



INTRODUCING LETSGROW SERVICES

PATRICK KALKMAN

SEPTEMBER 2009

This document is based on the 2009 release of LetsGrow Services.

CONTENTS

An Overview of LetsGrow Services	3
Security	4
LetsGrow Data.....	5
LetsGrow Services Interface.....	6
Using LetsGrow Services	8
Creating a Meta Data Retrieval Application.....	8
Creating a Data Retrieval Application	9
Creating a Data storage application	10
Conclusions	11
For Further Reading	11
About the author	11

AN OVERVIEW OF LETSGROW SERVICES

LetsGrow Services is part of the LetsGrow.com platform. LetsGrow Services enable external businesses to seamlessly integrate their information systems with LetsGrow. LetsGrow has been collecting data from different brands of climate computers and plant sensors for over five years. This wealth of information can be used and integrated into custom applications by using LetsGrow Services.

The goal of LetsGrow Services is to provide this data integration in a secure, stable and flexible way. Figure 1 illustrates this.

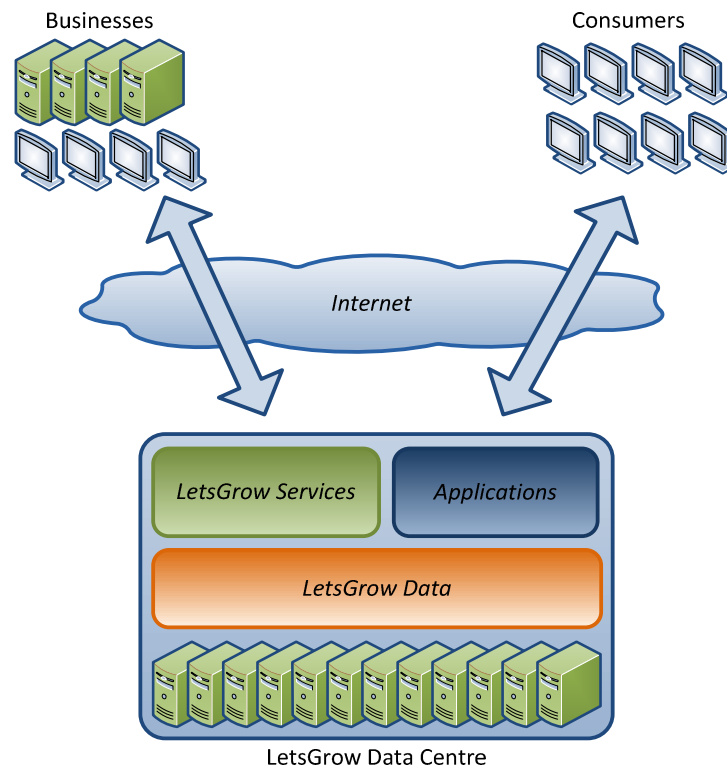


Figure 1: LetsGrow Services is part of the LetsGrow platform

As the figure shows LetsGrow Services run on machines in the LetsGrow Data Centre. Rather than providing software to connect to LetsGrow which should be installed on the customer's computer, LetsGrow Services provide services through standardized interfaces. Businesses can create applications that consume these services and provide new services to their customers. Here are some examples of the kinds of applications that might be built using LetsGrow Services:

- A research institute or university could combine the large amount of historically data that is available in LetsGrow. This may lead to better understanding and insights in the growing process that can for example help reduce the use of energy in greenhouses. Note that the data in LetsGrow is owned by the customers that have provided this data. Explicit consent of the customers is necessary to be able to use this data.
- A horticultural advisor of a customer could extract the data of the customer and perform additional calculations on the data. The results of these calculations could be stored in LetsGrow using LetsGrow Services. The original data and results of the calculation can be seen in one view by the customers.

- o Affiliates or partners of LetsGrow could create modules that provide additional features based on LetsGrow data. This data could lead for example to a model that predicts the flowering date of a crop. These additional modules may even lead to automatic changes in the greenhouse settings to aim for a specific flowering date. These additional modules could be offered to the customers of LetsGrow.

SECURITY

LetsGrow Services use multiple layers of security. These layers of security are different methods of security overlying one upon another like an onion to protect LetsGrow Services. Figure 2 shows the three different security layers of LetsGrow Services¹.

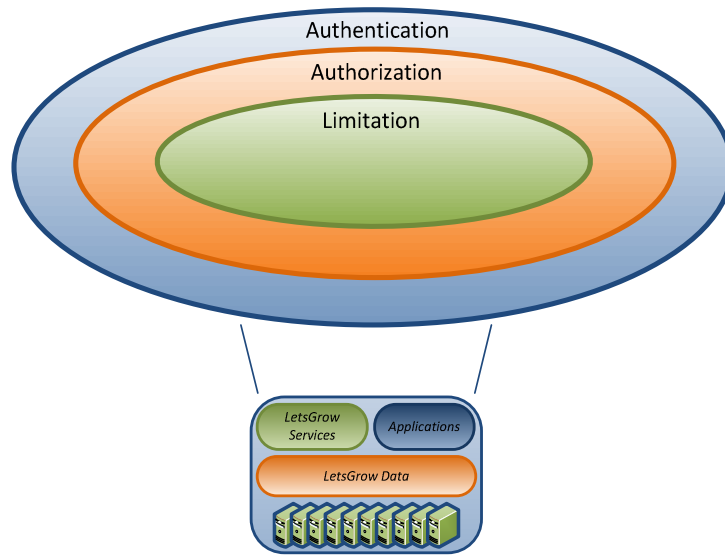


Figure 2 Security Layers of LetsGrow Services

The function of each layer is described below:

Authentication	This layer refers to the process where one entity verifies another entity's claim to holding a specific digital identity. LetsGrow Services uses a username password combination to authenticate a user. LetsGrow Services uses Windows Active Directory as the technical infrastructure for authentication.
Authorization	This layer refers to determining whether an access request from the entity shall be granted or rejected. A customer is identified using a username password in LetsGrow Services. This username password combination can only access the data that he or she has access to.
Limitation	This layer refers to the process of limiting the amount of data that can be retrieved or stored per time in LetsGrow Services. Currently this is limited by reading a maximum of 1 million samples per hour and mutating a maximum of 250 thousand samples per hour. By limiting this amount LetsGrow is able to provide the stability and performance expectations of their customers.

Only the authentication security layer is explicitly used when developing custom application using LetsGrow Services. The other layers are completely transparent.

¹ The firewall could be seen as the first layer making a total of four security layers.

LETSGROW DATA

This chapter describes how data is organized in LetsGrow. This is necessary to create the context in which the LetGrow Services interface is used.

Data in LetsGrow is organized around collectors. A collector is an entity that is described by several metadata attributes. A collector can contain an unlimited set of historical data. When historical data is requested through the LetsGrow Services interface the collector must be identified by the id of the collector. A subscription on LetsGrow Services includes a set of collectors or more specifically a set of collector ids. The LetsGrow applications use the term Item instead of Collector.

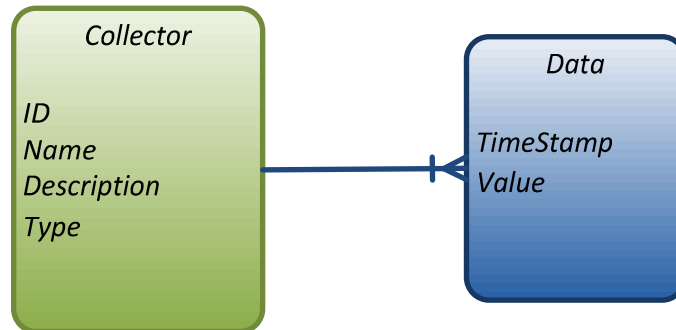


Figure 3 Data model of LetsGrow Data

Each collector has a number of data samples, each data sample consists of a timestamp identifying the date and time of the sample and the value itself. Reading and writing data via LetsGrow Services are two different processes. While data of all Collectors can be retrieved, data can only be written to collectors that are of type Hybrid². The user can change the type of any of its collectors to Hybrid through a service request of the service desk.

² Of course for both processes only the collectors that the user is authorized for can be accessed.

LETSGROW SERVICES INTERFACE

The interface of LetsGrow Services is exposed through a web service. This interoperable interface is described through WSDL and can be retrieved via the url <http://webservice.live.letsgrow.com/service.asmx>. The interface is described in text below.

```
1:  public interface ILetsGrowService
2:  {
3:      //Get one specific data sample of the given collector.
4:      double GetData(int id, DateTime time, int range);
5:
6:      //Write the data sample of the given collector.
7:      bool SetData(int id, DateTime utc, int utcOffset, double value);
8:
9:      //Retrieve a set of data from the given collector.
10:     ValueDataSet GetDataSet(int id, DateTime start, DateTime finish);
11:
12:     //Write the data set back to LetsGrow.
13:     void SetDataSet(ValueDataSet dataSet);
14:
15:     //Delete the range of samples of the given collector.
16:     void DeleteData(int id, DateTime start, DateTime finish);
17:
18:     //Retrieve all the meta data of all the authorized collectors.
19:     CollectorDataSet GetCollectors();
20:
21:     //Retrieve a single meta data attribute of the given collector.
22:     string GetMetaData(int id, MetaDataType type);
23: }
```

```
1:  //This typed data set represents a single sample.
2:  public class ValueDataSet
3:  {
4:      public int Id;
5:      public DateTime Utc;
6:      public int UtcOffset;
7:      public Double Value;
8:  }
```

```
1:  public enum MetaDataType
2:  {
3:      //Initial value
4:      None,
5:      //Retrieve the name of the collector
6:      Name,
7:      //Retrieve the name of the collector
8:      Description,
9:      //Retrieve the Item code of the collector
10:     ItemCode,
11:     //Retrieve the Device Item code of the collector
12:     DeviceItemCode,
13:     //Retrieve the Device Identity of the collector
14:     DeviceIdentity,
15:     //Retrieve date of the first sample of the collector
16:     FirstDate,
17:     //Retrieve date of the last sample of the collector
18:     LastDate,
19:     //Retrieve the total number of samples of the collector
```

```
20:     Count
21: }
```

The interface is explained in more detail in next chapter "Using LetsGrow Services". That chapter also includes sample applications.

USING LETSGROW SERVICES

Understanding the components of LetsGrow Services is important, but it is not enough to get started with developing custom applications. The best way to get acquainted with the platform is to walk through examples of how it can be used. This chapter looks at three scenarios for using LetsGrow Services: Creating a Meta Data Retrieval application, creating a Data Retrieval Application and creating a Data Storage Application.

All samples are implemented using the C# language and Microsoft Visual Studio 2008.

CREATING A META DATA RETRIEVAL APPLICATION

This sample application has the goal to retrieve the metadata of one of the collector lines of LetsGrow. As explained in "LetsGrow Data" a collector is described using meta data attributes and can contain a number of historical samples. This sample application retrieves the name of a specific collector and retrieved the total number of samples that are attached to this collector.

AUTHENTICATION AND AUTHORIZATION

Before being able to use the methods of the web service the credentials should be attached to the service reference. Below an example is shown on how to provide the credentials using Microsoft C#. The string "DemoUser" and "DemoPassword" should be replaced by the username and password received from LetsGrow. The Domain is fixed and should be set to "HiBlue.nl".

```
1: static void Main(string[] args)
2: {
3:     //A web reference LetGrowServices has been created through "Add Web Reference".
4:     using (LetsGrowServices.Service myService = new Service())
5:     {
6:         const string UserName = "DemoUser";
7:         const string Password = "DemoPassword";
8:         const string Domain = "HiBlue.nl";
9:         myService.Url = "http://webservice.live.letsgrow.com/service.asmx";
10:        myService.Credentials = new NetworkCredential(UserName, Password, Domain);
11:
12:        //Perform service request.
13:    }
14: }
```

RETRIEVING META DATA

The source code below uses the GetMetaData method to retrieve the name and the number of samples of a specific collector.

```
1: //Retrieve the name of the collector.
2: string name = myService.GetMetaData(1, MetaDataType.Name);
3: Console.WriteLine("The collector is named {0}.", name);
4:
5: //Retrieve the number of samples of the collector.
6: int numberOfSamples = Convert.ToInt32(myService.GetMetaData(1, MetaDataType.Count));
7: Console.WriteLine("The collector has {0} data samples", numberOfSamples);
```

CREATING A DATA RETRIEVAL APPLICATION

There are two methods of retrieving data from LetsGrow Services. The method `GetData` selects a single sample and the method `GetDataSet` retrieves a set of samples. For both methods a sample application will be described.

RETRIEVING A SINGLE SAMPLE

The method `GetData` retrieves a single value from a collector given the collectors id, a specific date-time and a range. The method selects the sample that is closest to the given date-time.

```
1: //Retrieve the value of one sample of collector with id = 1.
2: DateTime sampleDateTime = new DateTime(2009, 5, 10, 20, 0, 0);
3: int rangeInSeconds = 300;
4: int collectorId = 1;
5: double value = myService.GetData(collectorId, sampleDateTime, rangeInSeconds);
6: Console.WriteLine("The value of the sample is {0}", value);
```

The code snippet above retrieves the sample that is closest to 20:00 on the tenth of May in 2009. A range of 300 seconds is used. The `GetData` method looks for samples in the range from 19:55 until 20:05. For example in the scenario as show in Figure 4 the sample at 20:02 would be selected and the value 6.0 will be returned.

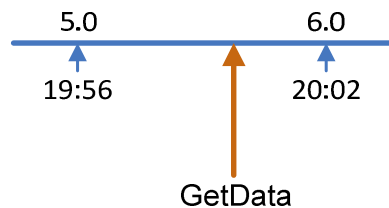


Figure 4 Single Sample Selection

RETRIEVING A SET OF SAMPLES

The method `GetDataSet` retrieves a set of samples from a given collector. The `GetDataSet` method has arguments for the start and end date. The return value is a `ValueDataSet` which is a typed `DataSet`. This `DataSet` is a .Net Framework specific type. To use the `ValueDataSet` in a different environment than Microsoft .Net a different approach is necessary. Look at the further reading section for more information.

```
1: //Retrieve the value of all the samples of collector with id = 1
2: //between 20:00 and 21:00.
3: DateTime start = new DateTime(2009, 5, 10, 20, 0, 0);
4: DateTime end = new DateTime(2009, 5, 10, 21, 0, 0);
5: int collectorId = 1;
6: ValueDataSet values = myService.GetDataSet(collectorId, start, end);
7: Console.WriteLine("There are {0} samples found.", values.Values.Count);
8: Console.WriteLine("The value of the 1st sample is {0}.", values.Values[0].Value);
```

The application above retrieves all the samples of collector with id 1 between 20:00 and 21:00 on the tenth of May in 2009.

CREATING A DATA STORAGE APPLICATION

As with retrieving data there are two methods of storing data via LetsGrow Services. The method `SetData` stores a single sample and the method `SetDataSet` stores a set of samples. For both methods a sample application will be described.

STORING A SINGLE SAMPLE

The method `SetData` stores a single sample in LetGrow. The sample code snippet stored a new sample of the collector with id 1.

```
1: //Write one sample of collector with id = 1
2: int collectorId = 1;
3: int utcOffset = 120;
4: DateTime sampleUtc = new DateTime(2009, 9, 3, 17, 5, 0);
5: myService.SetData(collectorId, sampleUtc, utcOffset, 17.3);
```

The value 17.3 is stored at 3 September 2009 UTC with a UTC offset of 120 minutes.

STORING A SET OF SAMPLES

The method `SetDataSet` stores a `ValueDataSet` in LetsGrow. The sample code snippet below creates a new `ValueDataSet` and fills it with two samples. It is possible to use samples of multiple collectors in one `ValueDataSet`. Each sample is created using a UTC `DateTime` and a UTC offset. Take a look at the [For Further Reading](#) chapter for more information about UTC.

```
1: //Write two samples of collector with id = 1
2: int collectorId = 1;
3: int utcOffset = 120;
4: ValueDataSet dataSet = new ValueDataSet();
5: DateTime first = new DateTime(2009, 9, 3, 17, 5, 0);
6: DateTime second = first + TimeSpan.FromMinutes(5);
7: dataSet.Values.AddValuesRow(collectorId, first, utcOffset, 17.0);
8: dataSet.Values.AddValuesRow(collectorId, second, utcOffset, 16.5);
9: myService.SetDataSet(dataSet);
```

The method `SetDataSet` is optimized for bulk inserts of large amounts of data.

CONCLUSIONS

Using the LetsGrow Services interface it is easy to develop custom applications. These custom applications can be developed and used by for example a research institute or a horticultural advisor. The large amount of historically data that is available in LetsGrow could be combined to provide new insights in horticultural processes. It is also easier for advisors of a LetsGrow customer to develop customer calculation models and simulations using the data of LetsGrow. In this way the advisor can add value for his customers by using existing data.

Third parties can create modules based on LetsGrow data. Rather than just combining and showing the data in a LetsGrow environment (website) it is also possible for these parties to create solutions under their own name.

FOR FURTHER READING

- LetsGrow Home Page
<http://www.letsgrow.com>
- Web Services and Datasets, about using Datasets in java environments
<http://msdn.microsoft.com/en-us/magazine/cc188755.aspx>
- Coordinated Universal Time
http://en.wikipedia.org/wiki/Coordinated_Universal_Time

ABOUT THE AUTHOR

Patrick Kalkman is Senior Software Architect at Hoogendoorn Growth Management (www.hoogendoorn.nl). From the main office in the Netherlands he develops high quality software solutions for the customers of LetsGrow and Hoogendoorn Growth Management. Patrick can be contacted at pka@hoogendoorn.nl.