# Data Service API

Technical Documentation

## General info

*Table 1 - General document info*

| | |
|---|---|
| **Project:** | **AE Site*Link*** |
| **Project version** | 5.1 |
| **Created by:** | M. Kress |
| **Date:** | 2013-07-20 |
| **Document version:** | 6 |
| **Date of change:** | 2014-12-13 |

## Contents

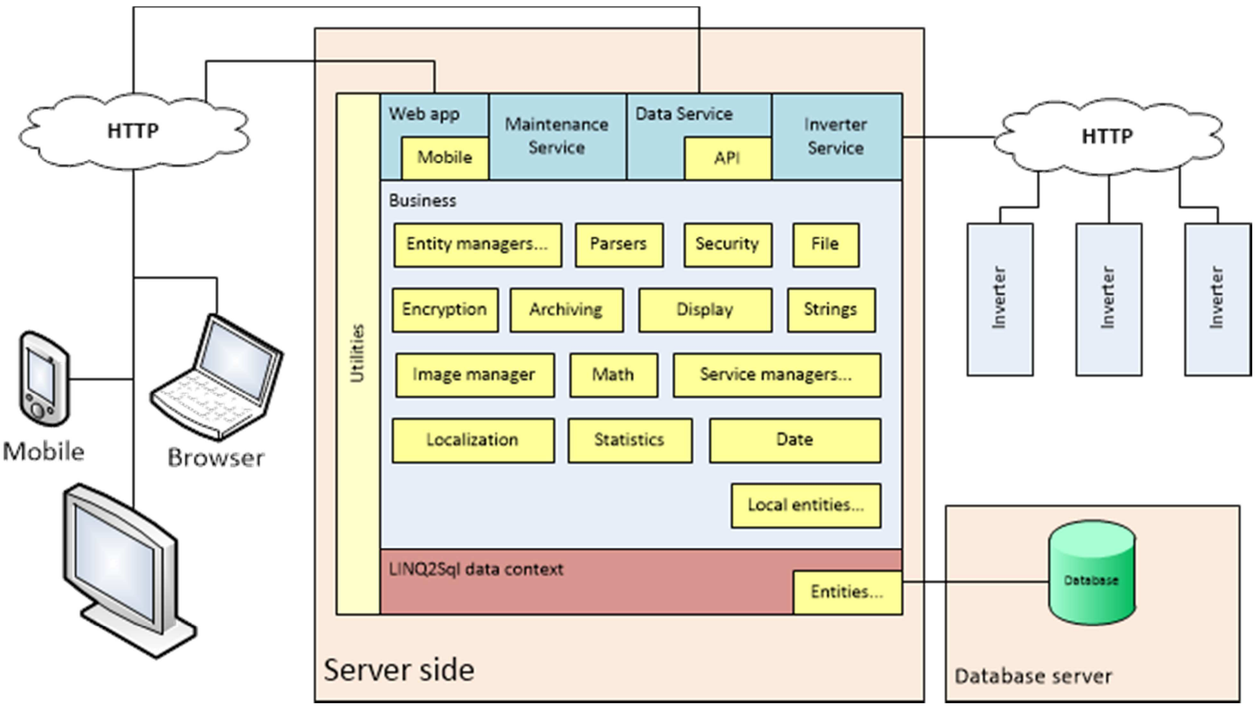
## Overview

AE Site*Link* provides an API through the Data Service project, which can be accessed via HTTP to retrieve solar object data. The API is a simple ASP.Net web service. The API publishes multiple methods that can be parameterized with a single XML parameter. The returned result is also a single XML.

## System settings

Data Service API can be accessed here: http://ds.aesitelink.de/DataServiceApi.asmx

## Architecture



*Architecture of the AE Site*Link *environment focusing on parts that are important from the point of view of the Data Service*

The *Data Service* is a big part of the AE Site*Link* infrastructure. The Data Service web application hosts three web services: *API module, Client dedicated module, REST service*.
The API module is the part of the Data Service which handles the API access of the various 3$^{rd}$ party clients.

## Recommended usage

Inverters send data mostly in 10 minute intervals so the frequency of the queries should be a maximum of 1/10 minutes for real-time data. There is no need to summarize daily data, because the daily, monthly and yearly summaries can be requested using the *GetObjectDataForInterval* method.

## API features

The current version the service contains the following entry points:

*Table 2 - API features, entry points*

| Entry point | Description |
|---|---|
| **GetInverterList** | Get the list of inverters of a plant. |
| **GetInverterData** | Get the historical data of an inverter for a given period. |
| **GetObjectDataForInterval** | Get the timestamp-value pairs of daily-, monthly-, yearly summaries of any solar object, any data type and any period. |
| **GetObjectErrors** | Get number of error messages for a certain solar object. |
| **GetObjectErrorsCount** | Get paged error messages for certain solar object based on the page size and the page number of the request. |

**Implementation note:**

The signatures of the entry points in the code look like this:

```
1. string GetInverterData(string xmlData)
2. string GetInverterList(string xmlData)
3. string GetObjectDataForInterval(string xmlData)
4. string GetObjectErrors(string xmlData)
5. string GetObjectErrorsCount(string xmlData)
```

Each of the methods accepts a string type parameter which is the XML data sent by the client. The following steps are taken when the methods are invoked:

- Load the *xmlData* parameter into an `XmlDocument` variable (Generate parse error if input is not correct)
- Position to the root of the document (*rws_request*)
- Validate the root of the document
- Extract the authentication part
- Authenticate user; if successful return the user from the database for later use
- Verify whether the user accessed the service more than it was allowed; if yes return error
- Generate the response data
- Return the results

## Correct input

The input provided by the client applications has to be correct:

- Has to be a correct XML file
- All the nodes have to be Pascal cased (Auth, Username, GetInverterList)
- All the specified fields have to be set

If the input is correct, the API will return the requested information.

```
<rws_response>
    [Contents]
</rws_response>
```

*Code 1 - Correct response packet structure*

### Case sensitivity

The XML has to be case sensitive, but the parameters don't have to be (except the password). The following package is completely acceptable:

```
<?xml version="1.0" encoding="utf-8"?>
<rws_request>
    <Auth>
        <Username>
            MyUserNAME
        </Username>
        <Password>
            [password]
        </Password>
    </Auth>
    <GetObjectDataForInterval>
        <ObjectId>11</ObjectId>
        <ObjectType>PLANT</ObjectType>
        <DataType>ACpower</DataType>
        <IntervalType>MontH</IntervalType>
        <FromDate>1309478400</FromDate>
        <ToDate>1341100800</ToDate>
    </GetObjectDataForInterval>
</rws_request>
```

### Decimal format

The values in the response packages are in decimal format, using English formatting rules (en-US culture): 1,234.56.

Exception from this rule is the GetInverterData entry point which returns decimals as integers and provides the decimal factor, indicating how many of the integer's digits are decimals.

### Date format

To avoid date formatting issues, all the dates in the application are encoded as UNIX timestamps.

**Note:**

You can use the following link to convert to- and from UNIX timestamp: http://www.epochconverter.com/

## Error messages

The text of the request has to be a valid XML. If the package is not valid the following non-localized package will be sent back:

```
<rws_response>
    <error>
        [Error message]
    </error>
</rws_response>
```

*Code 2 - Sample error response*

If the authentication fails, an error response is sent:

```
<rws_response>
    <error>Authentication failed</error>
</rws_response>
```

*Code 3 - Authentication failed error*

If a request arrives containing user data that is not associated with the plant in the request the following non-localized error message is sent back:

```
<rws_response>
    <error>Access denied. The requested plant's data is not accessible for the
specified user</error>
</rws_response>
```

*Code 4 - Error message in case the user has insufficient rights to access the data*

If the user requests error messages specifying a page size larger than 100 the following error is returned:

```
<rws_response>
    <error>Maximum size of a page is 100.</error>
</rws_response>
```

*Code 5 - Error message when the requested page size is >100 in the case of error messages*

**Note:**

Errors do not have code associated with them yet. They will be added in future releases.

## User authentication

Every requested towards the API has to be authenticated with a username and password. The username and password have to be the same as the registered user’s in AE Site*Link*.
This authentication part has to be the first entry in the request package:

```
<Auth>
    <Username>
        [Username]
    </Username>
    <Password>
        [Password]
    </Password>
</Auth>
```

*Code 6 - Authentication block*

**Note:**

The password has to be provided in non-encrypted form. This approach has to be further discussed with the consumers of the API's features.

For answers on incorrect authentication see Error messages.

## Entry points

### Get Inverter List

Entry point

Invoke URL: http://ds.aesitelink.de/DataServiceapi.asmx?op=GetInverterList

#### Request

The request has to contain the corresponding plant id and has to look like this:

```
<rws_request>
    <Auth>
        <Username>
            [Username]
        </Username>
        <Password>
            [Password]
        </Password>
    </Auth>
    <GetInverterList>
        <Plant>3</Plant>
    </GetInverterList>
</rws_request>
```

*Code 7 - Sample inverter list request packet*

#### Response

The response contains the whole structure of the plant with all sub-plants and inverters in it:

```
<rws_response>
    <Plant id='3' name='REFU Metzingen / Neuhausen'>
        <Subplant id='3' name='Dünnschicht'>
            <Inverter serial='A0264' name='211' id='1' />
            <Inverter serial='NB212' name='212' id='2' />
            <Inverter serial='NB213' name='213' id='3' />
            <Inverter serial='A0268' name='214' id='4' />
        </Subplant>
        <Subplant id='46' name='Monokristalin'></Subplant>
        <Subplant id='39' name='Polykristalin'></Subplant>
    </Plant>
</rws_response>
```

*Code 8 - Sample response packet containing inverters of a plant*

## Get Inverter Data

Entry point

Invoke URL: http://ds.aesitelink.de/DataServiceapi.asmx?op=GetInverterData

The users are able to request data of plants that are assigned to them.
The request has to contain the inverter id and two timestamps (UNIX time, u32) indicating the range of data the caller wants to have.

The GetInverterData entry point is the first implemented so it differs from the rest of the methods. Decimal values are encoded as integers with decimal factor provided in the *dec* attribute, instead of using English formatting rules.

### Request

```
<rws_request>
    <Auth>
        <Username>
            [Username]
        </Username>
        <Password>
            [Password]
        </Password>
    </Auth>
    <GetInverterData>
        <InverterId>1</InverterId>
        <FromDate>1308642151</FromDate>
        <ToDate>1308642751</ToDate>
    </GetInverterData>
</rws_request>
```

*Code 9 - Sample packet for getting inverter data*

## Response

The response contains the inverter data in that specified range. The response is limited to 100 records (set by the host using the *Settings.MaxAllowedInverterDataRecords* setting). If there are more records in that range the amount of not sent values is also indicated in that response. Always the first *Settings.MaxAllowedInverterDataRecords* records are sent back.

```
<rws_response>
    <InverterData InverterId='2' RecordsLeft='0'>
        <Data Time='1308642600'>
            <p i='PAC' dec='1'>24291</p>
            <p i='IAC' dec='1'>106</p>
            <p i='UAC' dec='1'>2286</p>
            <p i='FAC' dec='0'>50</p>
            <p i='PDC' dec='1'>24534</p>
            <p i='IDC' dec='1'>45</p>
            <p i='UDC' dec='1'>5262</p>
            <p i='Temp1' dec='1'>332</p>
            <p i='Temp2' dec='1'>404</p>
            <p i='Rad' dec='1'>1637</p>
            <p i='PanelTemp' dec='1'>226</p>
            <p i='Energy' dec='0'>2</p>
            <p i='TotalEnergy' dec='0'>22384</p>
            <p i='DaySunEnergy' dec='4'>1532</p>
            <p i='State' dec='0'>4</p>
        </Data>
    </InverterData>
</rws_response>
```

*Code 10 - Sample response packet with inverter data in it*

### *Parameter data*

The type of returned parameter is contained in the *i* attribute of each *p* tag. The dec attribute indicates how many of the digits inside the tag are of the decimal part.

**Example:**

The value 1,241,020.25 for total energy is encoded as

```
<p i='TotalEnergy' dec='2'>124102025</p>
```

The following table displays the parameters with their IDs along with information about the units, data types and decimal factors for the corresponding parameters:

*Table 3 - Parameter information*

| Name | Description | Unit | Data Type |
|---|---|---|---|
| **PAC** | AC Power | W | Float32 |
| **IAC** | AC Current | A | Float32 |
| **UAC** | AC Voltage | V | Float32 |
| **FAC** | Frequency | Hz | Float32 |
| **PDC** | DC Power | W | Float32 |
| **IDC** | DC Current | A | Float32 |
| **UDC** | DC Voltage | V | Float32 |
| **Temp1** | Internal temperature 1 | °C | Signed16 |
| **Temp2** | Internal temperature 2 | °C | Signed16 |
| **Rad** | Radiation | W/m² | Signed16 |
| **PanelTemp** | Panel temperature | °C | Signed16 |
| **Energy** | Daily energy | kWh | Unsigned32 |
| **TotalEnergy** | Total energy | kWh | Unsigned32 |
| **DaySunEnergy** | Daily sun energy | kWh/m² | Unsigned32 |
| **State** | State of the inverter | - | Signed16 |

The following table shows all possible inverter states:

*Table 4 - Possible inverter states*

| State ID | Description |
|---|---|
| **0** | Initialization |
| **1** | Switched off |
| **2** | Activation |
| **3** | Ready |
| **4** | Run |
| **5** | Ramp back to zero |
| **6** | Break |
| **7** | Fault |
| **8** | Testing |
| **9** | NoCountrySet |
| **10** | Update |
| **16** | ShutDown |
| **20** | Generated |
| **21** | NoData |

### *Number of returned records*

Inverters are sending 76 data packets a day in average. Rounding up this number gives us 80; further rounding up to 100 could be done to ensure that the whole day can be returned in one message. This value is set as initial value for the DataService project setting *MaxAllowedInverterDataRecords*.

### *Number of requests*

The number of requests a user can send can be limited by the host to a defined number for a certain period. Each call increments the counter by one; the content of the packet does not matter.

## Get Object Data for Interval (NEW)

Entry point

Invoke URL: http://ds.aesitelink.de/DataServiceapi.asmx?op=GetObjectDataForInterval

### Request

```
<?xml version="1.0" encoding="utf-8"?>
<rws_request>
    <Auth>
        <Username>
            [username]
        </Username>
        <Password>
            [password]
        </Password>
    </Auth>
    <GetObjectDataForInterval>
        <ObjectId>11</ObjectId>
        <ObjectType>Plant</ObjectType>
        <DataType>Energy</DataType>
        <IntervalType>Month</IntervalType>
        <FromDate>1309478400</FromDate>
        <ToDate>1341100800</ToDate>
    </GetObjectDataForInterval>
</rws_request>
```

*Code 11 - Sample request packet*

#### *ObjectType*

The object type tag specified the type of the solar object we need data for.

Possible values are:

- *Plant*
- *Subplant*
- *Inverter*

The values are not case sensitive.

#### *DataType*

The DataType node is a string based value. The available options are detailed below.

#### *Available monthly data types*

Monthly data is the 1 day resolution archived data of the inverters. Monthly data is the basis for the monthly charts and basically means production of inverters for each day.

The following DataType values can be set for the 'Month' IntervalType:

- *Energy* (synonym - *DailyYield)*
- *DailyEnergyNormalized*
- *PerformanceRatio*
- *Income*
- *ReducedCO2*
- *NormalizedEnergyDeviation* (only for inverters) (synonym - *NormalizedYieldDeviation)*

- *NormalizedEnergyDeviationPercent* (only for inverters) (synonym - *NormalizedYieldDeviationPercent)*
- *DailySunEnergy*

### *Available yearly data types*

Yearly data is the 1 month resolution archived data of the inverters. Yearly data is the basis for the yearly charts and basically means production of inverters for each month.

The following DataType values can be set for the 'Year' IntervalType:

- *Energy* (synonym - *DailyYield)*
- *DailyEnergyNormalized*
- *PerformanceRatio*
- *Income*
- *ReducedCO2*
- *NormalizedEnergyDeviation* (synonym - *NormalizedYieldDeviation)*
- *NormalizedEnergyDeviationPercent* (synonym - *NormalizedYieldDeviationPercent)*
- *EnergyForecast* (only for inverters) (synonym – *IdealEnergy)*
- *IncomeForecast* (only for inverters) (synonym – *IdealIncome)*
- *DailySunEnergy*

### *Available overall data types*

Overall data is the 1 year resolution archived data of the inverters. Overall/all year's data is the basis for the overall charts and basically means production of inverters for each year.

The following DataType values can be set for the 'Overall' IntervalType:

- *Energy (*synonym - *DailyYield)*
- *DailyEnergyNormalized*
- *PerformanceRatio*
- *Income*
- *ReducedCO2*
- *NormalizedEnergyDeviation* (synonym - *NormalizedYieldDeviation)*
- *NormalizedEnergyDeviationPercent* (synonym - *NormalizedYieldDeviationPercent)*
- *EnergyForecast* (only for inverters) (synonym – *IdealEnergy)*
- *IncomeForecast* (only for inverters) (synonym – *IdealIncome)*
- *DailySunEnergy*

### *IntervalType*

IntervalType possible values are:

- *Month* (days of the month)
- *Year* (months of the year)
- *Overall* (years)

## Response

The method returns the data in timestamp-value pairs. The values are in decimal format, using English formatting rules.

```
<rws_response>
    <d t='1309478400'>594.700004577637</d>
    <d t='1309564800'>894.700012207031</d>
    <d t='1309651200'>984.200004577637</d>
    <d t='1309737600'>1073.70000457764</d>
</rws_response>
```

*Code 12 - Sample response packet containing inverter data*

## Get Object Number of Error Messages (NEW)

Entry point

Invoke URL: http://ds.aesitelink.de/DataServiceapi.asmx?op=GetObjectErrorsCount

### Request

```
<?xml version="1.0" encoding="utf-8"?>
<rws_request>
    <Auth>
        <Username>
            [username]
        </Username>
        <Password>
            [password]
        </Password>
    </Auth>
    <GetObjectErrorsCount>
        <ObjectId>11309</ObjectId>
        <ObjectType>Inverter</ObjectType>
    </GetObjectErrorsCount>
</rws_request>
```

*Code 13 - Sample request packet*

#### *ObjectType*

The object type tag specified the type of the solar object we need data for; just like in *get object data for interval* request.

### Response

The method returns the error messages count for the specified object type and id.

```
<rws_response>
      <Errors count='279'>
      </Errors>
</rws_response>
```

*Code 14 - Sample response packet*

## Get Object Error Messages (NEW)

Entry point

Invoke URL: http://ds.aesitelink.de/DataServiceapi.asmx?op=GetObjectErrors

### Request

```
<?xml version="1.0" encoding="utf-8"?>
<rws_request>
    <Auth>
        <Username>
            [username]
        </Username>
        <Password>
            [password]
        </Password>
    </Auth>
    <GetObjectErrors>
        <ObjectId>11309</ObjectId>
        <ObjectType>Inverter</ObjectType>
             <PageSize>10</PageSize>
             <PageNumber>1</PageNumber>
    </GetObjectErrors>
</rws_request>
```

*Code 15 - Sample request packet*

#### *ObjectType*

The object type tag specifies the type of the solar object we need data for, just like in get object data for interval request.

#### *Page size*

The page size tag specifies how many error messages are returned. This value is limited to 100.

#### *Page number*

The page number tag indicates the number of page that is returned by the web service.

## Response

The method returns the paged error messages for the specified object type and id. According to the request the first page containing 10 items will be returned.

```
<rws_response>
       <Error code='0A0004' date='1329322671' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
       <Error code='100002' date='1329321817' stateId='4' state='Run'>
              <Text>Ethernet connectn. 2</Text>
       </Error >
       <Error code='0A0004' date='1329320904' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
       <Error code='0A0004' date='1329316194' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
       <Error code='0A0004' date='1329315749' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
     <Error code='0A0004' date='1329315324' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
       <Error code='0A0004' date='1329314930' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error >
       <Error code='0A0004' date='1329311845' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
       <Error code='0A0004' date='1329309594' stateId='4' state='Run'>
              <Text>Regulator voltage 4</Text>
       </Error>
       <Error code='100002' date='1329299661' stateId='4' state='Run'>
              <Text>Ethernet connectn. 2</Text>
       </Error>
</rws_response>
```

*Code 16 - Sample response packet*

## Getting started with programming

### HTTP GET

The following is a sample HTTP GET request and response. The placeholder *[XML request param]* shown need to be replaced with actual values.

```
GET /DataServiceapi.asmx/GetInverterData?xmlData=[XML request param] HTTP/1.1
Host: ds.aesitelink.de
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="DataService.API">[XML response param]</string>
```

### HTTP POST

The following is a sample HTTP POST request and response. The placeholder *[XML request param]* shown need to be replaced with actual values.

```
POST /DataServiceapi.asmx/GetInverterData HTTP/1.1
Host: ds.aesitelink.de
Content-Type: application/x-www-form-urlencoded
Content-Length: length

xmlData=[XML request param]
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="DataService.API">[XML response param]</string>
```