GET SERVER DETAILS FOR A CLIENT

A server is protected by passwords. These can default to `Const.ANON` for an open server, or you can set them to something else. You might want to add the security details for a server to your client application so that the server can be invoked at some later time. To call the server in any case, you need to know the full ip address. If it is password protected, then you also need to know the password. If you are the administrator of the server, then you should also know its admin key. If you do not know these and the server is running with default settings, then it will automatically return its password when asked for it, but it does not return the admin key.

You will then want to save these details in your own program for accessing later. You can use the licas `PasswordHandler` to store any of the password details. The code to ask for these details might look as follows:

```
//some imports that might be required
import org.licas.call.*;
import org.licas.util.*;
import org.jlog2.exception.ExceptionHandler;
import org.jlog2.*;
import org.licas_xml.abs.*;
import org.licas_xml.model.*;

//some variables that might be required
boolean isOK;                       //true if OK
String serverPassword;              //server password
String adminKey;                    //the server admin key
Element serverUri;                  //server uri
MethodInfo methodInfo;              //used to store the method information
CallObject callObject;              //used to make the remote method call
PasswordHandler passwordHandler;    //password handler
```

Each Service is automatically loaded with a password handler and is used to store the passwords for each service that it wants to interact with. You can however add one manually to any other class, when it can be used for the same purpose. Note the default settings, as they are possibly not used.

```
passwordHandler = new PasswordHandler(Const.ANON, Const.ANON);

//retrieve the server url, of the form server URL and Const.HTTPSERVER value.
serverUri = <Handle><U>http://123.4.5.6:8888/</U><S>HttpServer</S></Handle>
```

- Create a new MethodInfo object and fill with the appropriate parameter values. You can also use `MethodFactory.createMethodCall` instead.
- Ask the server for its password using the `getPassword` method. The parameters to this method can be anything if the default classes are used. This is because the default service

implementation always returns its password when asked for and so it does not check the values of the input parameters. The method structure still needs to be correct however.

- Therefore, a password value of 'anon' and an XML description with any content will suffice.

```
methodInfo = new MethodInfo();
methodInfo.setName(MethodConst.GETPASSWORD);        //name of method to invoke
methodInfo.setRtnType(TypeConst.STRING);            //method return type
methodInfo.setServiceURI(serverUri);                //the url of the server
methodInfo.addParam(Const.ANON);                    //any parameter value
methodInfo.addParam(XMLFactory.getInstance().createElement("abc", "abc"));

callObject = new CallObject();
serverPassword = (String)call.call(methodInfo);
```

- The second parameter value can be anything for default settings.
- If you add contracts that require negotiations, then you might need something specific here.
- You are not allowed to ask for any service 'admin key' as this provides access to values that should not be changed by anybody except for the server administrator or owner. You therefore need to know what the server admin key is, or a default
- Solution is to set it to be the same as the server password. If nothing is entered, they default to ‘Const.ANON’.

You can also check if the server has been initialised properly by checking if the enterprise service bus is running.

```
methodInfo = new MethodInfo();
methodInfo.setName(MethodConst.HASESB);
methodInfo.setRtnType(TypeConst.BOOLEAN);
methodInfo.setServiceURI(serverUri);
methodInfo.setServerPassword(serverPassword);
methodInfo.setPassword(serverPassword);
methodInfo.addParam(new ParamInfo(serverPassword));
isOK = ((Boolean)call.call(methodInfo)).booleanValue();
```

If this is OK, then you can store the passwords for later retrieval. Because there might be several servers and they all have the uuid of `Const.HTTPSERVER`, you should store the passwords with the 'whole url' as the server id, to tell them apart.

```
if (isOK)
{
        passwordHandler.addServicePassword(serverUri, serverPassword);
        passwordHandler.addAdminKey(serverUri, adminKey);  //if known
}
```