Documentation Acubiz EMS SOAP API

Version 1 / 23-12-2019

Contents

Acubiz A/S

Bregnerødvej 133C

DK-3460 Birkerød

Tel: +45 70 214

ACCESS TO ACUBIZ SOAP API

Different languages and frameworks have their own functionality for using a SOAP-based web service. Most of these involve automatically generating client code based on WSDL, but you can also build XML envelopes entirely manually.

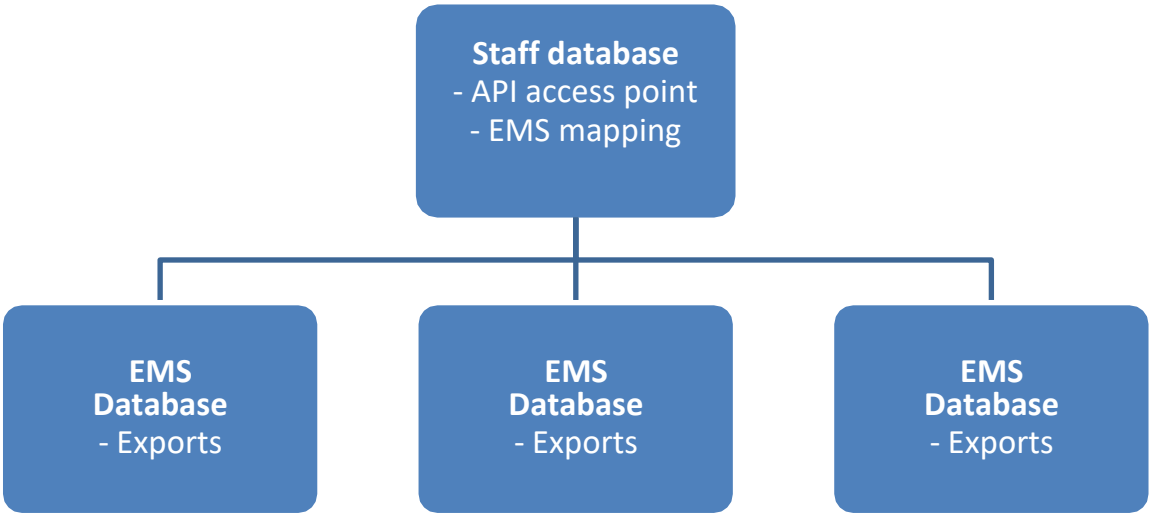
You need to make sure that the session cookie is sent with all the transactions. Some frame sets will do this automatically, but not all.

ARCHITECTURE

Acubiz EMS is built using a Staff and configuration database, which is the core of Acubiz EMS. There are also one or more EMS databases, depending on how many segments you want to have.

The contact point for Acubiz SOAP API is the Staff database, which contains users and mapping of the associated EMS databases. Through the API, you can also access the dimensions and exports of each EMS database.

This architecture means that you need to have an action for the EMS database that you want to access, in order to manage the dimensions and exports.



ACCESS VIA .NET

You can add a service reference to Acubiz SOAP API via the Service References menu in Visual Studio.

The URL for your SOAP API is displayed in the API module's settings.

In order to read the WSDL description, you must authenticate with a user on your system. It is recommended that you use a user who is specifically set up for API or an admin user (see Authentication).

Once you have done this, you need to modify the service reference to use the basic authentication and to send and receive larger packages. You do this by changing the service reference binding to contain a clientCredentialType called ”Basic”: ’

In short:

```
<security mode="Transport">
        <transport clientCredentialType="Basic" />
</security>
```

The total <bindings> will therefore look like this. When you imported the Web Service, it was automatically generated, which is why you can't enter it as it is and why you need to merge in your <security> tag.

```
<bindings>
  <basicHttpBinding>
    <binding name="DominoSoapBinding" closeTimeout="00:10:00"
openTimeout="00:10:00" receiveTimeout="00:10:00" sendTimeout="00:10:00"
maxReceivedMessageSize="1265536">
      <security mode="Transport">
        <transport clientCredentialType="Basic" />
       </security>
    </binding>
   </basicHttpBinding>
</bindings>
```

In order to authenticate and connect to the Web Service, you need to do the following. Here we have called our service reference Acubiz.

```
Acubiz.SessionClient WS = new Acubiz.SessionClient();
WS.ClientCredentials.UserName.UserName = <brugernavn>
WS.ClientCredentials.UserName.Password = <adgangskode> WS.Open();
```

To read which state the service reference is in, you need to use the following call:

```
WS.State.ToString();
```

Det vil give tre svar:

- Created
- Opened
- Closed

Authentication

Any user who is set up in the system can authenticate in the web service in order to retrieve WSDL and to communicate with Acubiz.

However, it is only possible to use the functions and view the data related to the user's access.

This means that if you are not an admin for the system, you can't set up or update users and dimensions, just as it's only the finance users who are able to export.

GENERAL PRINCIPLES IN ACUBIZ SOAP API

The `SessionClient` class contains methods that are used to read, write and update data in Acubiz EMS.

Data is carried in associated classes, such as USER, EMS and DIMENSION. These classes are received with data, and the USER and DIMENSION classes can be returned with updates. When an update is sent, the adjusted object is returned with any error codes.

Method call where an array of data is returned

In a method call, where a class with ARRAYOFEMS are returned, you can state how many you want returned and where in the overall list you want to start.

This option is provided to manage the size of data quantities that are returned. Here you can only state the value 0 in both instances, which returns all elements, or in case of longer lists, a sub-quantity.

ACCESSIBLE FUNCTIONS

The`SessionClient` class contains methods that are used to read, write and update data in Acubiz EMS.

GENERAL METHODS

These methods return data about the user who has been used to log in, which version of the API is being used, and which functions are being used for handling and describing errors.

- `APIVERSION()`
- `GETCURRENTUSER()`
- `GETUSERNAME()`
- `GETEFFECTIVEUSERNAME()`
- `ERRORSTRING()`
- `ISERROR()`

METHODS AND CLASSES FOR HANDLING EMS DATABASES

Unlike users, both dimensions and appendices can lie in several different data silos. These silos are called EMS Databases, here termed an EMS.

These methods and classes are used to find the EMS that is required when working with dimensions or export data. You cannot work with these two types of data without sending an EMS in the call too.

Methods

- `GETDEFAULTEMS()`
- `GETEMS(string)`
- `GETEMSUNITS()`

Classes

- EMS
  - EMSID : String
  - EMSName : String
  - EMSVersion : String
  - IsError : Boolean
  - Errorstring : String

- ARRAYOFEMS
  - Count : Long
  - Position : Long
  - Array() : User
  - IsValid : Boolean

METHODS AND CLASSES FOR HANDLING EXPORT DATA

Method

- `EXPORTCSVDATA(Acubiz.EMS, string, string, bool)`
- `EXPORTXMLDATA(Acubiz.EMS, string, string, bool)`

Class

- EXPORT DATA
  - ID : String
  - Data : Variant
  - Count : Long
  - Format : String
  - ExportProfileName : String
  - ExportDateTime : String
  - IsExported : Boolean
  - IsError : Boolean
  - ErrorInfo : String

Next, the number of returned entries can be read via `Users.COUNT`.

```
if (User.ISERROR)
```

```
      {
                        MessageBox.Show("Error: " + User.ERRORSTRING, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
User.ISADMIN =
                        checkBoxUserAdmin.Checked;
User.ISFINANCE =        checkBoxUserFinance.Checked;
User.ISPREAPPROVER =    checkBoxUserPreApprover.Checked;

User = WS.UPDATEUSER(User);

if (User.ISERROR)
      {
            MessageBox.Show(User.ERRORSTRING, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
```

1.1.1.1 Parameters

- The function doesn't receive any parameters.

1.1.1.2 Return value

- ARRAYOFEMS object

1.1.1.3 Example

In this example, information is retrieved about all the EMS databases, after which the number is written to a label and all the EMS databases are written to a listView.

Here, the actual EMS object is put on the listView's tag parameter. This means that the EMB object is retrieved directly when the listViewItem is selected, without needing to retrieve it a second time.

```
listViewEMS.Items.Clear();


ListViewItem item = null;


Acubiz.ARRAYOFEMS EMSUnits = WS.GETEMSUNITS();
labelEMSFound.Text = "EMS Found: " + EMSUnits.COUNT.ToString();
foreach (Acubiz.EMS EMS in EMSUnits.ARRAY)
{
      item = new ListViewItem();
      item.Text = EMS.EMSNAME;
      item.Tag = EMS;
      listViewEMS.Items.Add(item);
}
```

1.1.1.4 Example

In this example, the EMS action is retrieved from a listview (see example from GETEMSUNITS on page 14) and is used to find the dimension whose ID is written in the textBox textBoxDimSearch.

```
Acubiz.EMS ems = (Acubiz.EMS)listViewEMS.SelectedItems[0].Tag;
Acubiz.DIMENSION dimension = WS.GETDIMENSION (ems, textBoxDimSearch.Text);
```

In this example, the dimension “0010” is retrieved from the EMS database “marketing”.

```
Acubiz.EMS ems = WS.GETEMS(“marketing”);
Acubiz.DIMENSION dimension = WS.GETDIMENSION (ems, “0010”);
```

EXPORTING DATA

Exporting data is done from export profiles that have already been set up in the system.

Just as the exports can be exported as files, you can also export them via the SOAP API. This is done via the function EXPORTCSVDATA and EXPORTXTMLDATA.

EXPORTCSVDATA

EXPORTCSVDATA launches a data export and then delivers a data set of data by return.

1.1.1.5 Parameters

- EMS (EMS): Action for the EMS database you want to set up in
- ExportProfile (String): Export profile name
- ExportType (String): Export type
- IsTestExport (Bool): States whether it is a test export or data that needs to be marked as exported.

1.1.1.6 Response

- EXPORTCSVDATA object

Data is returned in an EXPORTDATA object that contains data and information about any errors, quantities and a unique ID for the export.

- EXPORTCSVDATA
  - ID : String
  - Data : Variant
  - Count : Long
  - Format : String
  - ExportProfileName : String
  - ExportDateTime : String
  - IsExported : Boolean
  - IsError : Boolean
  - ErrorInfo : String

The export is returned with an ID, which is a unique identification for this export.

Data lies in the DATA variant, as a string array of data lines. The data lines in this version of the API are semicolon-separated lines with those fields that are described in the export profile. So you can determine which fields you want returned in the export profile, which is set up in the application.

The error is returned in ErrorInfo, where IsError is set to true if there is an error.

If the export is complete, the IsExported is set to true. If a problem occurs that stops the data set from being exported, the IsExported is set to false.

The format is returned as a CSV in this

call. The count contains the number of

lines in the export.

1.1.1.7 Example

This example retrieves all the export data that has not been exported in the EMS database "marketing" from the export profile "01. Costs - without summary" with the export type "Expense Accounts".

```
Acubiz.EMS ems = WS.GETEMS(“marketing”);

Acubiz.EXPORTDATA exportData = WS.EXPORTCSVDATA(ems,” 01. Omkostninger - uden
summering”, ” Expense Accounts”, true);
```

EXPORTXMLDATA

EXPORTXMLVDATA launches a data export and then delivers a data set of data by return.

1.1.1.8 Parameters

- EMS (EMS): Action for the EMS database you want to set up in
- ExportProfile (String): Export profile name
- ExportType (String): Export type
- IsTestExport (Bool): States whether it is a test export or data that needs to be marked as exported.

1.1.1.9 Response

- EXPORTXMLDATA object

Data is returned in an EXPORTDATA object that contains data and information about any errors, quantities and a unique ID for the export.

- EXPORTXMLDATA
  - ID : String
  - Data : Variant
  - Count : Long
  - Format : String
  - ExportProfileName : String
  - ExportDateTime : String
  - IsExported : Boolean
  - IsError : Boolean
  - ErrorInfo : String

The export is returned with an ID, which is a unique identification for this export.

Data lies in the DATA variant, as a string array of XML data. The data lines in this version of the API are XML fields that are described in the export profile. So you can determine which fields you want returned in the export profile, which is set up in the application.

The error is returned in ErrorInfo, where IsError is set to true if there is an error.

If the export is complete, the IsExported is set to true. If a problem occurs that stops the data set from being exported, the IsExported is set to false.

The format is set to XML for this call.

The count contains the number of lines

in the export.

1.1.1.10 Example

This example retrieves all the export data that has not been exported in the EMS database "marketing" from the export profile "01. Costs - without summary" with the export type "Expense Accounts".

```
Acubiz.EMS ems = WS.GETEMS(“marketing”);

Acubiz.EXPORTDATA exportData = WS.EXPORTXMLDATA(ems,” 01. Omkostninger - uden summer-
ing”, ” Expense Accounts”, true);
```