	
LocationDefinition	LocationInfo	Lägeskod	Will be used by SDC.