SSE/05537/IFS/002 - Revision: 1.57 Date: 17 Oct 2011

Laboratory Interface to NorFor

05537 - NORFOR

Prepared for:

Søren Frichs Vej 39
DK-8000 Aarhus C

Gothersgade 14
DK-1123 Copenhagen K

Tel.: +45 8943 2000
Fax: +45 8943 2020

sse@systematic.dk
www.systematic.dk

Table of Contents

1 Introduction

The FAS solution described in this document extends the NorFor system with the needed ability to handle and administrate feed stuff analyses.

The webservice is called FAS for .Net applications and FASNonDotNet for non .Net applications, and is available at the following url's:

Test environment:

- **http://testwebservice.norfor.info/FAS.asmx**
- **http://testwebservice.norfor.info/FASNonDotNet.asmx**

Production environment:

- **http://webservice.norfor.info/FAS.asmx**
- **http://webservice.norfor.info/FASNonDotNet.asmx**

1.1 The NorFor system

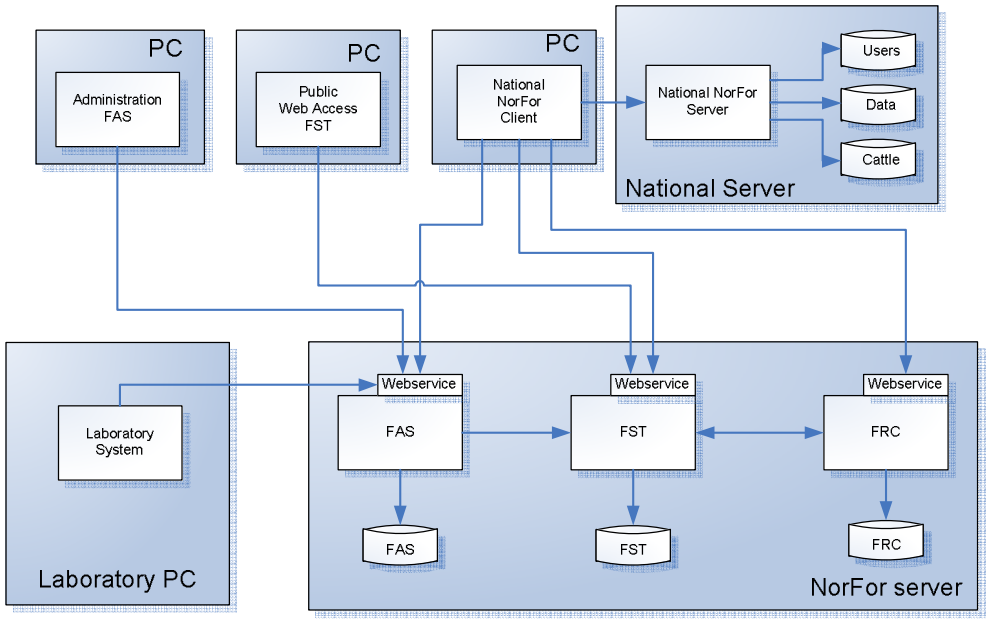


Figure 1 Overview of the entire NorFor system.

The abbreviation in Figure 1:

- FST is Feed Stuff Table
- FRC is Feed Ration Calculator

- FAS is Feed Analysis System

2 Technical solution

All services are provided through .Net WebServices. The following sections elaborate on this point for .Net applications and non .Net applications.

2.1 The Laboratory - FAS/NorFor interface for .Net

The interface between the laboratories and FAS/NorFor is .Net WebServices. By .Net WebServices is meant that the input and output parameters of the WebServices can be and most often are .Net datasets. This means that the laboratory applications have to be able to work with .Net datasets. Thus the new application that the laboratories will develop to integrate their existing system with the FAS/NorFor WebServices must be able to handle .Net datasets (standard if programmed in a .Net language, e.g. (C#, Visual Basic.Net, C++.Net, and others)). The following figure shows what has to be made by the laboratories:

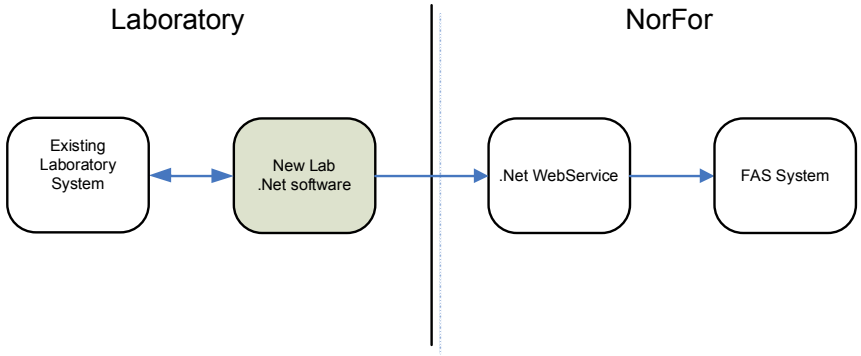


Figure 2. The integration the laboratories make to the FAS/NorFor system must be able to work with .Net datasets. The integration application is shown in dark.

2.2 The Laboratory - FASNonDotNet/NorFor interface for non .Net

The interface between the laboratory clients and FASNonDotNet/NorFor is .Net WebServices. But instead of using datasets as input and output parameters in the webservice methods there is used arrays of objects instead. These arrays of object contain the same variables/data types as were available in the .Net datasets.

This means that the laboratory applications don't have to be able to work with .Net datasets, so it can be written in program languages as Java.

2.3 Domain Object Model for FAS and FST

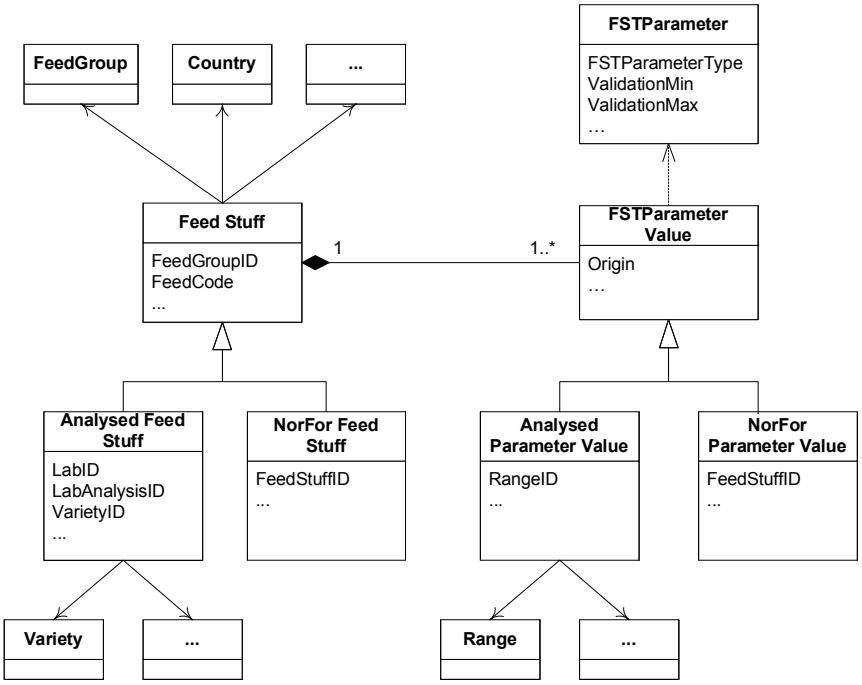


Figure 3 Conceptual Domain Object Model. Arrows indicates references (e.g. a Feed Stuff has a reference to a Country), triangles indicates inheritance (e.g. Analysed Feed Stuff *is a* Feed Stuff), black diamonds indicates hard aggregations (A FSTParameterValue cannot exist if not associated with a Feed Stuff).

2.3.1 Description of the Objects

Each of the object classes is briefly described.

Class	Description
Feed Stuff	Represents both NorFor feed stuffs and analysed feed stuffs. The Feed Stuff class defines the relations to other objects and the attributes common to NorFor and analysed feed stuffs. A Feed Stuff object: • Belongs to a specific feed group. (E.g. cereal grains, Forage and roughage, mineral etc.) • Is of a specific feed type. (E.g. Forage and roughage have the types Whole crops, Hay and straw, Grass pellets etc.) • Have a number of parameter values. (E.g. dry

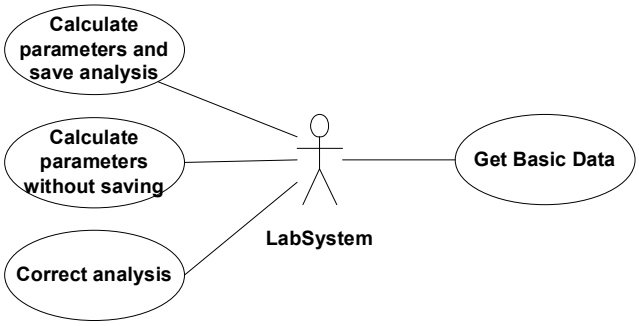
	matter, sugar, calcium and price).
NorFor Feed Stuff	Represents a feed stuff on the NorFor level. It is a specialization of Feed Stuff. The NorFor feed stuff is an *average* feed stuff, that has been created based on several analyses of several feed stuffs from all over the NorFor countries. It defines some standard parameter values that can be used for a feed stuff, in case the values of these parameters have not been analysed for the actual feed stuff.
Analysed Feed Stuff	Represents a feed stuff as delivered from the laboratory. It is a specialization of Feed Stuff. An Analysed Feed Stuff must be associated with a NorFor Feed Stuff. An analysed feed stuff stored in the NorFor system belongs to the laboratory that stored it. The laboratory can make changes to the analysed feed stuff if they wish to. Before an analysed feed stuff can be used in the rations by the farmer/advisor, it must be copied to the herd (another table).
FeedGroup, Country, Variety, …	Represents the valid values for different attributes of the feed stuff or the parameter values. In NorFor it has been given the name *Basic Data*.
FSTParameter	Describes a parameter of a feed stuff that may have a value or amount designation (e.g. dry matter, sugar, calcium and price). A FST Parameter has information like: Name, Abbreviation, Unit, etc.
FSTParameter-Value	Represents the value of one of the feed stuff parameters (e.g. the amount of sugar per kilo dry matter) common for both NorFor and analysed feed stuffs. The FSTParameterValue class defines the relations to other objects and the attributes common to NorFor and analysed parameters.
NorFor Parameter value	A specialization of the FSTParameterValue used for NorFor feed stuffs. A NorFor Parameter Value: • Feed stuff ID • Origin of the value. (Calculated or Entered)
Analysed Parameter Value	A specialization of the FSTParameterValue used for analysed feed stuffs. An Analysed Parameter Value:

	• Range • AnalysisReferenceMethod • Origin of the value. (Calculated or Entered)
Analysed Parameter Value With Statistics	A extension of the Analysed Parameter Value used as return value when calculating analysed feedstuffs. It is adding statistics to parameters values. An Analysed Parameter Value With Statistics: • Range • AnalysisReferenceMethod • Origin of the value. (Calculated or Entered) • Number Of Analysis • Average • Deviation • 10 Pct Percentile • 90 Pct Percentile • NorForValuesUsed (true if no statistics where found, and NorFor values was used instead)

3 Use case view

3.1 Overview

The following Use Case diagram gives an overview of the NorFor functions available to the laboratories. The Use cases are described in detail in the following paragraphs.



3.2 FAS WebServices

The Use Cases in this section describe the Web Method available to the laboratories.

One WebMethod is available to work with the analysed feedstuff. Using that WebMethod, the following operations can be done:

- Calculate an analysed feedstuff without saving
- Calculate and save a new analysed feedstuff
- Calculate and update an analysed feedstuff that has been saved before.

The operations are all covered in the following use case:

Calculate parameters of analysed feed stuff (and save it)

Precondition: The laboratory has a new or updated analysed feed stuff that must be calculated (and stored) in the system.

Steps:

1. Call web service method with an AnalysedFeedStuff data set.

2. The transfer is logged on FAS Server.
3. Simple validation of the analysis data set is performed.
4. Calculation of the calculated parameter values are performed, based on the analysed parameter values from the laboratory and the standard values from the NorFor feed stuff.
5. The analysis is saved in FAS, if the parameter "save" is set to true. If the analysis already exists in the database, it is updated with the new analysed values.
6. An AnalysedFeedStuff dataset is returned. This holds all the analysed parameter values together with the standard and calculated parameter values.

Comments: Input data is validated according to the rules that apply for the feed stuffs in the NorFor system, which basically is a rough min/max validation.

The parameter limits are standard limits for the parameter.

The analysed feed stuff is uniquely identified by the LabID and the LabAnalysisID.

Get Basic Data

Precondition: The laboratory wants all FAS Basic Data from NorFor

Steps:
1. Call web service method.
2. Retrieve the returned data set, which holds a list of all Basic data in FAS.

Comments: The Basic Data include several lists, such as the list of possible feed stuff parameters.

4 WebMethods

This section contains the details of each Web Method available to the laboratories.

The details of the datasets that are passed back and forth between the client and the Web Method are shown in the next chapter.

4.1 Calculate Analysed Feed Stuff

The following method is used for all operations that can be done on an analysed feedstuff.

- Calculate an analysis without saving
- Calculate and save a new analysis
- Calculate and update an analysis that has been saved before.

Name: CalculateAnalysedFeedStuff	
Input:	Username, Password, **AnalysedFeedStuff Dataset**, save (boolean)
Description:	The number of 'analysed parameters' in the **AnalysedFeedStuff** dataset is optional, since only a subset of parameters might be analysed in the given analysis. If the name is not given when creating a new **AnalysedFeedStuff**, it is given the name of the NorFor feed stuff. If the analysis already exists in the database then it is merged with the existing analysis. After this merge the analysis is merged with the the standard NorFor feed stuff identified by the FeedGroup and FeedCode of the given **AnalysedFeedStuff**. This update happens before the calculation is performed. If the FeedGroup or FeedCode has been changed, the **AnalysedFeedStuff** is merged with the new NorFor feed stuff identified by the newly FeedGroup and FeedCode. The attributes of the analysis are updated if a value is specified (e.g. If HarvestDate is specified then the previous value is overridden. If HarvestDate is NULL the old HarvestDate is used). If the Boolean *save* is set to true the Analysed feed stuff along with the values of the 'calculated parameters' and the standard NorFor parameters are saved in the NorFor database.

	If the OriginID is set to ENT (ENTered) and the Value is set to NaN (Not a Number, e.g. double.NaN) in an AnalysedParameterValue, then the parameter is deleted. This is done before the merge with the NorFor feed stuff. The resulting Feed Stuff is returned to laboratory system
Output:	An **AnalysedFeedStuff Dataset** containing the feed stuff and both calculated and non-calculated parameter values.
Exceptions:	*AccessDeniedException* if the username/password is invalid. *LabNotAuthorizedException* if the user is not authorized to save the given AnalysedFeedStuff dataset because it belongs to another Laboratory. *DatasetStructureException* if the **AnalysedFeedStuff** data set in the input is not structured correctly. *NorForFeedStuffNotFoundException* if no NorFor FeedStuff exists with the FeedGroup and FeedCode of the given NorFor feedstuff ID. *AnalysedParameterValueValidationException* if validation fails. If this is the case, information about which parameters failed is returned to the client. *CalculationFailedException* if calculations fail. (Calculations fail if some calculations results in division by zero or other mathematic impossibilities.).

4.1.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to locally create an AnalysedFeedStuff with one AnalysedParameterValue entry, call the CalculateAnalysedFeedStuff web-method and use the data returned by the web-method:

```
// Instantiate the web-service proxy
FAS wsFas = new FAS();

AnalysedFeedStuffDataset dsAnalysedFS = new
    AnalysedFeedStuffDataset();

// Create a new AnalysedFeedStuff data row and
// populate selected properties
AnalysedFeedStuffDataset.AnalysedFeedStuffRow aRow =
    dsAnalysedFS.AnalysedFeedStuff.NewAnalysedFeedStuffRow();

aRow.LabID          = 1001;
aRow.LabAnalysisID  = "1";
aRow.DateOfSampling = DateTime.Now;
aRow.DateArrivedLab = DateTime.Now;
aRow.StatusID       = 3;
aRow.LabComment     = "A comment";
aRow.FeedGroupID    = 1;
aRow.FeedCode       = 1;
```

```
aRow.HerdID          = 2;
aRow.CountryID       = "DK";
aRow.FeedName        = "New analysed feed stuff";

// Add the data row to the data set
dsAnalysedFS.AnalysedFeedStuff.AddAnalysedFeedStuffRow(aRow);

// Add a AnalysedParameterValue data rows to the data set
dsAnalysedFS.AnalysedParameterValue.AddAnalysedParameterValueRow(
    aRow,      // AnalysedFeedstuff reference
    "Ash",     // Parameter ID
    "ANA",     // Origin ID
    1,         // Range ID
    1,         // Method ID
    "Method",  // Method
    0.5,       // Std. Deviation
    33         // Value
);

// Call the web-method
dsAnalysedFS = wsFas.CalculateAnalysedFeedStuff(
    "MyUserName", "MyPassword", dsAnalysedFS, false);

// Use the calculated AnalysedFeedStuff
long   id = dsAnalysedFS.AnalysedFeedStuff[0].FeedStuffID;
double Mg =
   dsAnalysedFS.AnalysedParameterValue.FindByFeedStuffIDFSTParameterID(
       id, "Mg").Value;
```

4.1.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to locally create an AnalysedFeedStuff with one AnalysedParameterValue entry, call the CalculateAnalysedFeedStuff web-method and use the data returned by the web-method:

```
// Instantiate the web-service proxy
FASNonDotNet wsFAS = new FASNonDotNet();

AnalysedFeedStuffDataset dsAnalysedFS = new AnalysedFeedStuffDataset();

// Create a new AnalysedFeedStuff data row and
// populate selected properties
AnalysedFeedStuff aRow = new AnalysedFeedStuff();

aRow.LabID           = 1001;
aRow.LabAnalysisID   = "1";
aRow.DateOfSampling  = DateTime.Now;
aRow.DateArrivedLab  = DateTime.Now;
aRow.StatusID        = 3;
aRow.LabComment      = "A comment";
aRow.FeedGroupID     = 1;
aRow.FeedCode        = 1;
aRow.HerdID          = 2;
aRow.CountryID       = "DK";
aRow.FeedName        = "New analysed feed stuff";

// Add the data row to the data set
dsAnalysedFS.AnalysedFeedStuff = new AnalysedFeedStuff[1];
dsAnalysedFS.AnalysedFeedStuff[0] = aRow;

// Create a new AnalysedParameterValue data row and
// populate selected properties
AnalysedParameterValue newRow = new AnalysedParameterValue();

newRow.FeedStuffID = aRow.FeedStuffID;
newRow.FSTParameterID = "Ash";
newRow.OriginID = "ANA";
```

```
newRow.RangeID = 1;
newRow.MethodID = 1;
newRow.MethodIDSpecified = true;
newRow.Method = "Method";
newRow.StandardDeviation = 0.5;
newRow.StandardDeviationSpecified = true;
newRow.Value = 33;

// Add the data row to the data set
dsAnalysedFS.AnalysedParameterValue = new AnalysedParameterValue[1];
dsAnalysedFS.AnalysedParameterValue[0] = newRow;

// Call the web-method
dsAnalysedFS = wsFAS.CalculateAnalysedFeedStuff(
       "MyUserName", "MyPassword", dsAnalysedFS, false);

// Use the calculated AnalysedFeedStuff
long id = dsAnalysedFS.AnalysedFeedStuff[0].FeedStuffID;
double Mg =
Array.Find<AnalysedParameterValue>(dsAnalysedFS.AnalysedParameterValue,
       delegate(AnalysedParameterValue apv) {
           return apv.FeedStuffID == id && apv.FSTParameterID == "Mg";
       }
       ).Value;
```

4.2 Calculate Analysed Feed Stuff With Statistics

The following method is used for all operations that can be done on an analysed feedstuff. This is very similar to CalculateAnalysedFeedStuff except for the returned dataset contains statistics for each parameter value, based on local region, country or NorFor-statistics.

- Calculate an analysis without saving
- Calculate and save a new analysis
- Calculate and update an analysis that has been saved before.

Name: CalculateAnalysedFeedStuffWithStatistics	
Input:	Username, Password, **AnalysedFeedStuff Dataset**, save (boolean)
Description:	The number of 'analysed parameters' in the **AnalysedFeedStuff** dataset is optional, since only a subset of parameters might be analysed in the given analysis. If the name is not given when creating a new **AnalysedFeedStuff**, it is given the name of the NorFor feed stuff. If the analysis already exists in the database then it is merged with the existing analysis. After this merge the analysis is merged with the the standard NorFor feed stuff identified by the FeedGroup and FeedCode of the given **AnalysedFeedStuff**. This update happens before the calculation is performed. If the FeedGroup or FeedCode has been changed, the **AnalysedFeedStuff** is merged with the new NorFor

	feed stuff identified by the newly FeedGroup and FeedCode. The attributes of the analysis are updated if a value is specified (e.g. If HarvestDate is specified then the previous value is overridden. If HarvestDate is NULL the old HarvestDate is used). If the Boolean *save* is set to true the Analysed feed stuff along with the values of the 'calculated parameters' and the standard NorFor parameters are saved in the NorFor database. If the OriginID is set to ENT (ENTered) and the Value is set to NaN (Not a Number, e.g. double.NaN) in an AnalysedParameterValue, then the parameter is deleted. This is done before the merge with the NorFor feed stuff. The resulting Feed Stuff is returned to laboratory system
Output:	An **AnalysedFeedStuffWithStatistics Dataset** containing the feed stuff and both calculated and non-calculated parameter values and statistics for all parameters. The string field **StatisticsType** reports the source of the statistics in the return value. The possible values for **StatisticsType**: • **LOCAL_REGION_VALUES** • **NORFOR_STANDARD_VALUES** • **NATIONAL_VALUES**
Exceptions:	*AccessDeniedException* if the username/password is invalid. *LabNotAuthorizedException* if the user is not authorized to save the given AnalysedFeedStuff dataset because it belongs to another Laboratory. *DatasetStructureException* if the **AnalysedFeedStuff** data set in the input is not structured correctly. *NorForFeedStuffNotFoundException* if no NorFor FeedStuff exists with the FeedGroup and FeedCode of the given NorFor feedstuff ID. *AnalysedParameterValueValidationException* if validation fails. If this is the case, information about which parameters failed is returned to the client. *CalculationFailedException* if calculations fail. (Calculations fail if some calculations results in division by zero or other mathematic impossibilities.).

4.2.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to locally create an AnalysedFeedStuff with one AnalysedParameterValue entry, call the CalculateAnalysedFeedStuff web-method and use the data returned by the web-method:

```
// Instantiate the web-service proxy
FAS wsFas = new FAS();

AnalysedFeedStuffDataset dsAnalysedFS = new
    AnalysedFeedStuffDataset();

// Create a new AnalysedFeedStuff data row and
// populate selected properties
AnalysedFeedStuffDataset.AnalysedFeedStuffRow aRow =
    dsAnalysedFS.AnalysedFeedStuff.NewAnalysedFeedStuffRow();

aRow.LabID          = 1001;
aRow.LabAnalysisID  = "1";
aRow.DateOfSampling = DateTime.Now;
aRow.DateArrivedLab = DateTime.Now;
aRow.StatusID       = 3;
aRow.LabComment     = "A comment";
aRow.FeedGroupID    = 1;
aRow.FeedCode       = 1;
aRow.HerdID         = 2;
aRow.CountryID      = "DK";
aRow.FeedName       = "New analysed feed stuff";

// Add the data row to the data set
dsAnalysedFS.AnalysedFeedStuff.AddAnalysedFeedStuffRow(aRow);

// Add a AnalysedParameterValue data rows to the data set
dsAnalysedFS.AnalysedParameterValue.AddAnalysedParameterValueRow(
    aRow,      // AnalysedFeedstuff reference
    "Ash",     // Parameter ID
    "ANA",     // Origin ID
    1,         // Range ID
    1,         // Method ID
    "Method", // Method
    0.5,       // Std. Deviation
    33         // Value
);

// Call the web-method
var dsAnalysedFSWithStatistics =
    wsFas.CalculateAnalysedFeedStuffWithStatistics(
      "MyUserName", "MyPassword", dsAnalysedFS, false);

// Use the calculated AnalysedFeedStuffWithStatistics
long   id =

dsAnalysedFSWithStatistics.AnalysedFeedStuffWithStatistics[0].FeedStuff
ID;
double Mg =
    dsAnalysedFSWithStatistics.AnalysedParameterValueWithStatistics.
       FindByFeedStuffIDFSTParameterID(id, "Mg").Value;
```

4.2.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to locally create an AnalysedFeedStuff with one AnalysedParameterValue entry, call the CalculateAnalysedFeedStuff web-method and use the data returned by the web-method:

```
// Instantiate the web-service proxy
FASNonDotNet wsFAS = new FASNonDotNet();

AnalysedFeedStuffDataset dsAnalysedFS = new AnalysedFeedStuffDataset();

// Create a new AnalysedFeedStuff data row and
// populate selected properties
AnalysedFeedStuff aRow = new AnalysedFeedStuff();

aRow.LabID          = 1001;
aRow.LabAnalysisID  = "1";
aRow.DateOfSampling = DateTime.Now;
aRow.DateArrivedLab = DateTime.Now;
aRow.StatusID       = 3;
aRow.LabComment     = "A comment";
aRow.FeedGroupID    = 1;
aRow.FeedCode       = 1;
aRow.HerdID         = 2;
aRow.CountryID      = "DK";
aRow.FeedName       = "New analysed feed stuff";

// Add the data row to the data set
dsAnalysedFS.AnalysedFeedStuff = new AnalysedFeedStuff[1];
dsAnalysedFS.AnalysedFeedStuff[0] = aRow;

// Create a new AnalysedParameterValue data row and
// populate selected properties
AnalysedParameterValue newRow = new AnalysedParameterValue();

newRow.FeedStuffID = aRow.FeedStuffID;
newRow.FSTParameterID = "Ash";
newRow.OriginID = "ANA";
newRow.RangeID = 1;
newRow.MethodID = 1;
newRow.MethodIDSpecified = true;
newRow.Method = "Method";
newRow.StandardDeviation = 0.5;
newRow.StandardDeviationSpecified = true;
newRow.Value = 33;

// Add the data row to the data set
dsAnalysedFS.AnalysedParameterValue = new AnalysedParameterValue[1];
dsAnalysedFS.AnalysedParameterValue[0] = newRow;

// Call the web-method
AnalysedFeedStuffWithStatisticsDataset  dsAnalysedFSWithStatistics =
    wsFAS.CalculateAnalysedFeedStuffWithStatistics(
        "MyUserName", "MyPassword", dsAnalysedFS, false);

// Use the calculated AnalysedFeedStuffWithStatistics
long id =
dsAnalysedFSWithStatistics.AnalysedFeedStuffWithStatistics[0].FeedStuff
ID;
double Mg =
Array.Find<AnalysedParameterValueWithStatistics>(dsAnalysedFS.AnalysedP
arameterValueWithStatistics,
        delegate(AnalysedParameterValueWithStatistics apv) {
            return apv.FeedStuffID == id && apv.FSTParameterID == "Mg";
        }
        ).Value;
```

4.3 Delete Analysed Feed Stuff

The following method is used for deleting an existing analysed feed stuff.

Name: DeleteAnalysedFeedStuff

Input:	Username, Password, LabID, LabAnalysisID
Description:	Deletes an analysed feed stuff. All parameter values are implicitly deleted also.
Output:	Void
Exceptions:	*AccessDeniedException* if the user/password is invalid. *AnalysisFeedStuffNotFoundException* if no Analysis FeedStuff exists with the given LabID and LabAnalysisID. *LabNotAuthorizedException* if the user is not authorized to delete the given AnalysedFeedStuff dataset because it belongs to another Laboratory.

4.3.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to delete an existing analysed feed stuff:

```
// Instantiate the Web-service proxy class
FAS wsFas = new FAS();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
long labID = 1000; // The owner of the analysed feed stuff
string labAnalysisID = "MyAnalysisID";

// Call the web-method to deletethe analysed feed stuff
wsFas.DeleteAnalysedFeedStuff(username, password, labID,
labAnalysisID);
```

4.3.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to delete an existing analysed feed stuff:

```
// Instantiate the Web-service proxy class
FASNonDotNet wsFAS = new FASNonDotNet();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
long labID = 1000; // The owner of the analysed feed stuff
string labAnalysisID = "MyAnalysisID";

// Call the web-method to deletethe analysed feed stuff
wsFAS.DeleteAnalysedFeedStuff(username, password, labID,
labAnalysisID);
```

4.4 Find Analysed Feed Stuffs

The following method is used for finding analysed feed stuffs for a certain user that has been edited in a certain period of time.

Name: FindAnalysedFeedStuffs	
Input:	Username, Password, StartDate. EndDate (optional), ParameterID[] (optional)
Description:	Finds all analysed feed stuffs that are owned by the user and that was updated last time between StartDate and EndDate. EndDate may be null. Then no upper date limit is used. Only the parameter values corresponding to the IDs in the ParameterID[] array will be returned. If the array is not given all the parameter values will be returned.
Output:	An **AnalysedFeedStuff** data set.
Exceptions:	*AccessDeniedException* if the user/password is invalid.

4.4.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to delete an existing analysed feed stuff:

```
// Instantiate the Web-service proxy class
FAS wsFas = new FAS();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
DateTime startDate = new DateTime(2007, 1, 1);
DateTime? endDate = new DateTime(2008, 1, 1);

// Call the web-method to find the analysed feed stuffs of the user
that is updated last time in 2007
wsFas.FindAnalysedFeedStuffs(username, password, startDate, endDate,
null);
```

4.4.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to delete an existing analysed feed stuff:

```
// Instantiate the Web-service proxy class
FASNonDotNet wsFAS = new FASNonDotNet();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
DateTime startDate = new DateTime(2007, 1, 1);
DateTime? endDate = new DateTime(2008, 1, 1);

// Call the web-method to find the analysed feed stuffs of the user
that is updated last time in 2007
wsFas.FindAnalysedFeedStuffs(username, password, startDate, endDate,
null);
```

4.5 Get Analysed Feed Stuff

The following method is used for getting a certain analysed feed stuff for a certain.

Name: GetAnalysedFeedStuff	
Input:	Username, Password, FeedStuffID, ParameterID[] (optional)
Description:	Gets the analysed feed stuff with the given FeedStuffID. Only the parameter values corresponding to the IDs in the ParameterID[] array will be returned. If the array is not given all the parameter values will be returned.
Output:	An **AnalysedFeedStuff** data set.
Exceptions:	*AccessDeniedException* if the user/password is invalid. *AnalysisFeedStuffNotFoundException* if no Analysed FeedStuff exists with the given FeedStuffID. *LabNotAuthorizedException* if the user is not authorized to retreive the given AnalysedFeedStuff dataset because it belongs to another Laboratory.

4.5.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to delete an existing analysed feed stuff:

```
// Instantiate the Web-service proxy class
FAS wsFas = new FAS();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
long feedStuffID = 120;

// Call the webmethod to get an certain analysed feed stuff
wsFas.GetAnalysedFeedStuff(username, password, feedStuffID, null);
```

4.5.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to delete an existing analysed feed stuff:

```
// Instantiate the Web-service proxy class
FASNonDotNet wsFAS = new FASNonDotNet();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
long feedStuffID = 120;

// Call the webmethod to get an certain analysed feed stuff
wsFas.GetAnalysedFeedStuff(username, password, feedStuffID, null);
```

4.6 Basic Data

Basic data is made up of several lists that contain the allowed values for some of the attributes of the Analysed feedstuffs, e.g. the allowed parameters or the allowed processing. The lists can be used as pick lists

when the laboratory is creating the analysed feed stuff for input to the CalculateAnalysedFeedStuff WebMethod.

Name: GetFASBasicData	
Input:	Username, Password, Language (optional)
Description:	Returns lists of allowed FSTParameters, FSTParameterTypes, Processing, Origins, Feed Groups, Countries, Languages, etc. in the NorFor system. The lists that are language specific are returned in the specified language. If no language is specified or the specified language is invalid, English is used. The possible languages are given in the Language table of the FASBasicData dataset. At the moment the possible values are 'DK', 'IS', 'NO', 'SE', 'UK'.
Output:	A **FASBasicData** data set.
Exceptions:	*AccessDeniedException* if the user/password is invalid.

4.6.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to receive FAS Basic Data from the web-service:

```
// Instantiate the Web-service proxy class
FAS wsFas = new FAS();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
string language = "SE"; //SE = Swedish

// Call the web-method
FASBasicDataDataset dsFASBasicData =
    wsFas.GetFASBasicData(username, password, language);

// Access data
string dkLanguageName =
    dsFASBasicData.Language.FindByLanguageID("DK").Name;
```

4.6.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to receive FAS Basic Data from the web-service:

```
// Instantiate the Web-service proxy class
FASNonDotNet wsFAS = new FASNonDotNet();

// Initialize the parameters
string username = "MyUserName";
string password = "MyPassword";
string language = "SE"; //SE = Swedish
```

```
// Call the web-method
FASBasicDataDataset dsFASBasicData =
    wsFAS.GetFASBasicData(username, password, language);

// Access data
string dkLanguageName =
   Array.Find<Language>(dsFASBasicData.Language,
   delegate(Language language) {
       return language.LanguageID == "DK";
   }
   ).Name;
```

4.7 Change Password

Name: ChangePassword	
Input:	Username, old_password, new_password
Description:	Verifies that the user's current password is old_password. Then it changes the user's password to new_password.
Output:	Nothing. If the method returns normally, the operation succeeded. An exception is thrown if the operation failed.
Exceptions:	*AccessDeniedException* if the username/old_password is invalid.

4.7.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to change the Lab administrators password using the web-service:

```
// Instantiate the web-service proxy
FAS wsFas = new FAS();

// Call web-method to change password
wsFas.ChangePassword("MyUserName", "MyPassword", "MyNewPassword");
```

4.7.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to change the Lab administrators password using the web-service:

```
// Instantiate the web-service proxy
FASNonDotNet wsFAS = new FASNonDotNet();

// Call web-method to change password
wsFAS.ChangePassword("MyUserName", "MyPassword", "MyNewPassword");
```

4.8 Get NorFor Feedstuff

Name: GetNorForFeedStuff	
Input:	Username, Password, Formatted_FeedStuffID,

	ParameterID[] (optional).
Description:	This method returns a NorFor feed stuff as referenced by the given Formatted_FeedStuffID. All the returned parameter values will be set to 'Standard' in origin. Only the parameter values corresponding to the IDs in the ParameterID[] array will be returned. If the array is not given all the parameter values will be returned. **Note:** The NorFor feed stuff can not be edited.
Output:	A **NorForFeedStuff** data set. If the feed stuff does not exists then an empty data set is returned.
Exceptions:	*InvalidTokenException* if the Token is invalid.

4.8.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to receive a specific NorFor feed stuff from the web-service:

```
// Instantiate the Web-service proxy class
FAS wsFas = new FAS();

// FormattedFeedstuffID of the NorFor feedstuff
string formattedID = "001-0001";

// Parameter IDs to receive (ex. null = all, empty string[] = none)
string[] parameterIdArray = new string[] { "PS" };

// Call the web-method
NorForFeedStuffDataset dsNorForFeedStuff =
       wsFas.GetNorForFeedStuff("MyUserName", "MyPassword",
formattedID, parameterIdArray);

// Access data
string feedName =
    dsNorForFeedStuff.NorForFeedStuff[0].Name;

long feedStuffID =
    dsNorForFeedStuff.NorForFeedStuff[0].FeedStuffID;

double psValue =
dsNorForFeedStuff.NorForParameterValue.FindByFeedStuffIDFSTParameterID(
feedStuffID,"PS").Value;
```

4.8.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to receive a specific NorFor feed stuff from the web-service:

```
// Instantiate the Web-service proxy class
FASNonDotNet wsFAS = new FASNonDotNet();
```

```
// FormattedFeedstuffID of the NorFor feedstuff
string formattedID = "001-0001";

// Parameter IDs to receive (ex. null = all, empty string[] = none)
string[] parameterIdArray = new string[] { "PS" };

// Call the web-method
NorForFeedStuffDataset dsNorForFeedStuff =
       wsFas.GetNorForFeedStuff("MyUserName", "MyPassword",
formattedID, parameterIdArray);

// Access data
string feedName =
    dsNorForFeedStuff.NorForFeedStuff[0].Name;

long feedStuffID =
    dsNorForFeedStuff.NorForFeedStuff[0].FeedStuffID;

double psValue =
     Array.Find<NorForParameterValue>(dsNorForFeedStuff.NorForParameterValue,
     delegate(NorForParameterValue npv) {
         return npv.FeedStuffID == feedStuffID && npv.FSTParameterID == "PS";
     }
     ).Value;
```

4.9 Find NorFor Feedstuff

Name: FindNorForFeedStuff	
Input:	Username, Password, **SearchCriteria** data set, ParameterID[] (optional).
Description:	Searches for NorFor feed stuffs. The search criteria may limit the search to interval, feed names, etc. Returns a list of all feed stuffs found. The list may be empty if none was found. Besides the general feed stuff attributes, only the parameter values corresponding to the IDs in the ParameterID[] array will be returned. If the array is not given all the parameter values will be returned. **Note:** The NorFor feed stuff can not be edit.
Output:	A **NorForFeedStuff** data set containing all the found feed stuffs. The feed stuffs are sorted according to: group, code.
Exceptions:	*InvalidTokenException* if the Token is invalid. *DatasetStructureException* if the **SearchCriteria** data set is not structured correctly. *SearchCriteriaValueInvalidException* if a value in a search criteria is invalid

4.9.1 Code sample .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to search for NorFor feed stuffs from the web-service:

```
// Instantiate the Web-service proxy class
FAS wsFas = new FAS();

// FormattedFeedstuffID of the NorFor feedstuff
string formattedID = "001-0001";

// Parameter IDs to receive (ex. null = all, empty string[] = none)
string[] parameterIdArray = new string[] { "PS" };

// Create a SearchCriteria
// Find all feed stuffs where feedcode <= 10
SearchCriteriaDataset dsSearchCriteria = new SearchCriteriaDataset();
dsSearchCriteria.SearchCriteria.AddSearchCriteriaRow(1, 6,-1, "Kode",
"integer", "<|<=|=|>=|>|<>","","","",0,9999,"","<=",0,0,"10");

// Note: to find all NorFor feed stuffs don't add any rows to the
// SearchCriteriaDataset.

// Call the web-method
NorForFeedStuffDataset dsNorForFeedStuff =
       wsFas.FindNorForFeedStuff("MyUserName", "MyPassword",
dsSearchCriteria, parameterIdArray);

// Access data
foreach (NorForFeedStuffDataset.NorForFeedStuffRow fsRow in
dsNorForFeedStuff.NorForFeedStuff)
{
        string feedName =
           fsRow.Name;

        long feedStuffID =
            fsRow.FeedStuffID;
}
```

4.9.2 Code sample non .Net

Assuming that a web-service proxy has been generated from the wsdl, the following code shows how to search for NorFor feed stuffs from the web-service:

```
// Instantiate the Web-service proxy class
FASNonDotNet wsFAS = new FASNonDotNet();s

// FormattedFeedstuffID of the NorFor feedstuff
string formattedID = "001-0001";

// Parameter IDs to receive (ex. null = all, empty string[] = none)
string[] parameterIdArray = new string[] { "PS" };

// Create a SearchCriteria
// Find all feed stuffs where feedcode <= 10
SearchCriteria[] dsSearchCriteria = new SearchCriteria[1];
dsSearchCriteria[0] = new SearchCriteria();
dsSearchCriteria[0].SearchID = 1;
dsSearchCriteria[0].Number = 6;
dsSearchCriteria[0].TextID = -1;
```

```
dsSearchCriteria[0].Name = "Kode";
dsSearchCriteria[0].Datatype = "integer";
dsSearchCriteria[0].ValidOperations = "<|<=|=|>=|>|<>";
dsSearchCriteria[0].ValidateMin = 0;
dsSearchCriteria[0].ValidateMax = 9999;
dsSearchCriteria[0].AndOrOperator = "";
dsSearchCriteria[0].Operation = "<=";
dsSearchCriteria[0].ParenthesisStart = 0;
dsSearchCriteria[0].ParenthesisEnd = 0;
dsSearchCriteria[0].Value = "10";

// Note: to find all NorFor feed stuffs don't add any rows to the
// SearchCriteriaDataset.

// Call the web-method
NorForFeedStuffDataset dsNorForFeedStuff =
       wsFas.FindNorForFeedStuff("MyUserName", "MyPassword",
dsSearchCriteria, parameterIdArray);

// Access data
foreach (NorForFeedStuff fsRow in dsNorForFeedStuff.NorForFeedStuff)
{
        string feedName =
           fsRow.Name;

        long feedStuffID =
            fsRow.FeedStuffID;
}
```

5 Data specifications

The chapter shows the details of the datasets that are used as parameters by the WebMethods described in the previous section. It is based on the document "*FAS_FST_Parameters And Calculations*" produced by NorFor.

5.1 Overview

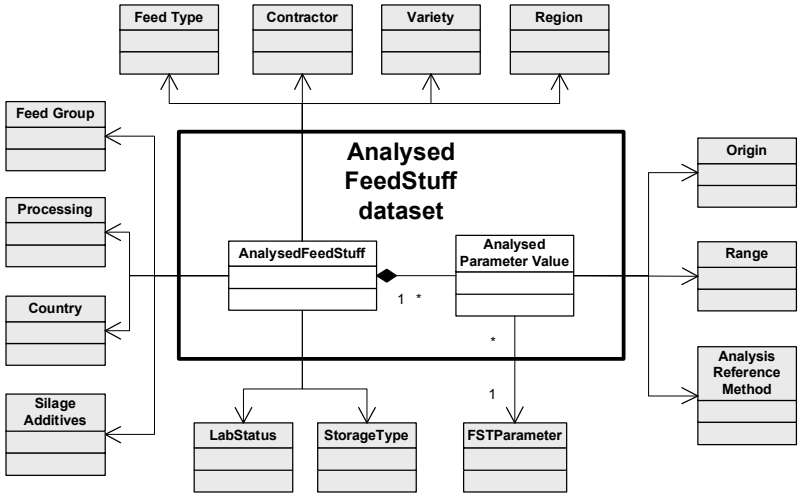


Figure 4 The AnalysedFeedStuff dataset. The grey tables are not included in the dataset.

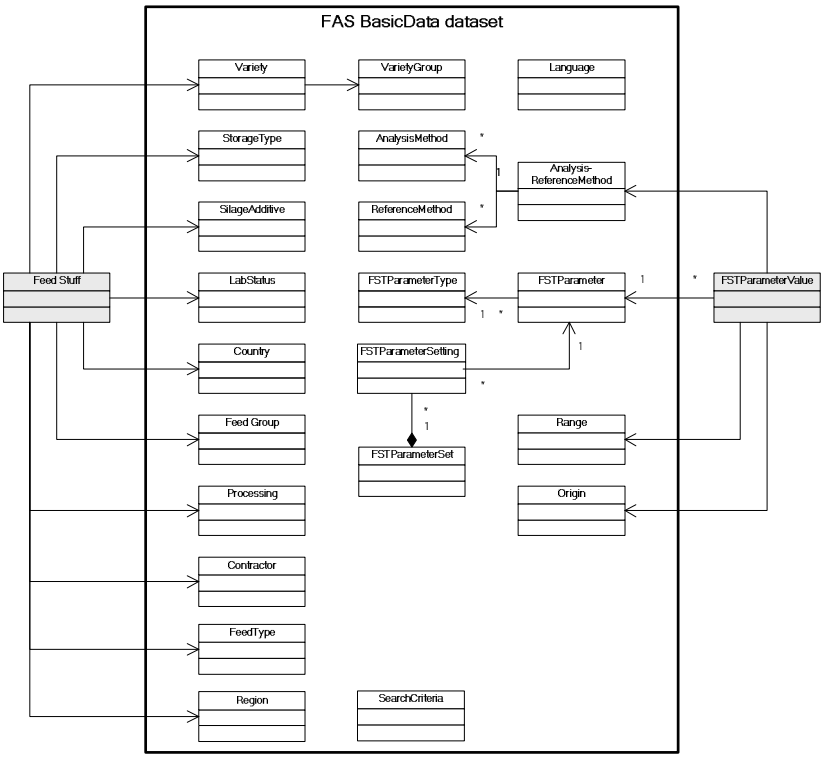


Figure 5 The BasicData Dataset. The grey tables are not included in the dataset.

Figure 6 The SearchCriteria Dataset.

5.2 Search criteria

The SearchCriteria dataset requires a little more detailed description than the other datasets since it requires implementation of some business logic beyond what is readily understandable from looking at the data attributes.

The search mechanism is designed to work the following way:

1. The client retrieves a populated SearchCriteria data set from the server as part of the basic data. The populated data set contains one of each of the available search criteria supported by the NorFor server. Not all columns contain data when retrieved from the server.

2. The client may use the information given in the retrieved search criteria to present search options to the user in a dynamically way, or build the search any other way desired. Standard searches can be hard coded into the client (e.g. a search that returns all feedstuffs on a given herd).

3. When the search must be performed, the client creates a new, empty SearchCriteria data set and populates it with the search criteria that should be part of the search. It is not necessary to return all the information originally retrieved from the server, but a few of the other columns must contain valid data.

4. The new, populated SearchCriteria data set is sent to the server as a parameter to the search method and the found feed stuffs are returned.

From this scenario it becomes obvious that the SearchCriteria dataset are used in two ways. When retrieved from the server it contains data that can be used by the client to build a dynamic search dialog. When sent to the server as parameter to the search methods, it contains only the data necessary for executing the search.

The following table shows each column in the SearchCriteria data set including a description. The SRV column will have an X if the column contains data when the dataset is retrieved from the server. The CLI column will have an X if the column must contain valid data when used as parameter to the search method by the client.

Field	Description	SRV	CLI
SearchCriteriaID	Obsolete. Was originally used as primary key in the database. You can ignore this.	X	
SearchID	Not used when searching for NorForFeedStuffs. Set to default value of 1.	X	
Number	This is the ID of the search criteria. The search criteria received in the BasicData contains the number to use for each of the different criteria available. A complete list is given in section '5.8.2' below the SearchCriteria table.	X	X
TextID	For internal use. Ignore this field.	X	

Name	When the search criteria dataset are retrieved from the server, this field contains the name of the search criteria in the language selected at login. This attribute can be used directly as label on the client GUI.	X	
DataType	The data type can be used if the client builds a dynamically search dialog. See description below this table.	X	
ValidOperations	Contains a list of valid comparison operations (separated by a | symbol). E.g. the string "=|Like" means that the client should return either a '=' or a 'LIKE' text string in the Operation field.	X	
ReferenceTable	If DataType is 'reference', then let the user select from this table from the BasicData dataset.	X	
ReferenceAttribute	If DataType is 'reference', then when the user selects an entry from the ReferenceTable table, return the value of this attribute in the Value field.	X	
DisplayAttribute	If DataType is 'reference', then this attribute from the ReferenceTable table is suitable for presenting in a drop-down list.	X	
ValidateMin	If DataType is 'integer', then the client may use this field for validation input before executing the search	X	
ValidateMax	If DataType is 'integer', then the client may use this field for validation input before executing the search	X	
AndOrOperator	If more than one search criteria is provided, then all but the last criterion must have either 'AND' or 'OR' in this field.		X
Operation	This attribute must contain the selected operation to use for this criterion. The operation must be included in the list given in the ValidOperations attribute.		X
ParenthesisStart	An integer containing the number of start parenthesis to place in front of this search criterion		X
ParenthesisEnd	An integer containing the number of end parenthesis to place after this search criterion.		X
Value	Must contain the value used in the search. This can be either a string, a reference (the value of the ReferenceAttribute), and integer or a Boolean		X

5.2.1 The DataType attribute

The DataType attribute can be used by the client if building a dynamic search dialog. The value of the DataType attribute indicates which kind of information is expected in the Value attribute when executing a search.

The DataType attribute can have any of the following values: "reference", "integer", "string", "date", "boolean" or "string:double". Here is a description of the different data types:

reference
This search criteria takes a reference to a table entry as the search

criteria value. Use the information in the ReferenceTable, ReferenceAttribute and DisplayAttribute to provide the user with valid selections.

If e.g. the ReferenceTable attribute contains the string "Owner", the ReferenceValue attribute contains the string "OwnerID" and the DisplayAttribute attribute contains the string "Name" then the Value attribute (when providing the new SearchCriteria data set to a search method) must contain a value from one of the OwnerID columns of the Owner table. The Name column of the Owner table can be presented as a pick-list to the user.

integer
The value of the SearchCriteria expects an integer in the Value attribute. Valid integers are in the range specified in the ValidateMin-ValidateMax attributes, but no errors are generated if not (but no results will be found).

string
The value of the SearchCriteria expects a string in the Value attribute.

date
The value of the SearchCriteria expects a date in the Value attribute. The date must be formatted like this: 2006-12-31. If the date is in another format an exception is thrown.

string:double
The value of the SearchCriteria expects that value is formatted in the following way: First a string followed by a colon, and finally a double.

boolean
The value of the SearchCriteria expects a string containing either "TRUE" or "FALSE" in the Value attribute.

5.2.2 Allowed SearchCriteria

The following table defines the currently defined searchcriteria numbers and if it is allowed to use it from FST and FAS. Only the ones with a mark in the FST column can be used when searching for feedcompany feedstuffs.

Number	Description	FST	FAS
1	Owner	X	X
2	Region	X	X
3	Organic	X	X
4	Feed Type	X	X
5	Feed Group	X	
6	Feed Code	X	

7	Sequence Number	X	
8	Feed group Name	X	
9	Feed type Name	X	
10	Feedstuff Name	X	
11	Harvest Date		X
12	Date Arrived Lab		X
13	Cutting Number		X
14	Silage Additive		X
15	Storage Type		X
16	Variety		X
17	Contractor		X
18	Country		X
19	Parameter Has Value		X
20	Parameter Value	X	X
21	Last Edited		X
22	Advisor Number		X

ParameterHasValue only finds the feedstuffs which has a value for the specified parametervalue. And ParameterValue makes it possible to narrow the search to feedstuffs where a specific parametervalue lies in a specified interval.

If any non-authorized SearchCriteria are used, an exception will be thrown.

5.3 Attributes of datatables

The tables in the following sections list the data attributes available for each data row in the tables contained in the data sets. The columns in the tables are used as follows:

- **PK/FK** - Indicates if the attribute is a primary key (PK) or a foreign key (FK) or both.
- **Column Name** – The name of the attribute.
- **Data Type** - Indicates which XML data type is used for the attribute in the soap message.
- **Length** – The data type width in bytes for simple types or storage capacity in characters for strings.

- **Allow null** – Marked with X if null value is allowed or may be expected. An (X) means that the field can be null in the dataset, but not in the database. The field will be set by the system.

- **Read only** – Marked with X if the client is not allowed to change the value; it must be returned as received. If the value is changed, the system either ignores the changes or may not be able to use the dataset at all.

- **Changed** – Specifies the version where the attribute is changed. If possible a description of the change is also provided. This column describes changes since NorFor 1.10 and is only shown if the table has any changes. The version specified is the version with the latest change (i.e. if an attribute is changed in multiple versions, only the last version will be shown).

5.4 Datatype specifications

The NorFor system uses the following datatypes: Boolean, int, long, double, string and datetime. The range for each datatype is the same as the normal datatypes in .Net with one exception. The valid range for the datetime is between 1/1/1753 12:00:00 AM and 12/31/9999 11:59:59 PM.

5.5 Analysed Feed Stuff

LabID and LabAnalysisID together constitute a unique identifier for an analysed feed stuff, but the Primary Key FeedStuffID is used internally in the NorFor system.

Table Name: AnalysedFeedStuff						
PK/ FK	Column Name	Data Type	Length	Allow null	Read only	Changed
PK	FeedStuffID	long	8		X	
FK	LabID	Long	8		X	
	LabAnalysisID	String	50			
	DateOfSampling	Datetime	8	X		1.11 allow null
	DateArrivedLab	Datetime	8			
	CostSampling	Double	8	X		
	CostAnalysis	Double	8	X		
FK	StatusID	Integer	4			
	LabComment	String	2000	X		
Feed Stuff Info						
FK	FeedGroupID	Long	8			

	FeedCode	Integer	4			
	SequenceNumber	Integer	4	X		
FK	RegionID	Long	8	(X)	X	
FK	FeedTypeID	Long	8	(X)	X	
FK	CountryID	String	2			
	HerdID	Long	8	X		
	FeedName	string	50	X		
	LocalRegionID	string	50	X		
Version						
	DateArrivedSystem	Datetime	8	(X)	X	
	LastEdited	Datetime	8	(X)	X	
	FSTversion	String	250	(X)	X	
	FRCversion	String	250	(X)	X	
	NorForLastEdited	Datetime	8	(X)	X	
Organic						
	Organic	Boolean	1	X		
	ConversionFeed	Boolean	1	X		
	OwnCrop	Boolean	1	X		
Harvest						
	HarvestDate	Datetime	8	X		
	CuttingNumber	Integer	4	X		
	Area	Double	8	X		
FK	ContractorID	Integer	4	X		
	LotNumber	Integer	4	X		
Structure						
FK	ProcessingID	Long	8	X		
Storage						
	StorageVolume	Double	8	X		
	StorageLength	Double	8	X		
	StorageWidth	Double	8	X		
	StorageHeight	Double	8	X		
	KgDryMatterPerM3	Double	8	X		
FK	StorageTypeID	Integer	4	X		
FK	SilageAdditiveID	Integer	4	X		

Crop						
FK	Variety1	Integer	4	X		
FK	Variety2	Integer	4	X		
	KgNPerCutting	Integer	4	X		
FK	VarietyGroup1ID	Integer	4	X		
FK	VarietyGroup2ID	Integer	4	X		
FK	VarietyGroup3ID	Integer	4	X		
FK	VarietyGroup4ID	Integer	4	X		
	VarietyGroup1Share	Double	4	X		
	VarietyGroup2Share	Double	4	X		
	VarietyGroup3Share	Double	4	X		
	VarietyGroup4Share	Double	4	X		
Advisor						
	AdvisorNumber	Integer	4	X		
	AdvisorComment	String	2000	X		
	UserCode	String	50	X		
Calculation						
	FillValueCorrection	Boolean	1	X		
	GrassSilageCPcalculation	Boolean	1	X		
	IVOScorrection	Boolean	1	X		
	EFOScorrection	Boolean	1	X		
	VOSCorrection	Boolean	1	X		
	UreaCorrection	Boolean	1	X		

Table Name: AnalysedParameterValue					
PK/FK	Column Name	Data Type	Length	Allow null	Read only
PK/FK	FeedStuffID	long	8		X
PK/FK	FSTParameterID	String	20		X
FK	OriginID	String	3		
FK	RangeID	Integer	4		
FK	MethodID	Integer	4	X	
	Method	String	100	X	
	StandardDeviation	Double	8	X	

	Value	Double	8		

5.6 Analysed Feed Stuff With Statistics

An AnalysedFeedStuffWithStatistics is like a AnalysedFeedStuff but just extended with some statistics data. Likewise for AnalysedParameterValueWithStatistics to the AnalysedParameterValue.

Table Name: AnalysedFeedStuffWithStatistics

PK/ FK	Column Name	Data Type	Length	Allow null	Read only	Changed
PK	FeedStuffID	long	8		X	
FK	LabID	Long	8		X	
	LabAnalysisID	String	50			
	DateOfSampling	Datetime	8	X		1.11 allow null
	DateArrivedLab	Datetime	8			
	CostSampling	Double	8	X		
	CostAnalysis	Double	8	X		
FK	StatusID	Integer	4			
	LabComment	String	2000	X		
Feed Stuff Info						
FK	FeedGroupID	Long	8			
	FeedCode	Integer	4			
	SequenceNumber	Integer	4	X		
FK	RegionID	Long	8	(X)	X	
FK	FeedTypeID	Long	8	(X)	X	
FK	CountryID	String	2			
	HerdID	Long	8	X		
	FeedName	string	50	X		
	LocalRegionID	string	50	X		
Version						
	DateArrivedSystem	Datetime	8	(X)	X	
	LastEdited	Datetime	8	(X)	X	
	FSTversion	String	250	(X)	X	
	FRCversion	String	250	(X)	X	
	NorForLastEdited	Datetime	8	(X)	X	

Organic						
	Organic	Boolean	1	X		
	ConversionFeed	Boolean	1	X		
	OwnCrop	Boolean	1	X		
Harvest						
	HarvestDate	Datetime	8	X		
	CuttingNumber	Integer	4	X		
	Area	Double	8	X		
FK	ContractorID	Integer	4	X		
	LotNumber	Integer	4	X		
Structure						
FK	ProcessingID	Long	8	X		
Storage						
	StorageVolume	Double	8	X		
	StorageLength	Double	8	X		
	StorageWidth	Double	8	X		
	StorageHeight	Double	8	X		
	KgDryMatterPerM3	Double	8	X		
FK	StorageTypeID	Integer	4	X		
FK	SilageAdditiveID	Integer	4	X		
Crop						
FK	Variety1	Integer	4	X		
FK	Variety2	Integer	4	X		
	KgNPerCutting	Integer	4	X		
FK	VarietyGroup1ID	Integer	4	X		
FK	VarietyGroup2ID	Integer	4	X		
FK	VarietyGroup3ID	Integer	4	X		
FK	VarietyGroup4ID	Integer	4	X		
	VarietyGroup1Share	Double	4	X		
	VarietyGroup2Share	Double	4	X		
	VarietyGroup3Share	Double	4	X		
	VarietyGroup4Share	Double	4	X		
	AboveSeaLevel	Int	4	X		1.19.1
Advisor						
	AdvisorNumber	Integer	4	X		

	AdvisorComment	String	2000	X		
	UserCode	String	50	X		
Calculation						
	FillValueCorrection	Boolean	1	X		
	GrassSilageCPcalculation	Boolean	1	X		
	IVOScorrection	Boolean	1	X		
	EFOScorrection	Boolean	1	X		
	VOSCorrection	Boolean	1	X		
	UreaCorrection	Boolean	1	X		
Statistics						
	StatisticsCalculated	DateTime	X	X		
	StatisticsType	String	24	X		
	StatisticsHarwestYear	Int	4	X		

Table Name: AnalysedParameterValueWithStatistics					
PK/FK	Column Name	Data Type	Length	Allow null	Read only
PK/FK	FeedStuffID	long	8		X
PK/FK	FSTParameterID	String	20		
FK	OriginID	String	3		
FK	RangeID	Integer	4		
FK	MethodID	Integer	4	X	
	Method	String	100	X	
	StandardDeviation	Double	8	X	
	Value	Double	8		
	NumberOfAnalysis	Integer	4	X	
	Average	Double	8	X	
	Deviation	Double	8	X	
	Percentile10Pct	Double	8	X	
	Percentile90Pct	Double	8	X	

5.7 NorFor Feed Stuff

FeedGroup and FeedCode together constitute a unique identifier (FormattedFeedStuffID) for a norfor feed stuff, but the Primary Key FeedStuffID is used internally in the NorFor system.

Table Name: NorForFeedStuff					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	FeedStuffID	long	8		X
	FormattedFeedStuffID	string	12	(X)	X
FK	FeedGroupID	long	8		X
	FeedCode	int	4		X
FK	FeedTypeID	long	8		X
FK	RegionID	long	8		X
FK	CountryID	string	2		X
FK	TextID	long	8		X
	Name (Language specific)	string	50		X

FK	DescriptionTextID	long	8		X
	SecretAnalysis	boolean	1		X
	LastEdited	dateTime	8	(X)	X
FK	ProcessingID	long	8		X
	Organic	boolean	1		X
	FillValueCorrection	boolean	1		X
	GrassSilageCPcalculation	boolean	1		X
	IVOScorrection	boolean	1		X
	EFOScorrection	boolean	1		X
	VOScorrection	boolean	1		X
	UreaCorrection	boolean	1		X
	CreateDate	datetime	8		X
	Note1	string	50	X	X
	Note2	string	50	X	X
	Note3	string	50	X	X

Table Name: NorForParameterValue					
PK/FK	Column Name	Data Type	Length	Allow null	Read only
PK/FK	FeedStuffID	long	8		X
PK/FK	FSTParameterID	string	20		X
FK	OriginID	string	3		
	NoOfAnalysis	Int	4	X	X
	Deviation	double	8	X	X
	Value	double	8		X

5.8 Basic Data

The Basic Data contains datasets that are used as choice-lists. E.g. the Parameter dataset contains the list of possible parameters in the system.

5.8.1 Picklists for AnalysedFeedStuff

Table Name: LabStatus					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only

PK	StatusID	Integer	4		X
	Name	String	50		X

Table Name: FeedGroup					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	FeedGroupID	Long	8		X
FK	OwnerID	Long	8		X
	Name (Language specific)	String	50		X
	SortOrder	Integer	4		X
	Deleted	Boolean	1		X

Table Name: FeedType					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	FeedTypeID	long	8		X
FK	FeedGroupID	long	8		X
FK	TextID	long	8		X
	Name (Language specific)	string	50		X
	Deleted	boolean	1		X

Table Name: Country					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	CountryID	String	2		X
	Name	String	50		X

Table Name: Contractor					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	ContractorID	Integer	4		X
	Name	String	100		X
FK	CountryID	String	2		X

Table Name: Processing					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	ProcessingID	Long	8		X
	Name (Language specific)	String	50		X
	ParticleSize	Integer	4		X
	ValidationMin	Integer	4		X
	ValidationMax	Integer	4		X

Table Name: SilageAdditive					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	SilageAdditiveID	Integer	4		X
	Name	String	50		X
FK	CountryID	String	2		X

Table Name: StorageTypes					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	StorageTypeID	Integer	4		X
	Name (Language specific)	String	50		X

Table Name: Variety					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	VarietyID	Integer	4		X
FK	VarietyGroupID	Integer	4		X
	Name	String	50		X
FK	CountryID	String	2		X

5.8.2 SearchCriteria

Table Name: SearchCriteria					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only

PK	SearchCriteriaID	long	8	X	X
	SearchID	int	4	X	
	Number*	int	4		X
FK	TextID	long	8	X	X
	Name (Language specific)	string	50	X	X
	Datatype	string	20	X	X
	VaildOperations	string	100	X	X
	ReferenceTable	string	30	X	X
	ReferenceAttribute	string	30	X	X
	DisplayAttribute	string	30	X	X
	ValidateMin	double	8	X	X
	ValidateMax	double	8	X	X
	AndOrOperator	string	3		
	Operation	string	20		
	ParenthesisStart	int	4		
	ParenthesisEnd	int	4		
	Value	string	100		

* See section 5.2.2 for a description of the allowed numbers.

5.8.3 Picklists for FSTParameterValue

Table Name: FSTParameter					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	FSTParameterID	String	20		X
FK	FSTParameterTypeID	Long	8		X
	Name (Language specific)	String	50		X
	ShortName(Language specific)	String	10		X
	Unit (Language specific)	String	50		X
	StandardValue	Double	8		X
	ValidationMin	Double	8		X
	ValidationMax	Double	8		X
	Editable	Boolean	1		X
	SortOrder	Integer	4		X
	NonAdditive	Boolean	1		X

	SecretParameter	Boolean	1		X
	DecimalPlaces	Integer	4		X
	ParameterCalculation	String	20	X	X
	DerivedCalculations	String	50	X	X
	MixtureCalculation	String	20	X	X

Table Name: Origin					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	OriginID	String	3		X
	Name (Language specific)	string	50		X

Table Name: Range					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	RangeID	Integer	4		X
	Name	String	20		X

An AnalysedParameterValue can be associated with an analysis method and a reference method through the AnalysisReferenceMethod. The AnalysisReferenceMethod table contains the many-to-many relation between AnalysisMethod and ReferenceMethod.

Table Name: AnalysisReferenceMethod					
PK/FK	Column Name	Data Type	Length	Allow null	Read only
PK	MethodID	Integer	4		X
FK	AnalysisMethodID	Integer	4		X
FK	ReferenceMethodID	Integer	4		X

5.8.4 Other

Language is a special choice-list. It is only used to specify in which language the other choice-lists should be presented.

Table Name: Language					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only

PK	LanguageID	String	2		X
	Name	String	50		X

VarietyGroup is a grouping of the varieties. It is not used directly when handling the analysed feed stuffs, but can be used as heading for the varieties if a laboratory wants to print out the list of varieties.

Table Name: VarietyGroup					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	VarietyGroupID	Integer	4		X
	Name (Language specific)	String	50		X

The parameters are grouped in types. The list FSTParameterType shows this grouping.

Table Name: FSTParameterType					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	FSTParameterTypeID	Long	8		X
	Name (Language specific)	String	50		X
	SortOrder	Integer	4		X

FSTParameterSet and FSTParameterSetting define together some lists that limit the list of parameters for different usages. E.g. the list FASParameters is the list of parameters that is relevant for laboratories.

Table Name: FSTParameterSet					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	SetName	String	30		X
	Name (Language specific)	String	50		X

Table Name: FSTParameterSetting					
PK/FK	Column Name	Data Type	Length	Allow null	Read only
PK/FK	SetName	String	30		X
PK/FK	FSTParameterID	String	20		X

	Standard	Boolean	1		X
	SortOrder	Integer	4		X

Table Name: AnalysisMethod					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	AnalysisMethodID	Integer	4		X
	Name	String	50		X
	Description	String	100	X	X

Table Name: ReferenceMethod					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	ReferenceMethodID	Integer	4		X
	Name	String	50		X
	Description	String	100	X	X

Table Name: Region					
PK/ FK	Column Name	Data Type	Length	Allow null	Read only
PK	RegionID	long	8		X
FK	TextID	long	8		X
	Name (Language specific)	string	50		X

6 Exceptions

If something went wrong during a call to a web method, an exception will be thrown indicating the nature of the error. The exceptions will be wrapped in a standard Soap Envelope containing a `<soap:Fault>` element and returned instead of the expected return value from the method. An example of such an exception is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>

<faultstring>System.Web.Services.Protocols.SoapException:
InvalidToken
   at NorFor.BLL.Distribution.ThrowSoapException(String type,
Object data) in
c:\sandbox\norfor\src\norfor\bll\distribution.cs:line 296
   at NorFor.BLL.Distribution.IsTokenValid(TokenID token) in
c:\sandbox\norfor\src\norfor\bll\distribution.cs:line 26
   at NorFor.WS_Client.Client.GetLanguages(String token) in
c:\sandbox\norfor\src\norfor\ws_client\client.asmx.cs:line
454</faultstring>
      <faultactor>WS_Client</faultactor>
      <detail>
        <exception>SystemError</exception>
        <technicalDescription>Xx xx</technicalDescription>
        <message>Xxxx xxx xxxx</message>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

The text shown in bold is the `<detail>` section. The detail section contains the information that the client needs in order to handle the exception.

Different exceptions have different detail sections. All detail sections contains an `<exception>` tag which provides the name of the exception. The other tags vary according to the type of exception.

Below is an exhaustive list of the exceptions that can be returned by the FAS system. Only the detail section is shown since the rest of the exception is as shown above.

6.1.1 SystemErrorException

This Exception can be thrown by all methods. It will be thrown when a system failure is encountered e.g. if no connection could be established to the database. Since it can be thrown by all methods, it is not listed under the Exceptions description for each method. The detail section:

```
<detail>
    <exception>SystemError</exception>
    <technicalDescription>Xxxx xx xxxx.</technicalDescription>
    <message>Xxxx xx xxxx xxx x xxxx x.</message>
</detail>
```

The `<technicalDescription>` element contains a technical description of the problem. This description is not language specific since it will be a copy of the description taken from the exception from the system or database. It can be used by technical support to better understand the reason for the error.

The `<message>` element contains a language specific description of the problem in a form that can be directly presented to the user. Examples could be: "A database error occurred. If the problem recurs contact your system administrator."

6.1.2 ServerIsLockedException

This Exception can be thrown by all methods. It will be thrown when the server is locked e.g. if a recalculation is performed or the software is upgraded. Since it can be thrown by all methods, it is not listed under the Exceptions description for each method. The detail section:

```
<detail>
    <exception>ServerIsLocked</exception>
    <reason>Xxxx xx xxxx xxx x xxxx x.</reason>
    <message>Xxxx xx xxxx.</message>
</detail>
```

The `<reason>` element contains a specific explanation of why the server is locked. This message is not language specific because it is entered directly by the administrator that locked the server.

The `<message>` element contains a general language specific description of why the server is locked. This could be a text explaining that the server is locked because the software is upgraded, or that the feedstuffs are being recalculated.

6.1.3 NorForFeedStuffNotFoundException

Exception thrown if the specified group and code identifies a NorFor feed stuff, which is not known in the system. The detail section:

```
<detail>
    <exception>NorForFeedStuffNotFound</exception>
    <norforFeedStuffId>xxx-yyyy</norforFeedStuffId>
    <feedStuffDBRef>n</feedStuffDBRef>
    <message>Xxxx xx xxxx.</message>
</detail>
```

The <norforFeedStuffID> element contains the id for which no NorFor feed stuff could be located. Used when the system was unable to find a feedstuff based on the formatted feedstuff ID (group+code).

The <feedstuffDBRef> element contains a database primary key value of the feedstuff. Used when the system was unable to find a feedstuff based on the database primary key.

The <message> element contains a general language specific description of which NorFor feed stuff that could not be found, in a form that can be directly presented to the user. The message contains the feed stuff name and FormattedFeedStuffID.

NOTE: Either the <norforFeedStuffId> or the <feedstuffDBRef> element is given. When the one is given, the other is empty.

6.1.4 DatasetStructureException

This exception is thrown if a dataset given as input parameter to a web method is not structured correctly. This will be due to an XML schema validation error. The detail section:

```
<detail>
    <exception>DatasetStructure</exception>
    <dataset>XxxxxXxxxxxXxx</dataset>
    <errorDescription>Xxxx xx xxxxx xxx xx</errorDescription>
    <message>Xxxx xx xxxx.</message>
</detail>
```

The <dataset> element contains the name of the dataset that was not structured correctly. For FAS it can only be "AnalysedFeedStuff".

The <errorDescription> element contains a technical description of the problem. This description is not language specific since it will be a copy of the description taken from the exception from the XML parser. It can be used during development to better understand the reason for the error.

The <message> element contains a general language specific description of the dataset structure error and the concerning dataset name, in a form that is technical but can be directly presented to the user.

6.1.5 SearchCriteriaValueInvalidException

This exception is thrown if one of the search criteria values given to one of the find methods are invalid.

The search criteria values are transferred as strings, but must be structured correctly. E.g. boolean values must be "TRUE" or "FALSE", numeric values must be numeric characters only. The detail section:

```
<detail>
    <exception>SearchCriteriaValueInvalid</exception>
    <criteriaId>n</criteriaId>
    <value>Xxxxxx</value>
    <message>Xxxx xx xxxx xxxxxx xx xxxxx xxxx.</message>
</detail>
```

The `<criteriaId>` element contains the id of the search criteria in error.

The `<value>` element contains the value given in the search criteria to the web method.

The `<message>` element contains a language specific description of the problem in a form that can be directly presented to the user. Since search criteria may change over time, a complete list of messages will not be given, but examples could be: "Sequence number must be numerical", "Organic must be either TRUE or FALSE", "Sequence number must be between 1 and 999 both included"

It makes sense to handle some of the exceptions in code, so the search criteria id is also provided allowing the client software to make a qualified decision as how to handle the exception.

6.1.6 CalculationFailedException

This exception will be thrown if the process of calculating calculated parameter values of a feed stuff fails. Reasons could be mathematical impossibilities such as divide by zero.

The cause would likely be related to errors in the formulas used in the calculations or missing validation information on parameter values (e.g. allowing negative values for parameters which may cause calculation errors).

```
<detail>
    <exception>CalculationFailed</exception>
    <calculation>Xxxx</calculation>
    <message>Xxxx xx xxxx xxxxxx xx xxxxx xxxx.</message>
</detail>
```

The `<calculation>` element contains the name of the calculation that failed.

The <message> element contains a language specific description of the problem in a form that is technical but can be directly presented to the user. No specific exception situations are currently identified, but an example of message could be: "Calculation failed. Please se log entry #n".

6.1.7 AccessDenied

Thrown if the user is not allowed access to the system, or has entered a invalid password. The detail section:

```
<detail>
    <exception>AccessDenied</exception>
    <userName>Xxx xx</userName>
    <message>Xxx xx</message>
</detail>
```

The <userName> element contains the username of the user.

The <message> element contains a language specific description of the problem in a form that can be directly presented to the user.

6.1.8 LabNotAuthorizedException

Thrown if trying to save an AnalysedFeedStuff, which the user is not authorized access to, i.e. it does not belong to the laboratory of user. The detail section:

```
<detail>
    <exception>LabNotAuthorized</exception>
    <labId>Xxx xx</labId>
    <labIdInAnalysedFeed>Xxx xx</labIdInAnalysedFeed>
    <labAnalysisId>Xxx xx</labAnalysisId>
    <message>Xxx xx</message>
</detail>
```

The <labId> element contains the LabID of the user. <labIdInAnalysedFeed> and <labAnalysisId> contain the ids of the feed, that the user is not authorized to access.

The <message> element contains a language specific description of the problem in a form that can be directly presented to the user.

6.1.9 AnalysedParameterValueValidationException

This exception will be thrown in situations where the parameter values are invalid according to the min/max given for the Parameter. The detail section:

```
<detail>
```

```
    <exception>AnalysedParameterValueValidation</exception>
    <parameterID>n</parameterID>
    <labId>Xxx xx</labId>
    <labAnalysisId>Xxx xx</labAnalysisId>
    <message>Xxxx xx xxxx xxxxxx xx xxxxx xxxx.</message>
</detail>
```

The <parameterID> element provides the parameter id of the Parameter whose value failed validation.

The <labId> and <labAnalysisId> contain the ids of the AnalysedFeedStuff the parameter belongs to.

The <message> element contains a language specific description of the problem in a form that can be directly presented to the user. Examples could be: "Analysed parameter value is not within the allowed interval".