

FlexWood

Flexible Wood Supply Chain

Del. no. 7.2
WP no. 7000
Date 13/10/2012
Version 1.0
Revision
Confidentiality PU

Design of the Overall Architecture



Responsible Partner University College Cork (UCC.)
Authors James Little, Oscar Manzano (UCC)
Project Coordinator University of Freiburg FeLis (ALU-FR)



The FlexWood project is funded by the European Commission within the Seventh Framework Programme (FP7). The Collaborative Project (small or medium sized focused research project) contributes to "Meeting industrial requirements on wood raw-materials quality and quantity" activities.

FP7 GRANT AGREEMENT No. 245136



Coordination:



FeLis – Department of Remote Sensing and Landscape Information Systems
University of Freiburg
Tennenbacherstr. 4, D-79106 Freiburg, Germany
www.felis.uni-freiburg.de



Contact:
Prof. Dr. Barbara Koch: barbara.koch@felis.uni-freiburg.de
Alicia Unrau: alicia.unrau@felis.uni-freiburg.de

Partners:

Albert Ludwigs Universität Freiburg	ALU-FR	Germany
Stiftelsen Skogsbrukets Forskningsinstitute	Skogforsk	Sweden
FCBA Institut Technologique	FCBA	France
University of Eastern Finland	UEF	Finland
Technical Research Centre of Finland	VTT	Finland
University College Cork	UCC	Ireland
TreeMetrics Ltd	TreeMetrics	Ireland
Forest Research Institute of Baden-Württemberg	FVA	Germany
Norwegian University of Life Sciences	UMB	Norway
University of Natural Resources and Applied Life Sciences	BOKU	Austria
Polish Research Institute	IBL	Poland
Logica Ltd	Logica	Sweden
Foran Ltd	FORAN RS	Sweden
University of Laval	UL	Canada



Website: www.flexwood-eu.org

Table of Contents

1. Executive Summary	4
2. Introduction	5
3. Objectives of the Deliverable	5
4. Description of the Architecture.....	6
4.1 Web Services	7
4.2 User Interface.....	8
4.3 Harvesting & Sawmill	14
5. Technology description	19
5.1 Client side	19
5.2 Server Side.....	20
6. Stakeholders	23
7. Appendices	23
Appendix A – Stem File XML.....	23
Appendix B – Log Breakout File XML (for WoodCIM)	25
Appendix C – Sawmill Products File XML.....	25

1. Executive Summary

The deliverable on the 'Design of the Overall Architecture' describes first its objectives to the project, secondly how they are implemented and finally the end result in terms of user experience. The main objective of this work package is to support the delivery of the partners' research results and to provide an effective medium to engage with stakeholder in the industry to evaluate and use the system. The implementation of the architecture uses industry standard techniques for a distributed application such as web services hosted at each relevant partner.

2. Introduction

The role of the FlexWood architecture is to provide the IT infrastructure for collaboration of several forest related services along the supply chain. This collaboration is a key to the success of the FlexWood project as it allows a complete integrated model of the forest supply chain to be achieved. Only by bringing the various services together can there be a synergistic effect, to realise many new types of knowledge which can be determined before, during and after harvesting. The other important aspect of the architecture is the multitude of potential stakeholders in different locations capable of using the system. The architecture therefore supports multiple views of the operations and data from a wide variety of devices. Therefore, only appropriate functions and results are available for each stakeholder.

In these types of collaborative projects, the architecture also provides the mechanism for integration between partners (work packages). An appropriate architecture can assist the partners in sharing and deploying their research software and at the same time not to establish many bilateral discussions which necessarily splits the focus of the project. For the above reasons we have adopted a distributed web service based architecture with centralised controller. Therefore each partner's software is packaged as a web service with strict published input and output formats through xml files. The web service is hosted on the partner's web server and can be independently maintained and developed. The cueing in of each of these services is determined through the FlexWood portal user interface and its controller service. This function decides the appropriate service(s) to invoke (with data). The open xml format also allows other web services to be plugged in, where national or regional ways of operating need to be respected. It also allows alternative web services to be provisioned for wider commercial exploitation.

3. Objectives of the Deliverable

This deliverable seeks to justify and illustrate the choices made in implementing the IT infrastructure to support the FlexWood research project. The architecture also supports a prototype implemented system which can deliver knowledge to many types of users with varying set of requests.

4. Description of the Architecture

Figure 1 and Figure 2 show the various components making up the entire architecture of FlexWood. The implementation uses JavaScript, SOAP, Ajax and HTTP – all standard open software components for building a distributed, interactive web-based system. In Figure 1, the user at the top interacts over http with the FlexWood user interface and its controller to view information, build scenarios and execute various simulation services.

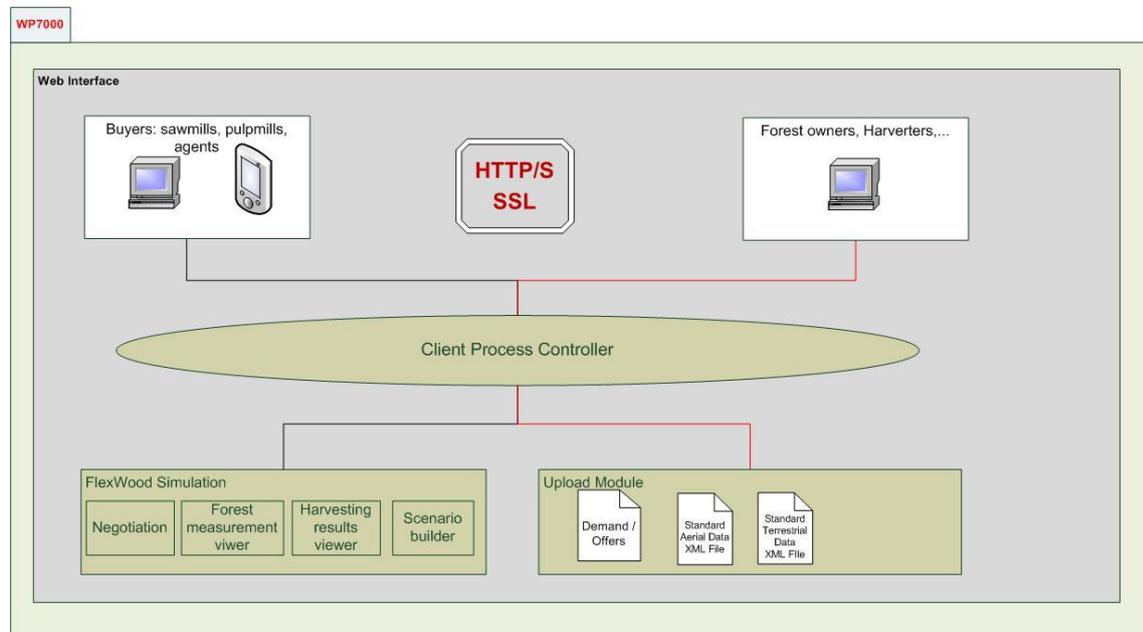


Figure 1: Process Controller Web Service and Interaction with the User Interface

In Figure 2, the different forestry services are broken down into types and responsibility of each work package. The controller web service is the interface between the user specifying some action through the UI and it being carried out. There are four main interfaces with other forest web services from the controller. The first is the measurement web service WP4000, which provides data and visualisations on forests, stands and stems. The second service is harvesting, which provides a simulation service for turning stem files into expected log output volume by length. This service also provides a catalogue of APT files (harvesting instructions) which the user can select, through the FlexWood interface, as input to the harvest simulation. These files describe the way that a stem is cut into logs and are therefore dependent on the user's desired demand. They are also highly national, since different tree species require different parameters. A third service concerns scheduling harvesting. This is an optimisation service which accepts as input the stands to be harvested, the resources to carry this out and location information. The service then returns a schedule to describe the sequence in which each stand is harvested and by which team. The fourth and final service is the sawmill simulation, which accepts a set of logs and applies a cross-cutting pattern to return a set of end products. The cross cutting patterns are accessed via the same web service, for users to make a selection.

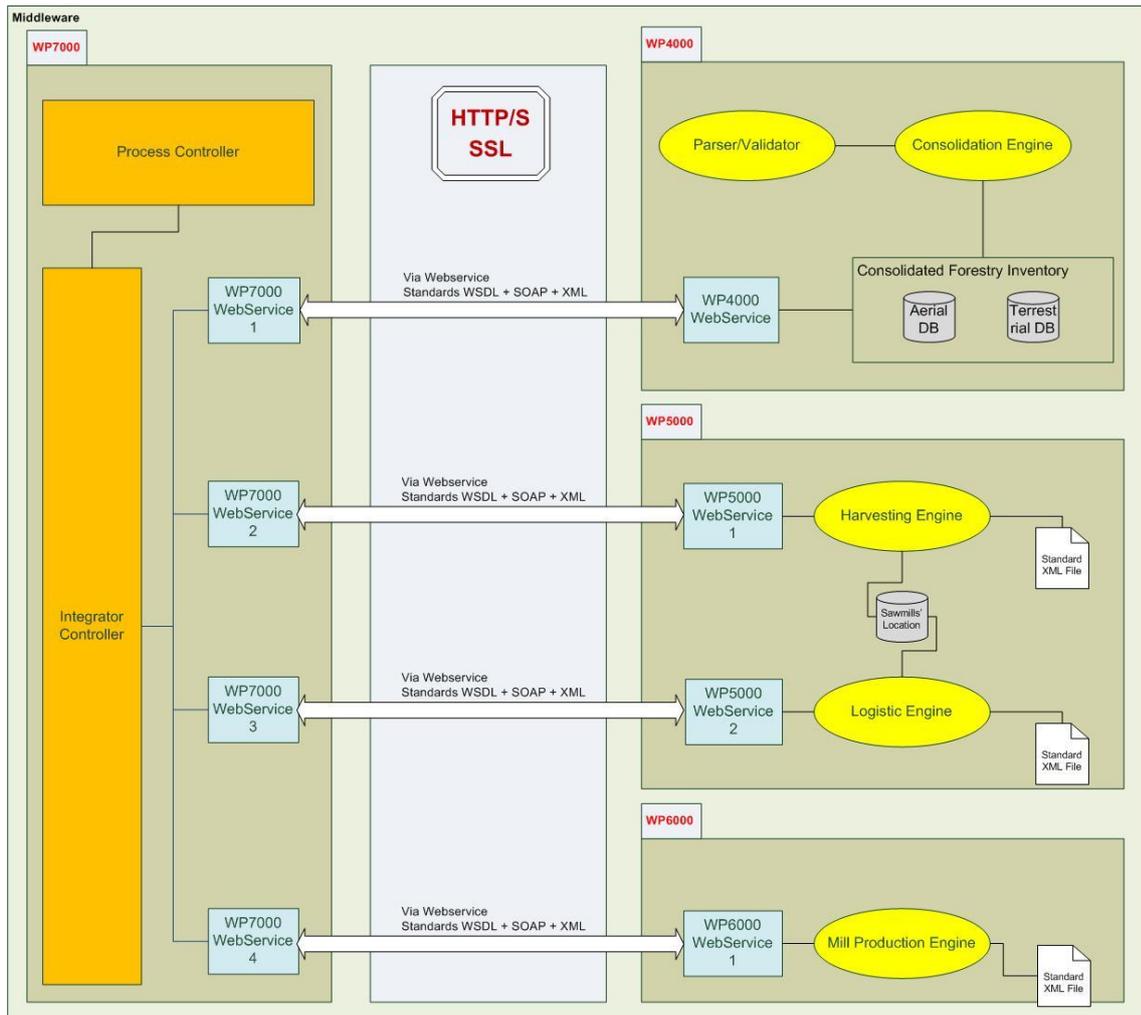


Figure 2 : Process Controller Web Service and Interaction with other Forest Web Services

4.1 Web Services

There are number of integrated web services making up the FlexWood system. Each can only be invoked by the stakeholder through the user interface. The main web services and descriptions are shown in Table 1.

Web Service	Hosted by	Description	URL
Controller	University College Cork, Ireland	This service takes all the user requests, checks for quality of input, create the input file and sends it to the appropriate web service. The controller service will also collect and output from the remote service and pass it to the user interface for display	http://4c110.ucc.ie:8080/FlexWoodPortal/Controller
User Interface	University College Cork, Ireland	This service allows the user to navigate, view and edit through the various aspects of the forest supply chain. It also allows the user to create various scenarios	http://4c110.ucc.ie/FlexWoodPortal/demos.jsp

		of typical operations they would like to carry out. Finally, the service provides output tables, pie charts and histograms of the various results of the services	
Measurement	Felis, University of Freiburg, Germany	This service provides all the measurement data on the forest. This includes stand and stem properties, locations and routes.	http://flexwood.felis.uni-freiburg.de:8080/de/egree-wfs/services?service=WFS&version=1.1.0&request=DescribeFeatureType&typeName=flw:StandingStem&namespace=xmlns%28flw=http://flexwood.uni-freiburg.de/flw%29&outputformat=text%2Fxml%3B+subtype%3Dgml%2F3.1.1
Harvesting	Skogforsk, Sweden	The harvesting web service takes a description of a stand or stem and then simulates harvesting it according to demand and/or selected APT file. The results are a breakout of log and pulp products	http://fw.skogforsk.se/BuckingSim/SkogforskBuckingSim_CGI.exe/RunBuckingSimulation?AptMessageName=TestSwe.apr&StmMessageName=TestSwe.stm
Logistics/Scheduling	Skogforsk, Sweden	This service determines the best schedule for harvesting given the stands, their location and the resources (labour and machines) which are available. The results are a work schedule to harvest efficiently the stands and deliver the product to the sawmills	
Sawmill Simulation	VTT, Finland	The sawmill service simulates the cutting of a breakout of logs with a selected cutting pattern. This results in quantities of sawn products	http://130.188.4.177/flexwood/WCIM_OUTPUT.XML

Table 1 : Description of Existing Forest Web Services

4.2 User Interface

The user interface needs to support the stakeholders in the different case studies to carry out what they want to do. Part of the objective of the UI is to help the stakeholders reach a specification of their problem – we do this by presenting a number of forest entities and ways of combining them into scenarios to be passed to the appropriate forest web service. These entities are instantiated to actual forests, stands, sawmills, etc.

The initial login window (Figure 3) allows the user to specify what type of user they are. The user then arrives at two choices of Viewers and Scenarios. Taking the Viewer option (Figure 4) allows the user to inspect what is in the FlexWood database regarding uploaded measured forests across Europe. In the figure the button at the bottom goes to the web service at Freiburg and downloads the names and co-ordinates of all known forests.



©University College Cork

Figure 3 : Opening Screen of the FlexWood Portal

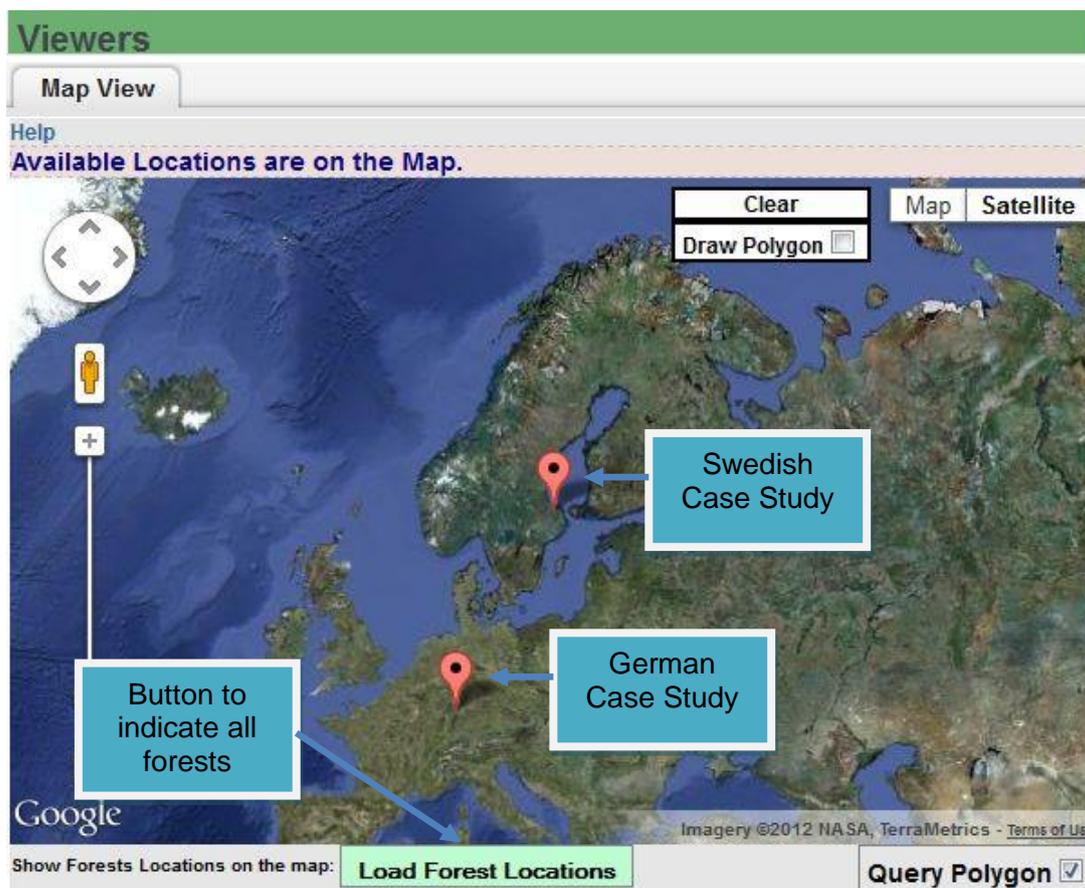


Figure 4 : Location of Forest Case Studies

Clicking on any of the forest markers will show basic information on location, number of trees and number of stands (Figure 5).

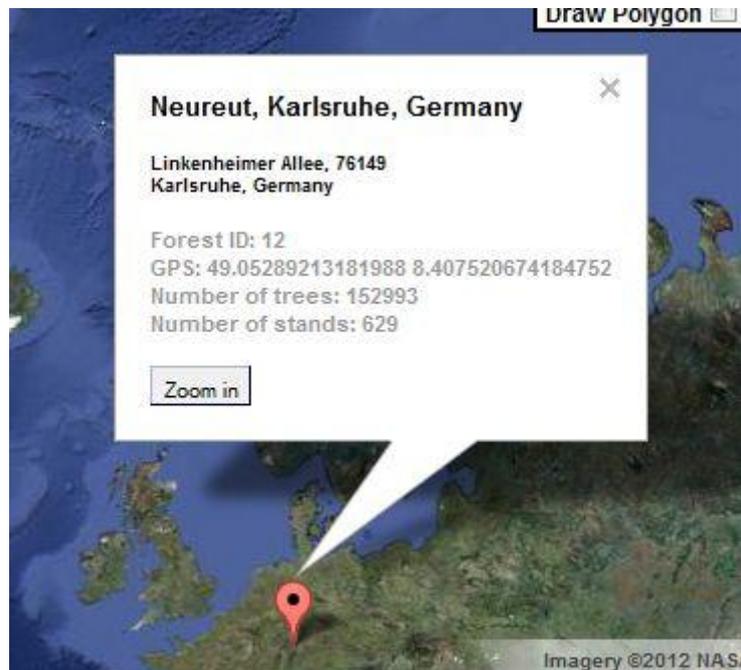


Figure 5 : Information on the Karlsruhe Forest, Germany

The user interface needs to support the stakeholders through the different phases of the forest supply chain and in the FlexWood case, to allow the case studies to be carried out. Part of the objective of the UI is to help the stakeholders reach a description of their problem – we do this by presenting a number of forest entities and ways of combining them into scenarios to be passed to the appropriate forest web service. These entities are instantiated to actual forests, stands, demands, sawmills, etc. Within each forest, it is broken down into stands and then stems, each of which has properties which can be inspected. Note that there can be many forests available from across Europe in the FlexWood measurement database, these are loaded by the user as needed. Forests are catalogued on the measurement web service and available to the authorised user.

The user can then zoom into a particular forest and use the “Draw Polygon” option to identify a particular area they are interested in. In Figure 6 we have drawn out a polygon around the central point of the forest. We can then interrogate this polygon to a) make a direct query on all the trees within it or b) show the stands which are contained within the polygon. This is done by right buttoning on the polygon area (Figure 7)



Figure 6 : Defining a Polygon



Figure 7 : Menu Associated with Polygon

Choosing the “Get Stands” option shows the layout of stands from that forest within the polygon in different colours (Figure 8).

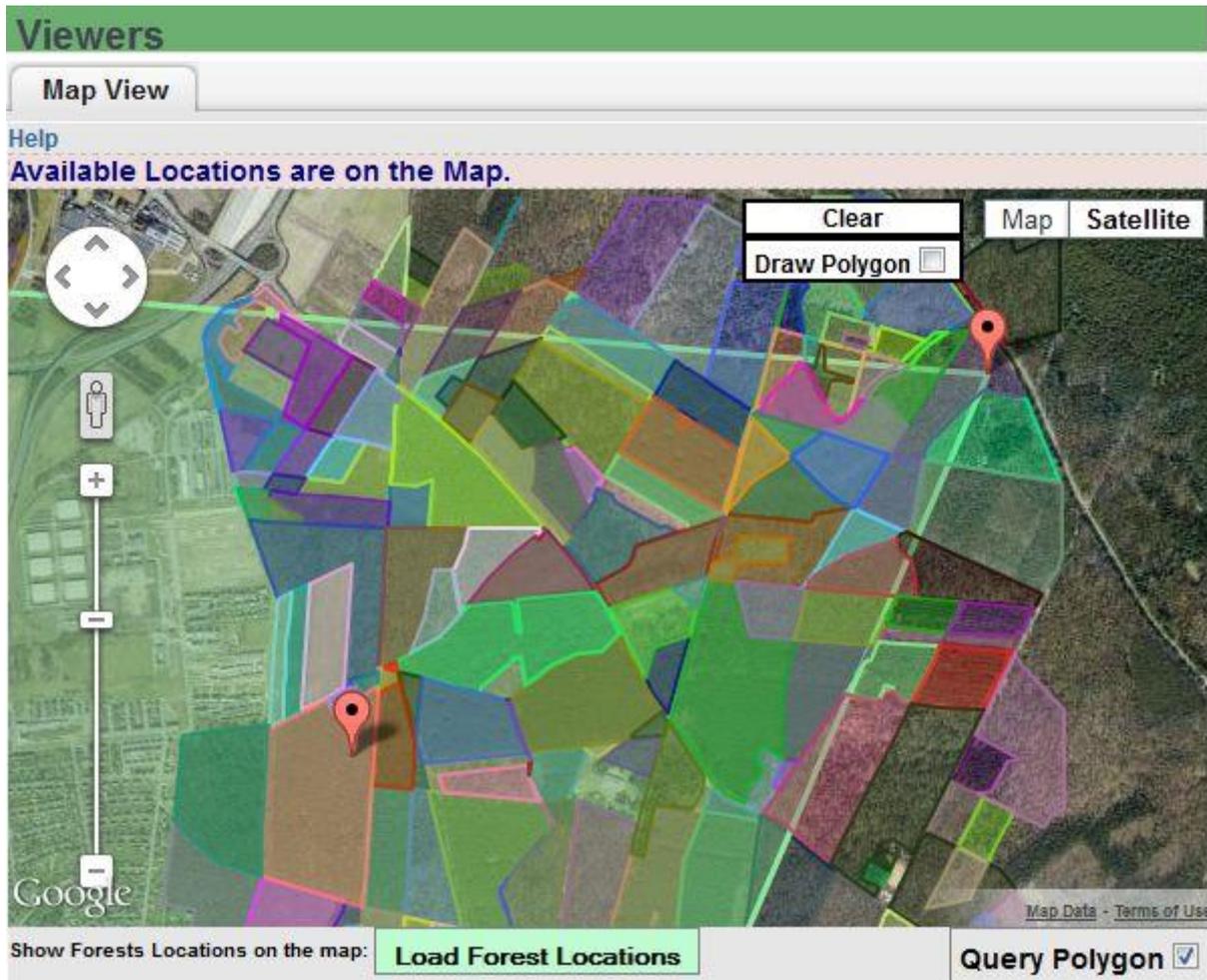


Figure 8 : Overlapping Stand Outlines within the Initial Polygon

These stands can individually be selected and queried through their characteristics by right clicking on any of the coloured stand polygons. This will present the user with a separate pop-up query window which allows the user to interrogate the database against a number of forest parameters such as DBH, Crown Diameter, etc. Figure 9 shows this query interface on a numbered stand 211 from the Karlsruhe forest retrieving stems which are between 20 and 30 cm DBH.

Query STAND_211
Please select parameters from the options provided.

Show: Trees

Where: DBH

Is Between 20

30

Unit: cm

Tree Species:

- Pinus sylverstris
- Picea abies
- Abies alba
- Abies procera
- Fagus sylvatica
- Mixed

Reference Or Derived: ▼

Go

Figure 9 : Forest/Stand Query Interface

The results of any query are shown at the bottom of the viewer window (see Figure 10). The stem files associated with this query are accessed by the controller in XML format. An example of which is shown in Appendix A.

Feature Id: STAND_216
Number of Hits: 44

Create a Stem file based on this query:

Id	Lower Corner	Upper Corner	Parameter Single Tree	Value Unit
SINGLETREE_576908	8.424408888712613 49.06070352533952	8.424408888712613 49.06070352533952	dbh	16.93 cm
SINGLETREE_579399	8.423628040995505 49.059881722613305	8.423628040995505 49.059881722613305	dbh	17.81 cm
SINGLETREE_595096	8.423950022834255 49.060626515156024	8.423950022834255 49.060626515156024	dbh	16.30 cm
SINGLETREE_595097	8.423901859086236 49.060377556983795	8.423901859086236 49.060377556983795	dbh	16.55 cm
SINGLETREE_599255	8.424128246947568 49.0606442185904	8.424128246947568 49.0606442185904	dbh	17.43 cm
SINGLETREE_603322	8.423920570427587 49.0605109113518	8.423920570427587 49.0605109113518	dbh	17.44 cm
SINGLETREE_603926	8.423797977384229 49.06010736962098	8.423797977384229 49.06010736962098	dbh	17.94 cm
SINGLETREE_604061	8.42439410239041 49.060573787400834	8.42439410239041 49.060573787400834	dbh	17.74 cm
SINGLETREE_615592	8.423976586043874 49.06070793504232	8.423976586043874 49.06070793504232	dbh	16.60 cm
SINGLETREE_619336	8.423999719859381 49.06056489813448	8.423999719859381 49.06056489813448	dbh	17.70 cm

Figure 10 : Results of a Query to Select Stems to be Harvested

4.3 Harvesting & Sawmill

The user needs to be able to carry out certain simulations with the software and these are typically quite complex in terms of the inputs and outputs. Therefore separate scenario builders are made to compose the input and viewers to display the output.

The first stage in the simulation process is harvesting. Harvesting represents the ways and means of cutting stem files – these represent actual stems in the forest. Harvesting encompasses the instructions necessary for cutting the forest (APT files) as well as information on the machines and labour necessary to carry it out for scheduling. From the previous example, the selected stems from the stand can be packaged for harvesting. To do this, the user must create a stem file and locate this on the harvest simulator server. This is all carried out automatically by the FlexWood Controller service when the user selects “create Stem File” (see Figure 11).



Figure 11 : Uploading of Selected Stem Descriptions from Measurement to Harvesting Servers

The user interface now moves to the “Scenarios” area where all harvesting and sawmill simulation scenarios will be constructed and evaluated. Figure 12 shows the initial scenario window presented to the user. There is a natural sequential order to the simulations and these are reflected in the order of the buttons. Each stage can be saved and restarted later.

Stage 1. Load STM files – this button invokes a query on the harvester web service for all known and named sets of stem files. The results are shown in the “Stem Files” menu.

Stage 2. Load APT files - this button invokes a query on the harvester web service for all known and named APT files. These are files which have been pre-generated, for example by the forest owner, harvesting company or sawmill. The results are shown in the “Stem Files” menu.

Stage 3. Harvest Simulation – this button invokes the harvesting web service on the selected STM and APT files. A breakout file is generated as a result and displayed through the UI.

Stage 4. Sawmill Simulation – this button invokes the sawn products web service on the selected STM and APT files. An end products file is generated (and breakout too) as a result and displayed through the UI.

These 4 steps are presented in the Scenarios interface (*Figure 12*) through appropriate button press and menu selections. Notice that different inputs can be segregated through the use of the Scenario tabs.

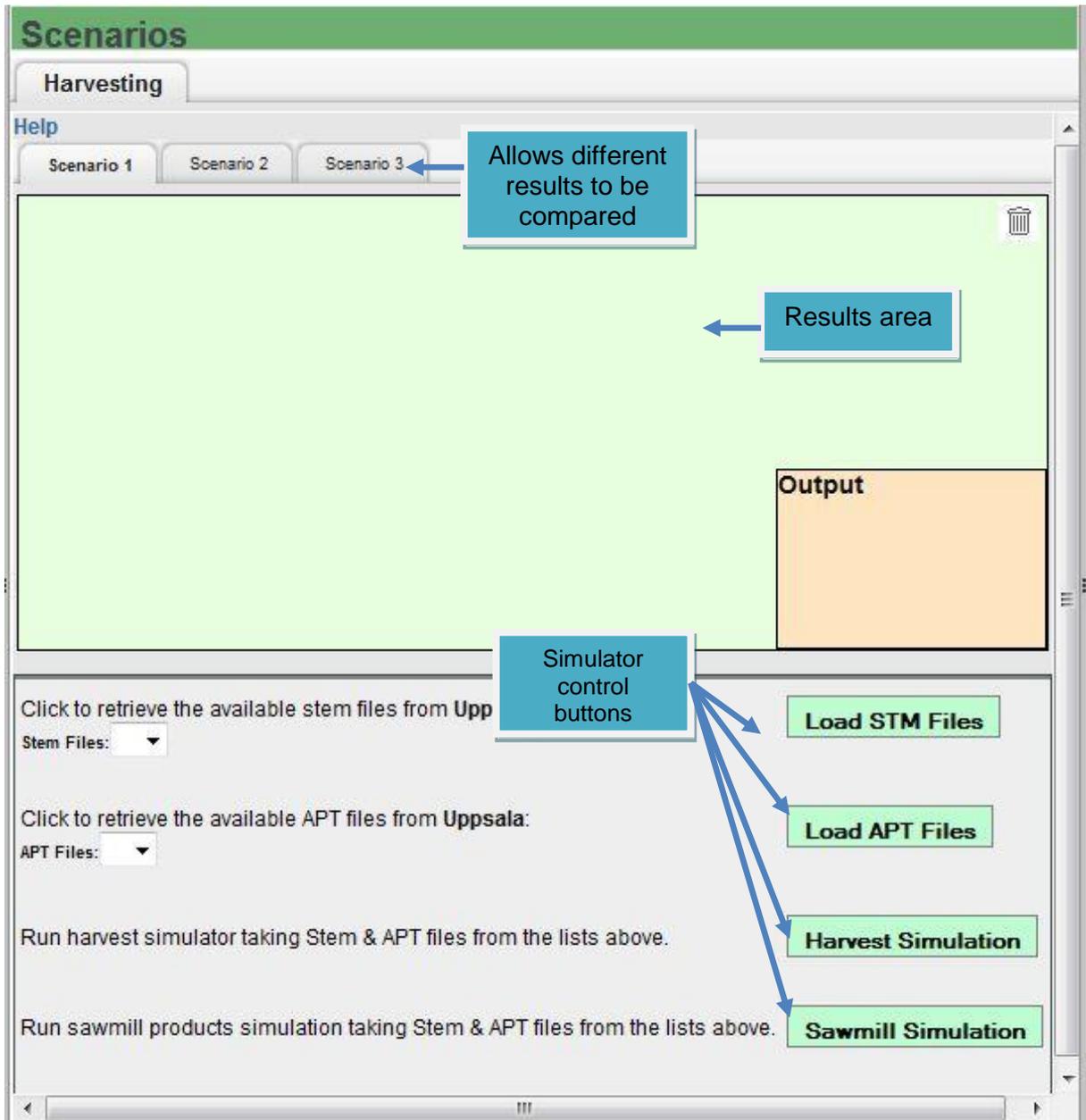


Figure 12 : Main Page for Scenario Building and Simulation

An example of the harvest simulation input is shown in Figure 13 with stem files selected from the harvesting server and associated with an APT file, again from the same server. The harvesting simulator is now ready to be run.

Files loaded

Click to retrieve the available stem files from **Uppsala**:
 Stem Files: Load STM Files

Click to retrieve the available APT files from **Uppsala**:
 APT Files: Load APT Files

Run harvest simulator taking Stem & APT files from the lists above. Harvest Simulation

Run sawmill products simulation taking Stem & APT files from the lists above. Sawmill Simulation

Figure 13 : Harvesting Input Selected for Scenario

During harvest simulation, the user is continually informed of the status of the different web services as data is passed around the network. In the case of harvesting, data will have already been sent and stored on the Skogforsk web server on stems. The harvesting web service is then invoked by the controller at UCC with the names of the input files on the harvesting web service. Once complete, Figure 14 then shows a number of options. The first is to see the resulting harvest breakout in a separate window, while the second is to associate a scenario name to these results.

Scenarios

Harvesting

Help

FlexWood1
Scenario 2
Scenario 3

[View Harvest Simulation Breakout Charts](#)

No sawmill simulation executed for this scenario.

Rename the tab as:

Rename
Hide

Output

Breakout

Figure 14 : Harvesting Results Ready for Inspection in one Scenario

One such breakout is shown in Figure 15 where different volumes of log lengths are presented to the user. Notice that this breakout only reflects the chosen stems from a single stand. If a whole stand is to be considered, then the stand query must return all the stems.

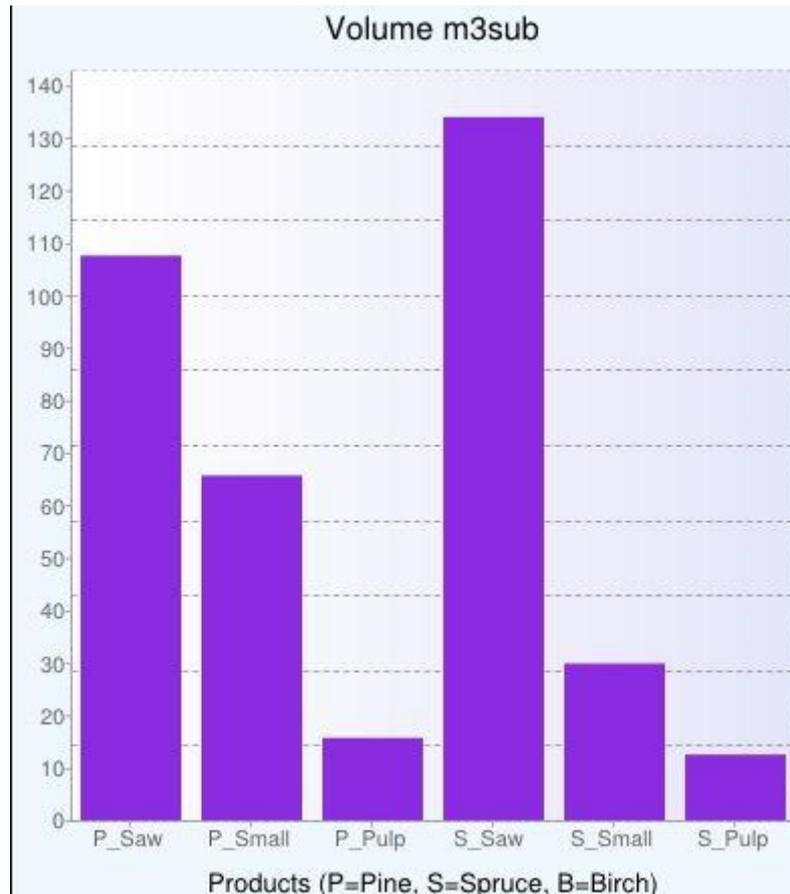


Figure 15 : Breakout Results of Harvest Simulator

Through varying the stems or stands and APT files, different scenarios can be generated. The user can then select which outcome (logs or end products) best matches their demands or requirements.

The next stage in the simulation process is the sawmill end product simulator. The sawmill represent the ways and means of cutting logs (generated by the harvesting) into timber pieces. Therefore sawmilling encompasses the cutting patterns necessary for turning logs into end products. From the previous example, the breakout logs from the harvest simulation can be packaged in XML format for the sawmill simulation. This is described in Appendix B. In order to do this, the system creates a harvester results file and this is sent through to the sawmill web service. Figure 16 shows a typical result of sawmill simulation. Here, the different volumes and values of the sawn timber products (from the harvest simulator) are shown. The XML format for the end-product timber is described in Appendix C.

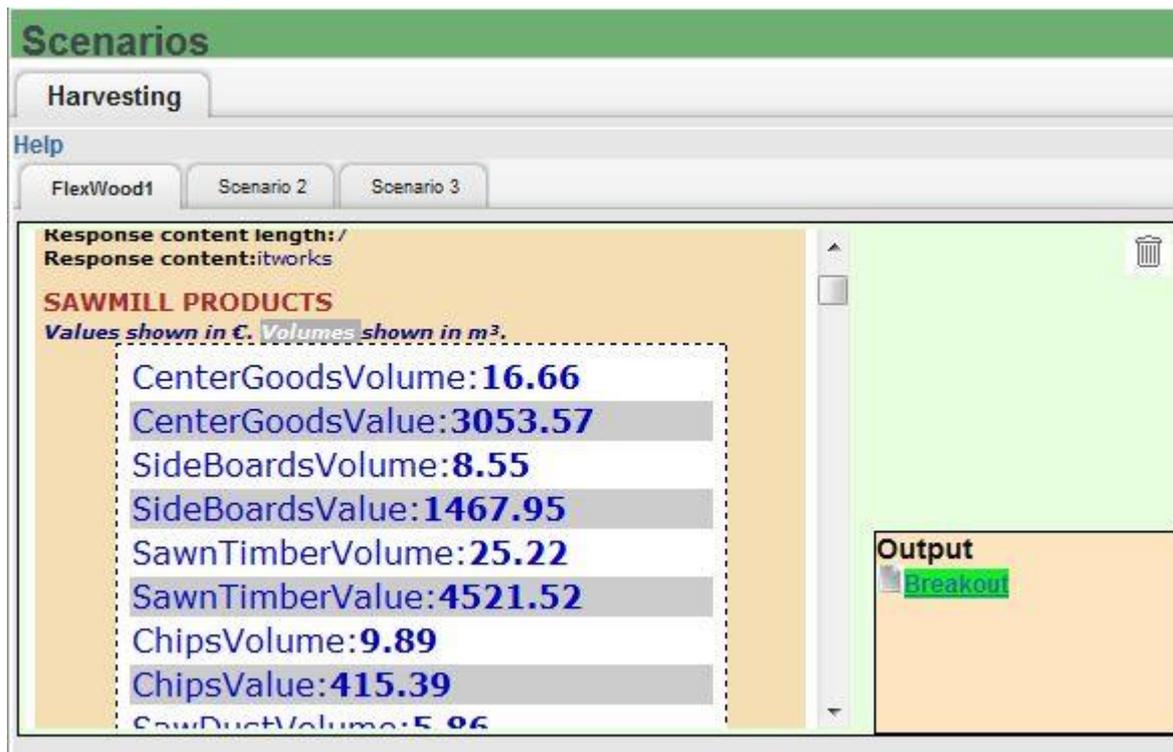


Figure 16 : Results of Sawmill Simulation

5. Technology description

We have chosen several standards solutions available in the open source market in order to demonstrate that there is no need of an extra investment for developing the full set of functionalities that Flexwood provides as for the client and server side.

5.1 Client side

Besides the HTML required by the browser to render the outputs, Flexwood uses also Javascript and Ajax technologies. A small set of libraries are described below:

MooTools

As a basement for the User Interface we use MooTools, which is a compact, modular, Object-Oriented JavaScript framework designed for the intermediate to advanced JavaScript solutions. It allows the programmer to write powerful, flexible, and cross-browser code provided through a very well documented API (application program interface).

There are several reasons why MooTools is used for developing Flexwood and these are the main ones:

- **Respects standards:** MooTools' code respects strict standards of the JavaScript language and doesn't throw any warnings. It's extensively documented and has meaningful variable names.
- **Open Source License:** MooTools is released under the Open Source MIT license, which gives you the possibility to use it and modify it in every circumstance.

- **Browser Compatibility:** MooTools is compatible and fully tested with Safari, Internet Explorer 6+, Firefox, Opera, and Chrome.
- **Support:** MooTools offers an online documentation and samples which are available for free.

OpenLayers

The project needs to show information depicted on online maps, and that is achieved with OpenLayers and Google Maps.

Overview

OpenLayers is a pure JavaScript library for displaying map data in most modern web browsers, with no server-side dependencies. OpenLayers implements a JavaScript API for building rich web-based geographic applications, similar to the Google Maps and MSN Virtual Earth APIs.

Reasons for using OpenLayers in Flexwood:

- **Easy-to-use:** OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds.
- **Support:** OpenLayers is developed and supported by a number of organizations around the world. We are also looking for sponsors to help support the community. If you are in a position where you want to support the development of OpenLayers, but do not have development resources to share, you may be interested in supporting through our sponsorship program.
- **Open Source License:** OpenLayers is Free Software, developed for and by the Open Source software community. It's completely free, Open Source JavaScript, released under the 2-clause BSD License (also known as the FreeBSD).

5.2 Server Side

Flexwood process controller and the rest of the servlets set have been built in Java EE 5, JDK version 1.6 and runs under the Apache Tomcat version 6.0.26.

Below is described in more detail, information regarding libraries used in this section.

SOAP

SOAP is fundamentally a stateless, one-way message exchange paradigm represented in XML format, but applications can create more complex interaction patterns like request/response messages that Flexwood is using by combining such one-way exchanges.

A SOAP message is fundamentally a one-way transmission between SOAP nodes, from a SOAP sender to a SOAP receiver, but SOAP messages in Flexwood are combined to implement more complex interaction patterns from request/response to multiple, back-and-forth "conversational" exchanges.

In summary, SOAP lets FlexWood send transparently messages to the web services that runs across the Flexwood system.

SOAP Sample

The *Figure 17* below show an example of a SOAP message used for a filtered query:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature version="1.1.0"
  xmlns:app="http://flexwood.uni-freiburg.de/flw"
  xmlns="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs
    http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
  <wfs:Query typeName="app:Stand">
    <ogc:Filter>
      <ogc:BBOX>
        <ogc:PropertyName>app:Stand/app:StandPolygon</ogc:PropertyName>
        <gml:Polygon name="1" srsName="EPSG:4326">
          <gml:outerBoundaryIs>
            <gml:LinearRing>
              <gml:posList><!-- dynamic list --></gml:posList>
            </gml:LinearRing>
          </gml:outerBoundaryIs>
        </gml:Polygon>
      </ogc:BBOX>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

Figure 17 : Example SOAP message for Filtered Query

Apache HttpComponents

In order to deal with HTTP requests within the server side Java code, FlexWood relies on the well-known HTTP library from the Apache Software Foundation in its version 4.1.2.

The Apache HttpComponents project is responsible for creating and maintaining a toolset of low level Java components focused on HTTP and associated protocols. This project functions under the Apache Software Foundation and is part of a larger community of developers and users.

The library is designed for extension while providing robust support for the base HTTP protocol and besides its robustness and ease-to-use facts; others have been taken into account for using it in Flexwood:

- **Integration:** It is integrated within the Flexwood Process Controller and let us building the HTTP server communication layer with other web services more efficiently and easy to manage.
- **Licensing:** It is released under Apache License v2.0. The Free Software Foundation considers the Apache License, Version 2.0 to be a free software license, compatible with version 3 of the GPL.

HttpComponents FlexWood sample

Below in *Figure 18* are some Java code lines demonstrating the ease of use of this library.

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
.
.
.
Integer connectionTimeoutMillis = 180000;
Integer socketTimeoutMillis = 180000;

String result = null;
try{
    //Set the client object
    HttpClient httpclient = new DefaultHttpClient();

    try {
        //Set the GET object
        HttpGet httpget = new HttpGet(harvestSimulatorURL);

        //Set the connection and socket timeout
        HttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams, connectionTimeoutMillis);
        HttpConnectionParams.setSoTimeout(httpParams, socketTimeoutMillis);

        //Set timeout params
        httpget.setParams(httpParams);

        //Execute the GET
        result = httpget.getRequestLine();
        HttpResponse resGet = httpclient.execute(httpget);
        .
        .
        .
    } finally {
        try {
            httpclient.getConnectionManager().shutdown();
        } catch (Exception ignore) {}
    }
}
}
```

Figure 18 : Example using the HTTP library for FlexWood

Google Charts

Flexwood needs to represent values in some suitable graphical manner and for this task we chose Google Chart Tools (<http://code.google.com/apis/chart/>) which gives us the ability of dealing with an API for returning a chart in an URL format, which is sent to the user interface straight away.

The Google Chart library for Java is called charts4j, and it is incorporated into our java web application environment because it's 100% pure core Java solution. There is no need for special graphics libraries and no need to pay fees.

- **Charts4j library:** charts4j is a free, lightweight chart & graph Java API. It enables us to programmatically create the charts available in the Google Chart Tools through a straightforward Java API.

A typical URL to generate the breakout of log would look like the following,

```
http://chart.apis.google.com/chart?cht=bvs&chco=8A2BE2&chs=600x450&chbh=50,10,8&chxt=y,y,x,x&chd=e:reghWB&chts=000000,16&chtt=Volume+m3sub&chg=100.0,10.0,3,2&chxp=1,50.0|3,50.0&chxs=1,000000,13,0|3,000000,13,0&chxr=0,0.0,29.0|1,0.0,100.0|3,0.0,100.0&chf=bg,s,F0F8FF|c,lg,0,E6E6FA,1.0,FFFFFF,0.0&chxl=1:|2:|P_C11|P_C13+|P_pulp|3:|Products+%28P%3DPine%2C+S%3DSpruce%2C+B%3DBirch%29
```

6. Stakeholders

The architecture supports various stakeholders in different locations. Depending on the identification of the stakeholder (e.g. forest planner, forest owner, harvesting company or sawmill) only certain services are available through the user interface. The user can save their own parameters for harvesting or cutting and apply these to whatever scenario they create or is available on the FlexWood system. The stakeholder can also upload information at any point along the supply chain and therefore an interaction can start with FlexWood, for example, at roadside logs.

By adopting a web service approach means that wherever the FlexWood URL can be obtained using Wi-Fi, Internet, Modem or Cell Phone, so FlexWood can be used. In particular it can be used in the office and in the forest to view or simulate directly various features or operations which are not easily obtained otherwise.

7. Appendices

Appendix A – Stem File XML

```
<?xml version="1.0" encoding="UTF-8" ?>
± <wfs:FeatureCollection numberOfFeatures="4" xmlns:app="http://flexwood.uni-
freiburg.de/flw" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://flexwood.uni-freiburg.de/flw http://flexwood.felis.uni-
freiburg.de:8080/deegree-
wfs/services?SERVICE=WFS&VERSION=1.1.0&REQUEST=DescribeFeatureTy
pe&TYPENAME=app:StandingStem&NAMESPACE=xmlns\(app=http://flexwood.
uni-freiburg.de/flw\)
http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.1.0/wfs.xsd>
± <gml:boundedBy>
± <gml:Envelope srsName="EPSG:31467">
<gml:lowerCorner>3456709.998 5432596.53</gml:lowerCorner>
<gml:upperCorner>3458011.778 5435600.872</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
± <gml:featureMember>
± <app:StandingStem gml:id="stem_ID_1">
± <gml:boundedBy>
± <gml:Envelope srsName="EPSG:31467">
<gml:lowerCorner>3457901.746 5434805.945</gml:lowerCorner>
<gml:upperCorner>3457901.879 5434806.08</gml:upperCorner>
```

```

        </gml:Envelope>
        </gml:boundedBy>
    <app:species />
        <app:age />
    : <app:stemdisks>
    - <!--
    xlink:href="#stemdisk_ID_51189"
    -->
    : <app:stemdisks gml:id="stemdisk_ID_51189">
        : <gml:boundedBy>
        : <gml:Envelope srsName="EPSG:31467">
            <gml:lowerCorner>3457901.777 5434806.08</gml:lowerCorner>
            <gml:upperCorner>3457901.777 5434806.08</gml:upperCorner>
        </gml:Envelope>
        </gml:boundedBy>
            <app:diameter>188</app:diameter>
            <app:height>0.0</app:height>
            <app:volume>0</app:volume>
        : <app:stemDiskPosition3d>
        : <gml:Point srsName="EPSG:31467">
        <gml:pos srsDimension="3" srsName="EPSG:31467">3457901.777 5434806.08
        0.0</gml:pos>
        </gml:Point>
        </app:stemDiskPosition3d>
        </app:stemdisks>
        </app:stemdisks>
    : <app:stemdisks>
    - <!--
    xlink:href="#stemdisk_ID_51190"
    -->
    : <app:stemdisks gml:id="stemdisk_ID_51190">
        : <gml:boundedBy>
        : <gml:Envelope srsName="EPSG:31467">
            <gml:lowerCorner>3457901.781 5434806.073</gml:lowerCorner>
            <gml:upperCorner>3457901.781 5434806.073</gml:upperCorner>
        </gml:Envelope>
        </gml:boundedBy>
            <app:diameter>187</app:diameter>
            <app:height>1.0</app:height>
            <app:volume>2.76</app:volume>
        : <app:stemDiskPosition3d>
        : <gml:Point srsName="EPSG:31467">
        <gml:pos srsDimension="3" srsName="EPSG:31467">3457901.781 5434806.073
        0.1</gml:pos>
        </gml:Point>
        </app:stemDiskPosition3d>
        </app:stemdisks>
        </app:stemdisks>
    : <app:stemdisks>

```

Appendix B – Log Breakout File XML (for WoodCIM)

```
- <FW_WoodCim_Output xsi:schemaLocation="urn:skogforsk:BuckingSim:1:draft
FW_WoodCim.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
- <Log>
  <Species>Pine</Species>
  <Grade>P_C11</Grade>
  <Top_DiameterOb>161</Top_DiameterOb>
  <Top_DiameterUb>154</Top_DiameterUb>
  <Length>494</Length>
  <TaperOb>0.0910931155085564</TaperOb>
  <TaperUb>0.0890688225626945</TaperUb>
  <VolumeOb>0.1184</VolumeOb>
  <VolumeUb>0.1090</VolumeUb>
```

```
</Log>
```

Appendix C – Sawmill Products File XML

```
- <FlexWoodSawModelOutput>
```

```
- <General>
```

```
  <CenterGoodsVolume>17.58</CenterGoodsVolume>
  <CenterGoodsValue>3221.48</CenterGoodsValue>
  <SideBoardsVolume>9.11</SideBoardsVolume>
  <SideBoardsValue>1556.69</SideBoardsValue>
  <SawnTimberVolume>26.69</SawnTimberVolume>
  <SawnTimberValue>4778.17</SawnTimberValue>
  <ChipsVolume>10.33</ChipsVolume>
  <ChipsValue>433.98</ChipsValue>
  <SawDustVolume>6.18</SawDustVolume>
  <SawDustValue>141.41</SawDustValue>
  <BarkVolume>5.39</BarkVolume>
  <BarkValue>134.15</BarkValue>
  <LogsVolume>59.01</LogsVolume>
  <LogsValue>4425.90</LogsValue>
  <SawnTimberYieldPercentage>45.23</SawnTimberYieldPercentage>
```

```
</General>
```

```
- <CenterGoods>
```

```
- <Product>
```

```
  <Thickness>38</Thickness>
```

```
<Width>200</Width>
<Grade>VI</Grade>
<Volume>0.23</Volume>
</Product>
- <Product>
  <Thickness>50</Thickness>
  <Width>150</Width>
  <Grade>VI</Grade>
  <Volume>0.16</Volume>
</Product>
</CenterGoods>
- <SideBoards>
  - <Product>
    <Thickness>22</Thickness>
    <Width>150</Width>
    <Grade>US</Grade>
    <Volume>0.46</Volume>
  </Product>
  - <Product>
    <Thickness>22</Thickness>
    <Width>150</Width>
    <Grade>V</Grade>
    <Volume>1.00</Volume>
  </Product>
- </SideBoards>
```