Documentation Acubiz EMS SOAP API

Version 1.7 / 28-08-2017

Acubiz A/S

Bregnerødvej 133C

DK-3460 Birkerød

Tel: +45 70 214

Contents

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.

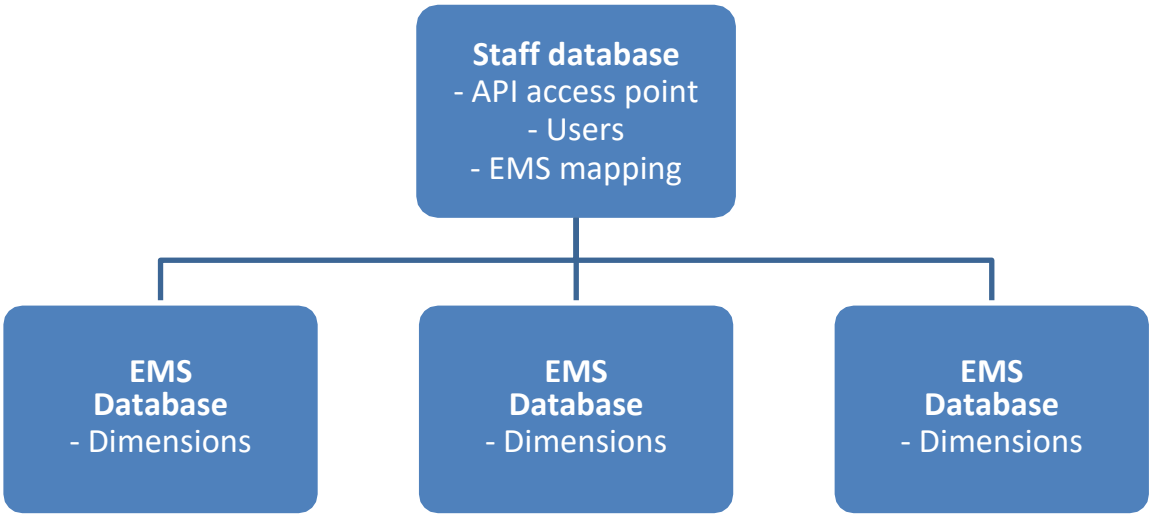
You need to make sure that the session cookie is sent with all the transactions. Some frame sets will do this automatically, but not all.

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" maxBufferSize="999999" maxReceivedMes-
sageSize="999999">
      <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 ARRAYOFUSER, ARRAYOFEMS or ARRAYOFDIMENSION 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.

In the ARRAYOFUSER class, the number of elements in the array (COUNT) is returned, and where in the list the first element comes from (POSITION). In order to get more data, you can repeat the call, setting the position to POSITION + COUNT.

1.1.1.1 Example

If you want to obtain all the dimensions, you would need to start by finding the EMS database you want to search in, and then get the dimensions using the `GETDIMENSIONS` method`(EMS, int POSITION, int COUNT);`

In this call, we download the reference for the EMS database 'emsid' and then download the first 1000 elements.

```
Acubiz.EMS ems = WS.GETEMS("emsid");
Acubiz.ARRAYOFDIMENSION dimensions = WS.GETDIMENSIONS(ems, 1, 1000);
```

If you want to download all the dimensions, you can change the call to:

```
Acubiz.EMS ems = WS.GETEMS("emsid");
Acubiz.ARRAYOFDIMENSION dimensions = WS.GETDIMENSIONS(ems, 0, 0);
```

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 USERS

These functions are used to find one or more users, to set up a user or update a user.

Data about the user is carried in the USER class, which is returned when you look up, set up or update a user.

If you need to modify a user, first look up the user and then use the USER object, which is returned to update the data you need to change. Finally, the USER object is returned with any error codes.

If you need to retrieve several users, an ARRAYOFUSER class is returned, which contains an array with USER objects and data about how many are being returned and where from in the overall list.

Methods

- `FINDUSER(string)`
- `GETUSERS(int, int)`
- `CREATEUSER(string, string, string, string, string, bool, bool, bool)`
- `UPDATEUSER(Acubiz.USER)`

Classes

- USER
  - FirstName : String
  - LastName : String
  - Initials : String
  - Email : String
  - EmployeeID : String
  - DefaultEMS : EMS
  - AcessableEMS : arrayOfEMS
  - Authoriser : User
  - AltAuthoriser : User
  - Secretary : User
  - Booker : User
  - ApproverLimit : Double

- CashAccount : String
- SalaryAccount : String
- IsAuthrizer : Boolean
- IsSecretary : Boolean
- IsBooker : Boolean
- IsFinance : Boolean
- IsAdmin : Boolean
- IsPreApprover : Boolean
- IsFinanceTime : Boolean
- IsEmployee : Boolean
- IsError : Boolean
- Errorstring : String

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

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 DIMENSIONS

These methods and classes are used to retrieve, update and set up dimensions.

Common to all these methods is that you also need to send an EMS object in order to tell the method which EMS database you want to look in.

Information about the user is carried in a DIMENSION class, which is returned when you look up the dimension, set up a dimension or update a dimension.

If you want to modify a dimension, first you need to look up the dimension, then you need to use the DIMENSION object that is returned to update the relevant data. Finally, the DIMENSION object is returned and an updated DIMENSION object is received with any error codes.

If you want to retrieve several dimensions, you need to return an ARRAYOFDIMENSION class, which in addition to an array with DIMENSION objects also contains information about how many are returned and from where in the overall list.

Methods

- `GETDIMENSION(Acubiz.EMS, string)`
- `GETDIMENSIONS(Acubiz.EMS, int, int)`
- `GETDIMENSIONSBYTYPE(Acubiz.EMS, int, int, string)`
- `CREATEDIMENSION(Acubiz.EMS, string, string, string, string, string, string)`
- `UPDATEDIMENSION(Acubiz.DIMENSION)`

Classes

- DIMENSION
  - ID : String
  - ParentID : String
  - DisplayValue : String
  - NameValue : String
  - ID2 : String
  - ID3 : String
  - IsError : Boolean
  - Errorstring : String
  - EMS : EMS

- ARRAYOFDIMENSION
  - 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

METHODS AND CLASSES FOR HANDLING IMPORT DATA

Method

- `IMPORTCSVDATA(Acubiz.EMS, string, string)`
- `IMPORTXMLDATA(Acubiz.EMS, string, string)`

USERS: READING, UPDATING AND SETUP

You can also list, update and set up users on the system.

This is done via the USER class and the functions linked to the SessionClient class (See: Access via .NET, page 3).

LISTING USERS

Listing users is done via the following functions:

- `FINDUSER(string)`
- `GETUSERS(int, int)`

FINDUSER

FINDUSER is used to look up a given user using their initials or EmployeeID. This function returns a USER object with data if a user was found, and a null object if no user was found.

1.1.1.2 Parameters

- Initals (String): Initials of the user you want to have returned.

1.1.1.3 Response

- USER object with data for the user enquired about

1.1.1.4 Example

The function is called as follows:

```
Acubiz.USER User = WS.FINDUSER(”initialer”);
```

The USER object that is returned can be modified and used together with the UpdateUser function. Here the same USER object is given as a parameter for the UpdateUser function.

GETUSERS

GETUSERS is used to find all users for any bulk handling.

This function returns an array of USER objects, packaged in the ARRAYOFUSERS object.

1.1.1.5 Parameters

- Position (long): Where in the overall user list you need to read from. If 0, reading starts from the beginning
- Count (long): How many elements need to be read

1.1.1.6 Response

- ARRAYOFUSER object

1.1.1.7 Example

The function is called as follows:

```
Acubiz.ARRAYOFUSER Users = WS.GETUSERS(0, 0);
```

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

SETUP OF USERS

Users are set up via the following function:

- `CREATEUSER(string, string, string, string, string, bool, bool, bool)`

CREATEUSER

CREATEUSER sets up a user and returns the USER object that has been set up, applying the information that has been provided and the default values that are configured in the system.

The USER object that is then returned can be modified and used as a parameter for the UPDATEUSER function, to change/set any properties that were not set during actual setup.

1.1.1.8 Parameters

- Initials (String): User initials
- Firstname (String): User's first name
- Lastname (String): User's surname
- Email (String): User's e-mail address
- EmployeeID (String): The user's employee identification number
- IsAuthrizer (Boolean): Set to true if the user is an approver
- IsSecretary (Boolean): Set to true if the user is a secretary
- IsBooker (Boolean): Set to true if the user is a booker

1.1.1.9 Response

- USER object with the user that has been setup. If an error is made during setup, the object's ISERROR property would have been set to true and the ERRORSTRING would contain the errors found.

1.1.1.10 Example

The function is used as follows:

Input is provided via a textBox from the user interface and forwarded to the setup function. Checks for errors are then made.

Finally, admin, finance and the pre-approver sets flags on the new user, who is then updated. The response is then checked for errors again.

```
Acubiz.USER User = WS.CREATEUSER(    textBoxUserInitials.Text,
                    textBoxUserGivenName.Text,
                    textBoxUserSurName.Text,
                    textBoxUserEmail.Text,
                    "",
                    checkBoxUserAuthorizer.Checked,
                    checkBoxUserSecretary.Checked,
                    checkBoxUserBooker.Checked);
```

```
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);
      }
```

UPDATING USERS

Updating users is done via the function:

- `UPDATEUSER(Acubiz.USER)`

UPDATEUSER

UPDATEUSER receives a USER object containing the user that needs updating in the system.

The USER object, which is sent to UPDATEUSER, must previously have been received either through user setup or via a search function. If the USER function doesn't originate from the API's other functions, the UPDATEUSER will reject the update.

1.1.1.11 Parameters

- User (USER): The USER object from the user that needs updating.

1.1.1.12 Response

- User (USER): The updated user with any error information.

1.1.1.13 Example

See example of CREATEUSER page 11

DIMENSIONS: READING, UPDATING AND SETUP

It is possible to read, update and set up dimensions in each of the associated EMS databases that are set up in the system.

To access the dimensions in the EMS databases, you need an action for the EMS database you want to interact with.

SEARCHING VIA EMS DATABASES

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

GETDEFAULTEMS

GETDEFAULTEMS returns the EMS database that has been set up as the standard for the user who is logged into the SOAP API,

if you are using the SOAP API frontend.

1.1.1.14 Parameters

- The function doesn't receive any parameters.

1.1.1.15 Return value

- EMS: An EMS object belonging to the user's default EMS database is returned.

1.1.1.16 Example

In order to get gold of the authenticated SOAP Api user's EMS, the following is called:

```
Acubiz.EMS ems = WS.GETDEFAULTEMS();
```

GETEMS

GETEMS returns the EMS database that is being enquired about as a parameter for the function.

1.1.1.17 Parameters

- EmsID (String): ID identification for the EMS database that needs to be returned.

1.1.1.18 Return value

- EMS: the EMS object that is returned belongs to the EMS database that has been provided as the parameter. If no database is found with the ID provided then NULL will be returned.

1.1.1.19 Example

To retrieve an EMS database with the name 'marketing', the following is called:

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

GETEMSUNITS

This function returns all the registered EMS databases in the system.

1.1.1.20 Parameters

- The function doesn't receive any parameters.

1.1.1.21 Return value

- ARRAYOFEMS object

1.1.1.22 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);
}
```

LISTING DIMENSIONS

The following functions are used to list dimensions:

- `GETDIMENSION(Acubiz.EMS, string)`
- `GETDIMENSIONS(Acubiz.EMS, int, int)`
- `GETDIMENSIONSBYTYPE(Acubiz.EMS, int, int, string)`

GETDIMENSION

GETDIMENSION finds and returns a single dimension based on the dimension's ID.

1.1.1.23 Parameters

- EMS (EMS): Action for the EMS database you want to search in
- ID (String): ID of the dimension that needs to be returned

1.1.1.24 Return value

- DIMENSION object

1.1.1.25 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”);
```

GETDIMENSIONS

GETDIMENSIONS retrieves all those dimensions in the EMS database that have been provided as parameters.

1.1.1.26 Parameters

- EMS (EMS): Action for the EMS database you want to search in
- Position (long): Where to read from in the overall list of dimensions. If 0, reading starts from the beginning
- Count (long): How many elements need to be read

1.1.1.27 Return value

- ARRAYOFDIMENSIONS object

1.1.1.28 Example

This example builds on our previous listView of EMS databases and populates a listview with the dimensions that are returned.

The selected EMS database in the listViewEMS is found and used as an EMS database to retrieve all the dimensions.

Next, the number of dimensions is written in a label and each dimension is reviewed and written to a listView.

```
Acubiz.EMS ems = (Acubiz.EMS)listViewEMS.SelectedItems[0].Tag;
Acubiz.ARRAYOFDIMENSION dimensions = WS.GETDIMENSIONS(ems, 0, 0);
labelDimFound.Text = "Dimensions found: " + dimensions.COUNT.ToString();


ListViewItem item = null;
listViewDim.Items.Clear();


string[] arr = new string[5];


foreach (Acubiz.DIMENSION dimension in dimensions.ARRAY)
{
   arr[0] = dimension.ID;
   arr[1] = dimension.NAMEVALUE;
```

```
            arr[2] = dimension.DISPLAYVALUE;
            arr[3] = dimension.ID2;
            arr[4] = dimension.ID3;

            item = new ListViewItem(arr);
            listViewDim.Items.Add(item);
        }

}
```

This example retrieves all the dimensions in the EMS database "marketing".

```
Acubiz.EMS ems = WS.GETEMS(“marketing”);
Acubiz.ARRAYOFDIMENSION dimensions = WS.GETDIMENSIONS(ems, 0, 0);
```

GETDIMENSIONSBYTYPE

GETDIMENSIONS retrieves all those dimensions of a specific type in the EMS database that has been provided as parameters.

1.1.1.29 Parameters

- EMS (EMS): Action for the EMS database you want to search in
- Position (long): Where to read from in the overall list of dimensions. If 0, reading starts from the beginning
- Count (long): How many elements need to be read
- Type (string): The dimension type you want to retrieve. This tends to be the ID of the dimensions on the top level.

1.1.1.30 Return value

- ARRAYOFDIMENSIONS object

1.1.1.31 Example

This example retrieves all the dimensions in the EMS database "marketing" of the type "Dim1":

```
Acubiz.EMS ems = WS.GETEMS(“marketing”);
Acubiz.ARRAYOFDIMENSION dimensions = WS.GETDIMENSIONS(ems, 0, 0, “Dim1”);
```

SETTING UP DIMENSIONS

To set up a dimension, use the following function:

- `CREATEDIMENSION(Acubiz.EMS, string, string, string, string, string, string)`

CREATEDIMENSION

CREATEDIMENSION sets up a user and returns the DIMENSION object that has been set up with the information provided.

The DIMENSION object that is returned can be modified and used as a parameter for the UPDATE DIMENSION function, to change/set the properties that weren't set during actual setup.

1.1.1.32 Parameters

- EMS (EMS): Action for the EMS database you want to set up in
- ID (String): ID of the dimension you want to set up. The ID must be unique in the EMS database.
- Parent ID (String): ID of the dimension that lies above
- Name (String): Name of dimension
- Display (String): Name to display the dimension
- ID2 (String): 2. key to dimension
- ID3 (String): 3. key to dimension

1.1.1.33 Response

- DIMENSION object with the user who has been set up. If an error is made during setup, the object's ISERROR property would have been set to true and the ERRORSTRING would contain the error found.

1.1.1.34 Example

The function is used as follows:

```
Acubiz.DIMENSION dimension = WS.CREATEDIMENSION(ems, textBoxDimID.Text, textBoxDim-
Level.Text, textBoxDIMName.Text, textBoxDimDisplay.Text, textBoxDimID2.Text, text-
BoxDIMID3.Text);
```

UPDATING DIMENSIONS

To update dimensions, use the following function:

- `UPDATEDIMENSION(Acubiz.DIMENSION)`

UPDATEDIMENSION

UPDATEDIMENSION receives a DIMENSION object with the dimension that needs updating in the system.

The DIMENSION object that is sent to UPDATEDIMENSION, must previously have been received either through setup or via the search function. If the DIMENSION function doesn't originate from the API's other functions, the UPDATDIMENSION will reject the update.

1.1.1.35 Parameters

- Dimension (DIMENSION): The DIMENSION object from the dimension that needs updating.

1.1.1.36 Response

- Dimension (DIMENSION): The updated dimension with any error information.

1.1.1.37 Example

```
Acubiz.DIMENSION dimension = WS.GETDIMENSION(ems, ID);

dimension = WS.UPDATEDIMENSION(dimension);
```

IMPORTING DATA

Importing data is done from import profiles that have already been set up in the system.

Importing data is used to send multiple data at the same time. Here no error messages or validation errors will be returned, apart from the validation that takes place confirming whether the format that has been sent is correct.

IMPORTCSVDATA

IMPORTCSVDATA launches a dataimport and then returns an indication of whether the data set has been accepted for import or not.

A data set is rejected if the Import Profile is incorrect, if the format is incorrect or if there are missing fields or entries in the data format.

1.1.1.38 Parameters

- ImportProfile (String): Export profile name
- ImportType (String): Export type
- ImportData (String): Import data in CSV format

The import format is in CSV format and comes from the relevant setup profile

1.1.1.39 Response

- Boolean: If the ImportData was accepted for import and the import is launched. This is not an indication of whether the actual import is successful or not.

1.1.1.40 Example

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

            Acubiz.EXPORTDATA IMPORTCSVDATA = WS.IMPORTCSVDATA(ems, textBoxImportPro-
file.Text, textBoxImportType.Text, CSV);
```

IMPORTXMLDATA

IMPORTXMLDATA launches a data import and then returns an indication about whether the data set was accepted for import or not.

A data set is rejected if the Import Profile is incorrect, if the format is incorrect or if there are missing fields or entries in the data format.

1.1.1.41 Parameters

- ImportProfile (String): Export profile name
- ImportType (String): Export type
- ImportData (String): Import data in XML format

The import format is in XML format, and comes from the relevant setup profile.

1.1.1.42 Response

- Boolean: If the ImportData was accepted for import and the import is launched. This is not an indication of whether the actual import is successful or not.

1.1.1.43 Example

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

Acubiz.EXPORTDATA IMPORTXMLData = WS.IMPORTCSVDATA(ems, textBoxImportPro-
file.Text, textBoxImportType.Text, XML);
```

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.

1.1.1.44 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.

1.1.1.45 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.

1.1.1.46 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.

1.1.1.47 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.

1.1.1.48 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.

1.1.1.49 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);
```

INDEX

Acubiz A/S

Bregnerødvej 133C

DK-3460 Birkerød

Tel: +45 70 214