



Intelligent Sensing Anywhere

# iCenter OS – Data Provision

## System Integration

*Ricardo Clérigo & André Oliveira*  
Software Engineering Department

Private  
Version 1.0

[www.isasensing.com](http://www.isasensing.com)

**DOCUMENT HISTORY**

---

<b>DATE</b>	<b>ISSUE</b>	<b>DESCRIPTION</b>	<b>AUTHOR</b>
01-09-2010	1.0	Document Creation	Ricardo Clérigo & André Oliveira

**TABLE OF CONTENTS**

---

- 1 Introduction ..... 1
- 2 Authentication and Common Structures ..... 2
  - 2.1 Authentication ..... 2
  - 2.2 Updating a password ..... 2
  - 2.3 Data Structures ..... 3
    - 2.3.1 Outcome of an operation ..... 3
    - 2.3.2 Consumption ..... 3
    - 2.3.3 Conversion ..... 4
    - 2.3.4 Data ..... 4
    - 2.3.5 Dataltem ..... 4
    - 2.3.6 DataltemAgg ..... 4
    - 2.3.7 Device ..... 5
    - 2.3.8 Tag ..... 5
    - 2.3.9 Unit ..... 5
  - 2.4 Obtaining arrays of structures ..... 5
- 3 Global Methods ..... 6
  - 3.1 Default values ..... 6
    - 3.1.1 Available Measures ..... 6
    - 3.1.2 Available Measurement Units ..... 6
- 4 Data ..... 8
  - 4.1 Obtain Consumptions ..... 8
  - 4.2 Bulk Insert of Data ..... 9
  - 4.3 Request a segmented Insertion ..... 9
  - 4.4 Segmented Bulk Insert of Data ..... 10
  - 4.5 Commit a Segmented Insertion ..... 11
  - 4.6 Rollback a Segmented Insertion ..... 11
  - 4.7 Removal of Data ..... 11
- 5 Hardware Management ..... 13
  - 5.1 Insert Unit ..... 13
  - 5.2 Edit Unit ..... 13
  - 5.3 Delete Unit ..... 13
  - 5.4 Insert Device ..... 14
  - 5.5 Edit Device ..... 14
  - 5.6 Delete Device ..... 15
  - 5.7 Insert Tag ..... 15
  - 5.8 Edit Tag ..... 16
  - 5.9 Delete Tag ..... 16
  - 5.10 GetTags ..... 17

## 1 INTRODUCTION

---

This document aims at describing how the data from the iCenter OS platform may be accessed by third party systems.

The iCenter OS platform essentially stores data regarding electrical energy. However and depending on the host system where iCenter OS will be deployed it may also store data and provide the same services for other types of utilities such as water and gas.

iCenter OS contains a module that provide access to the intended information, the module was implemented using WCF technology. Accessing this module is similar to accessing a Web Service.

The iCenter OS service revolves around the following concepts:

- Local - represents a physical location that is being monitored. In reality this entity represents each client's house.
- Unit - corresponds to the physical measuring equipment (iMeterBox). Each location has a unit associated.
- Tag - Represents a measure point. A tag may measure electrical consumption, water consumption, gas consumption, temperature, etc.

In this service a unit is only used for registering the hardware and for validation, afterwards the only entities used are Local and Tag that correspond to the client's home and the configured measure points.

## 2 AUTHENTICATION AND COMMON STRUCTURES

This chapter specifies how the client authenticates himself and describes the data structures that are used by and in this service.

### 2.1 AUTHENTICATION

All invocations to this service require the user to provide an access token. The token depends on the user, the user password and a shared key.

The shared key for this service is “ISAiEnergy2009”.

In order to obtain the token the SHA512 (Secure Hash Algorithm) and HMAC (keyed-Hash Message Authentication Code) algorithms are needed.

In C# the following method may be used:

```
string getToken(string user, string passwd){
    HMAC hash2 = new HMACSHA512();
    hash2.Key = Encoding.ASCII.GetBytes("ISAiEnergy2009");
    byte[] hashed =
    hash2.ComputeHash(Encoding.ASCII.GetBytes(user+""+passwd));
    return Convert.ToBase64String(hashed);
}
```

In PHP the following function may be used:

```
function getToken($user,$pass){
    mb_internal_encoding("ASCII");
    $key = "ISAiEnergy2009";
    $hmac = hash_hmac('sha512',$user.$pass,$key,true);
    return base64_encode($hmac);
}
```

### 2.2 UPDATING A PASSWORD

The method for updating a user password is:

```
ResultCode UpdateUserPassword(string token, string oldPassword,
                              string newPassword);
```

Argument	Meaning
token	The user's access token
oldPassword	The user's current password
newPassword	The user's new password
<i>Return value</i>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
InvalidParameter	oldPassword is null or empty newPassword is old or empty

	oldPassword equals newPassword newPassWord equals previous password
<b>InvalidUser</b>	Wrong token unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database
<b>Unauthorized</b>	User can not access unit

## 2.3 DATA STRUCTURES

In this section all the common data structures and enumerations that are used in this service are specified. Data structures specific to a method or set of methods are specified along with those methods.

### 2.3.1 OUTCOME OF AN OPERATION

All method invocations return a code that indicates the success or failure of its execution. The possible values are encoded in the following enumeration:

```
enum ResultCode : int
{
    Success = 1,
    UnexpectedError = 2,
    InvalidUser = 3,
    InvalidParameter = 4,
    NoData = 5,
    Unauthorized = 6
}
```

The first three values are common to all operations of the service, the remaining are specific to each method and will be detailed in each case.

The meanings of the values are:

Value	Meaning
<b>Success</b>	Operation was successful
<b>UnexpectedError</b>	Error during execution
<b>InvalidUser</b>	Invalid user
<b>InvalidParameter</b>	Invalid parameter
<b>NoData</b>	Missing data for the required operation
<b>Unauthorized</b>	The user does not have the required permissions to access some resource

### 2.3.2 CONSUMPTION

The structure Consumption is used to store statistics about energy consumptions.

```
Struct Consumption
{
    decimal avgConsumption; // consumption average (default measurement unit of tag)
    decimal avgCost; // consumption average in currency (default currency of supplier)
    decimal? evolution; // comparison with the last day/month/year
```

```
// if evolution value is null, there are not previous data or the actual
value is 0
string consumptionUnit; // consumption unit
string type; // measurement type (Energy, Gas, Water)
int tagId; // tag id
decimal? lastConsumptionCurrency; // consumption in currency of last
month/year. This depends on the request type.
}
```

### 2.3.3 CONVERSION

The structure is used to return information about conversion factors between two measurement units.

```
Struct Conversion
{
    Data source; // base measurement unit
    Data dest; // converted measurement unit
    Decimal factor; // conversion factor
}
```

### 2.3.4 DATA

The structure Data is used for transferring data that can be represented as a pair of an integer and a string. The structure has the following definition:

```
struct Data
{
    int id; // pair id
    string Name; // pair description or name
}
```

### 2.3.5 DATAITEM

The structure DataItem is used for transferring scalar values associated with a date.

```
struct DataItem
{
    int id; // id of record
    decimal value; // scalar value of record
    DateTime date; // date value of record
}
```

### 2.3.6 DATAITEMAGG

The structure DataItemAgg is used to aggregate DataItems.

```
struct DataItemAgg
{
    int id; // id related to the group
    DataItem[] values; // group of scalar values
}
```

### 2.3.7 DEVICE

The structure Device is used to represent a device in the system.

```
struct Device
{
    int id; // device id
    string name; // device name
    int typeId; // device type id
    int unitId; // unit id of the device
    int interval; // interval of acquisition of the tags of the device
    int remoteId; // device id in the remote system
}
```

### 2.3.8 TAG

The structure Tag is used to represent a tag in the system.

```
struct Tag
{
    int id; // tag id
    string name; // tag name
    int type; // tag type
    int deviceId; // device id of the tag
    int remoteId; // tag id in the remote system
}
```

The options for TagType are available in the method GetTagTypes.

### 2.3.9 UNIT

The structure Unit is used to represent a unit in the system.

```
struct Unit
{
    int id; // unit id
    string name; // unit name
    int interval; // unit interval of communication
    int remoteId; // unit id in the remote system
    string serial; // unit serial id
}
```

## 2.4 OBTAINING ARRAYS OF STRUCTURES

Several methods of this service return arrays of structures. In some situations these arrays can be very large. Since the communication channel imposes limits on the maximum size of each message sent between client and server, there may be cases where one may not be able to obtain all data with just one invocation.

As an example consider a request for data in a given time interval. If the amount of data to be returned is large, the iCenter OS service may not immediately provide all the data. One way of overcoming this limitation is by breaking the original interval into smaller non overlapping consecutive intervals that the communication channel can handle.

### 3 GLOBAL METHODS

#### 3.1 DEFAULT VALUES

There are some fixed tables in the system that client applications need to know in order to invoke some methods.

The following methods provide a way of retrieving data from auxiliary tables:

- GetMeasurements - Returns a list of measures that are available.
- GetMeasurementUnits - Returns a list of the units supported by the system.

**Note:** auxiliary tables are fixed in regard to the pair identifier-name.

##### 3.1.1 AVAILABLE MEASURES

The following method obtains a list of available measures:

```
ResultCode GetMeasurements(string token, out Data [] measurements);
```

Argument	Meaning
Token	The user's access token
measurements	Array containing the measures
<i>Return value</i>	Outcome of the operation

The array obtained contains all the measures defined in the system.

The values that may be returned by the method are:

ResultCode	Meaning
InvalidUser	Wrong token Unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database
NoData	No measurements found

##### 3.1.2 AVAILABLE MEASUREMENT UNITS

The following method obtains list of measurement units that are available:

```
ResultCode GetMeasurementUnits (string token,out Data[] measurementUnits);
```

Argument	Meaning
token	The user's access token

<b>measurementUnits</b>	Array containing the available measurement units
<b>Return value</b>	Outcome of the operation

The obtained array of measurement units contains all the measurement units defined in the system.

The values that may be returned by the method are:

<b>ResultCode</b>	<b>Meaning</b>
<b>InvalidUser</b>	Wrong token Unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database
<b>NoData</b>	No measurement units found

## 4 DATA

There are several methods available and that can be used to obtain data. The following methods allow the user to:

- Obtain consumptions

For manual management of data some methods also exist:

- ReportConsumption
- BulkInsertData
- DeleteData

**Note:** Each data consumption record has the date relative to the beginning of the interval that the record represents (i.e. if we have two records, one with date 01-01-09 00:00 and another with 01.01.09 00:15, the first record represents the consumption between 00:00 and 00:15 and the second record the consumption between 00:15 and the time of the next record).

### 4.1 OBTAIN CONSUMPTIONS

The following method is used to obtain a list of consumptions:

```

resultCode GetConsumptions(string token, int[] tagId, DateTime beginDate,
    DateTime endDate, int aggregationType,
    int measurementUnitId, out List<DataItemAgg>
    consumption);
    
```

Argument	Meaning
<b>token</b>	The user's access token
<b>tagId</b>	Ids of the tags for which consumptions values are obtained
<b>beginDate</b>	Start date of the interval
<b>endDate</b>	End date of the interval
<b>aggregationType</b>	Id of the type of intended aggregation (see note 1 below)
<b>measurementUnitId</b>	Id of the intended measurement unit
<b>consumption</b>	List of requested consumptions
<b>Return value</b>	Outcome of operation

The array of consumptions obtained contains all the consumptions in a given interval.

**Note 1:** The aggregation type may take values from following table:

Id	Description
0	Instantaneous Values
1	Hourly aggregation
2	Daily aggregation
3	Monthly Aggregation

**Note 2:** When aggregated data is requested, data is truncated by hour/day/month and treated that way (i.e. if daily aggregation is requested and the date provided was 15-01-09, the service will return the consumptions of each day of January 2009).

The values that may be returned by the method are:

ResultCode	Meaning
InvalidParameter	tagId is null tagId.length <= 0 beginDate is null endDate is null endDate < beginDate aggregationType < 0 measurementUnitId <= 0
InvalidUser	Wrong token Unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database
NoData	No consumptions found
Unauthorized	Unauthorized access to tag

## 4.2 BULK INSERT OF DATA

In some environments the data is not communicated directly by the hardware to the system and a middleware is used. In these situations the web service has this method to insert data:

```
ResultCode BulkInsertData(string token, DataItem[] data);
```

Argument	Meaning
token	The user's access token
data	The data to insert in the database
<i>Return value</i>	Outcome of the operation

The DataItem structure is used with the following instructions:

- Id: the tag id
- Date: the date when the reading was made
- Value: the value read

The values that may be returned by the method are:

Argument	Meaning
InvalidParameter	data is null data.length = 0
InvalidUser	Wrong token Unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database

## 4.3 REQUEST A SEGMENTED INSERTION

In some situations the amount of data that is to be sent over the network violates the maximum allowable by the underlying protocols. In such cases it is up to the client to break the data into transmissible segments that can be sent without violating protocols. It is also usually in the interest of the client to treat the multiple requests in the context of a transaction, guaranteeing that all the segments of data are inserted or none are. The

following three methods allow the clients to use and control transactions using the web service.

In the given scenario the first step the client has to tell the web service that he intends to send several arrays of data over the network and that following requests to should be considered part of a single transaction. The following method is used to inform the server of the intent of the client:

```
ResultCode RequestSegmentedBulkInsertData(string token);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>Return value</b>	Outcome of the operation

The values that may be returned by the method are:

Argument	Meaning
<b>InvalidUser</b>	Wrong token Unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database

Note: If a transaction is requested while another transaction is still open the data regarding the first is lost. Each client may have only one transaction active at each moment.

#### 4.4 SEGMENTED BULK INSERT OF DATA

The insertion of segment of data is achieved using the following method:

```
ResultCode SegmentedBulkInsertData(string token, DataItem[] data);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>data</b>	The data to insert in the database
<b>Return value</b>	Outcome of the operation

The DataItem structure is used with the following instructions:

- Id: the tag id
- Date: the date when the reading was made
- Value: the value read

The values that may be returned by the method are:

Argument	Meaning
<b>InvalidParameter</b>	data is null data.length = 0
<b>InvalidUser</b>	Wrong token Unauthorized access to ws method

<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database

#### 4.5 COMMIT A SEGMENTED INSERTION

After the client has sent all the data over the network using as many requests as needed the server should be informed in order to commit the data and make it available to the rest of the platform. The following method commits the data:

```
ResultCode CommitSegmentedBulkInsert(string token);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>Return value</b>	Outcome of the operation

The values that may be returned by the method are:

Argument	Meaning
<b>InvalidUser</b>	Wrong token Unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database

#### 4.6 ROLLBACK A SEGMENTED INSERTION

If a client encounters an error during a transaction and wishes that the data sent be ignored by the server the following method should be invoked:

```
ResultCode RollbackSegmentedBulkInsert(string token);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>Return value</b>	Outcome of the operation

The values that may be returned by the method are:

Argument	Meaning
<b>InvalidUser</b>	Wrong token Unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database

#### 4.7 REMOVAL OF DATA

The user can delete data from iCenter OS database using the following method:

```
ResultCode DeleteData(string token, int tagId, DateTime beginDate,
    DateTime endDate);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>tagId</b>	Id of the tag

<b>beginDate</b>	Start date
<b>endDate</b>	End date
<b>Return value</b>	Outcome of operation

The values that may be returned by the method are:

Argument	Meaning
<b>InvalidParameter</b>	tagid < 1 beginDate is null endDate is null endDate < beginDate
<b>InvalidUser</b>	Wrong token Unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database

## 5 HARDWARE MANAGEMENT

Some users of the iCenter OS platform may need to manage directly the hardware in the database. Methods to insert, edit and delete units, devices and tags are available.

### 5.1 INSERT UNIT

The method to insert a unit record in the system is:

```
ResultCode InsertUnit(string token, Unit u, out int id);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>u</b>	The unit information
<b>id</b>	The id generated
<b>Return value</b>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
<b>InvalidParameter</b>	u is null u.interval < 0 u.name is null or empty u.name length > 25 u.remoteld < 0 u.serial is null or empty
<b>InvalidUser</b>	Wrong token unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database
<b>Unauthorized</b>	User can not access unit

### 5.2 EDIT UNIT

The method to edit a unit record in the system is:

```
ResultCode EditUnit(string token, Unit u);
```

Argument	Meaning
<b>token</b>	The user's access token
<b>u</b>	The unit information
<b>Return value</b>	Outcome of the operation
<b>Unauthorized</b>	User can not access to unit

### 5.3 DELETE UNIT

The method to edit a unit record in the system is:

```
ResultCode DeleteUnit(string token, int unt_id);
```

Argument	Meaning
token	The user's access token
unt_id	The unit id to delete
<i>Return value</i>	Outcome of the operation

This method deletes the unit and all its devices and tags.

The values that can be returned by the method are:

ResultCode	Meaning
InvalidParameter	unt_id < 0
InvalidUser	Wrong token unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database
Unauthorized	User can not access unit

## 5.4 INSERT DEVICE

The method to insert a device record in the system is:

```
ResultCode InsertDevice(string token, Device d, out int id);
```

Argument	Meaning
token	The user's access token
d	The device information
id	The id generated
<i>Return value</i>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
InvalidParameter	d is null d.interval < 0 d.name is null or empty d.name.length > 25 d.remoteld < 0 d.unitid < 0 d.unitid is invalid
InvalidUser	Wrong token unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database

## 5.5 EDIT DEVICE

The method to edit a device record in the system is:

```
ResultCode EditDevice(string token, Device d);
```

Argument	Meaning
token	The user's access token
d	The device information
<i>Return value</i>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
InvalidParameter	d is null d.id < 0 d.interval < 0 d.name is null or empty d.name.length < 25 d.remoteld < 0 d.unitld < 0
InvalidUser	Wrong token Unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database
Unauthorized	User can not access device

## 5.6 DELETE DEVICE

The method to delete a device record in the system is:

```
ResultCode DeleteDevice(string token, int d_id);
```

Argument	Meaning
token	The user's access token
d_id	The device id to delete
<i>Return value</i>	Outcome of the operation

This method deletes the device and all its tags.

The values that can be returned by the method are:

ResultCode	Meaning
InvalidParameter	d_id < 0
InvalidUser	Wrong token Unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database
Unauthorized	User can not access device

## 5.7 INSERT TAG

The method to insert a tag record in the system is:

```
ResultCode InsertTag(string token, Device t, out int id);
```

Argument	Meaning
token	The user's access token
t	The tag information
id	The id generated
<i>Return value</i>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
InvalidUser	Wrong token unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database

## 5.8 EDIT TAG

The method to edit a tag record in the system is:

```
ResultCode EditTag(string token, Tag t);
```

Argument	Meaning
token	The user's access token
t	The tag information
<i>Return value</i>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
InvalidUser	Wrong token unauthorized access to ws method
Success	Operation occurred with success
UnexpectedError	Unexpected error with the database
Unauthorized	User can not access tag

## 5.9 DELETE TAG

The method to delete a tag record in the system is:

```
ResultCode DeleteTag(string token, int tag_id);
```

Argument	Meaning
token	The user's access token
tag_id	The tag id to delete
<i>Return value</i>	Outcome of the operation

This method deletes the tag.

The values that can be returned by the method are:

ResultCode	Meaning
InvalidParameter	tag_id < 0
InvalidUser	Wrong token unauthorized access to ws method

<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database
<b>Unauthorized</b>	User can not access tag

## 5.10 GETTAGS

The method to get a list of tags that the user has permissions to access:

```
ResultCode GetTags(string token, out Tag[] tags);
```

Argument	Meaning
<b>Token</b>	The user's access token
<b>tags</b>	The tags list
<b>Return value</b>	Outcome of the operation

The values that can be returned by the method are:

ResultCode	Meaning
<b>InvalidUser</b>	Wrong token unauthorized access to ws method
<b>Success</b>	Operation occurred with success
<b>UnexpectedError</b>	Unexpected error with the database
<b>NoData</b>	There are no tags that the user has access