

AmanoNet API Guide v1.0.0.0

INTEGRATORS MANUAL

IMPRO TECHNOLOGIES

Table of Contents

Document Control.....	2
Revision Control.....	2
AmanoNet API Requirements	3
API Sample Kit	3
API Usage Guide.....	5

The information contained in this document is confidential to Impro technologies (Pty) Ltd and/or its suppliers and Agents. Such information is made available subject to the conditions that, 1 it may not be disclosed to third parties other than employees or consultants who need to have access thereto for the purposes of evaluating Impro Technologies (Pty) Ltd.'s response and negotiating any agreement that may arise from it. You may use the information only for such evaluation and negotiation purposes.

Dated: 07/06/2015
UP SLA: V1.0

Document Control

Item	AmanoNet API Guide v1.0.0.0
Impro Technologies Reference Number	ANUM_06_2015
Authors:	Clive van Eeden
Security Level:	Confidential – Not for redistribution without consent.

Revision Control

Revision	Dated	Revised By	Description of Major Changes
1.0.0.0	2015/06/07	Clive van Eeden	AmanoNet API Guide

The information contained in this document is confidential to Impro technologies (Pty) Ltd and/or its suppliers and Agents. Such information is made available subject to the conditions that, 1 it may not be disclosed to third parties other than employees or consultants who need to have access thereto for the purposes of evaluating Impro Technologies (Pty) Ltd.'s response and negotiating any agreement that may arise from it. You may use the information only for such evaluation and negotiation purposes.

Dated: 07/06/2015
UP SLA: V1.0

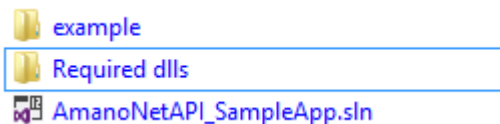
AmanoNet API Requirements

This API sample app has 3 files required to integrate into AmanoNet found in the /lib folder of the project.

1. AmanoNetSDK.dll – you need to add this as a reference to your project
2. EntityFramework.dll and EntityFramework.SqlServer.dll – You don't need to add these as references, but you will need to copy them to your 'Debug' or 'Release' folder as they have to be in the same folder as the AmanoNetSDK.dll file for it to work, as it uses Entity Framework to make database mapping. And also as part of your deployment for your integration program.

API Sample Kit

The sample API Kit is a Visual Studio 2013 project (recommend using Community edition) named 'AmanoNetAPI_SampleApp.sln', and all code for project resides in the 'example' folder:



When you debug the project you will see a Settings section where you will setup the database connection string that points to your AmanoNet database:

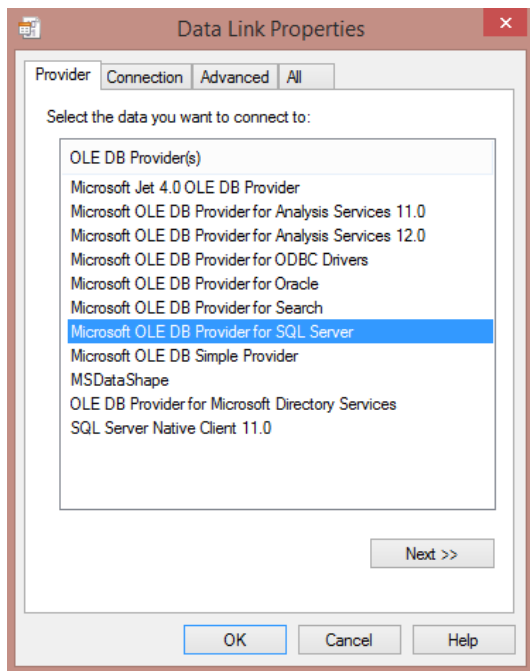


Once you have setup the string it is advised that you copy and paste that string to the Text Property of the 'AmanoNetDbTextBox', that way you will not have to setup each time you debug the project.

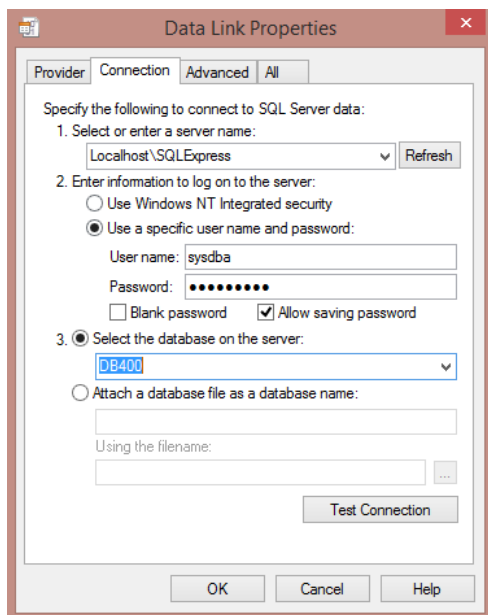
To set up the database connection string, click on the 'Database Setup' button:

Database Setup

Double click on 'Microsoft OLE DB Provider for SQL Server':



Type in server instance information (server name, user name, password and select database) and click OK when done. This will populate the AmanoNetDbTextBox with the connection info.



You will now be able to test the API

The information contained in this document is confidential to Impro technologies (Pty) Ltd and/or its suppliers and Agents. Such information is made available subject to the conditions that, 1 it may not be disclosed to third parties other than employees or consultants who need to have access thereto for the purposes of evaluating Impro Technologies (Pty) Ltd.'s response and negotiating any agreement that may arise from it. You may use the information only for such evaluation and negotiation purposes.

Dated: 07/06/2015
UP SLA: V1.0

API Usage Guide

Before making any calls to the API, you need to declare its usage:

```
12 //using AmanoNetSDK
13 using AmanoNetSDK;
```

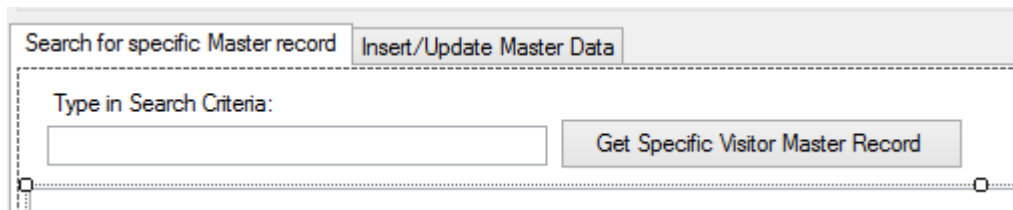
And also declare an instance of it within your class/form:

```
namespace AmanoNetAPI_SampleApp
{
    public partial class Form1 : Form
    {
        //Reference an instance of the API class
        public AmanoNetAPI amanoNetApi;
```

And in your Form_Load event assign your details, passing your app name and the connection details:

```
private void Form1_Load(object sender, EventArgs e)
{
    //on load create new instance of AmanoNetAPI passing your app name and database string of AmanoNet database
    amanoNetApi = new AmanoNetAPI("My App", AmanoNetDbTextBox.Text);
```

The sample app first tab has a search facility:



The screenshot shows a user interface with two tabs at the top: "Search for specific Master record" and "Insert/Update Master Data". The "Search for specific Master record" tab is active. Below the tabs, there is a label "Type in Search Criteria:" followed by a text input field. To the right of the input field is a button labeled "Get Specific Visitor Master Record".

The information contained in this document is confidential to Impro technologies (Pty) Ltd and/or its suppliers and Agents. Such information is made available subject to the conditions that, 1 it may not be disclosed to third parties other than employees or consultants who need to have access thereto for the purposes of evaluating Impro Technologies (Pty) Ltd.'s response and negotiating any agreement that may arise from it. You may use the information only for such evaluation and negotiation purposes.

Dated: 07/06/2015
UP SLA: V1.0

You can type in First name, Last name, middle name or ID Number here and a wildcard search is done like *search value*, so it's best to be specific here or else you will obtain a lot of results, for example:

The screenshot shows the AmanoNet A application window. At the top, there's a 'Settings' section with a text box for 'AmanoNet Database Connection:' containing 'Data Source=LOCALHOST\SQLEXPRESS;Initial Catalog=DB400'. Below this are two tabs: 'Search for specific Master record' (selected) and 'Insert/Update Master Data'. Under the 'Search for specific Master record' tab, there's a 'Type in Search Criteria:' label and a text box containing 'bob'. To the right of the text box is a button labeled 'Get Specific Visitor Master Record'. Below the search criteria, the results are displayed in a list box. The results show three master records, each with a title, first name, middle name, last name, suffix, ID number, gender, pin, master type, current status, site SLA number, site name, and user profile number.

Master Record 0:
MST_SQ = 1
Title = Mr
First Name = Bob
Middle Name = Charles
Last Name = Hope
Suffix =
ID Number = 1
Gender = M
Pin =
Master Type = 1
Current = 1
Site SLA No = 01000000
Site Name = Site: 01000000
User Profile No =

Master Record 1:
MST_SQ = 2
Title = Mr
First Name = Bobby
Middle Name = newton
Last Name = Naidoo
Suffix =
ID Number = 2
Gender = M
Pin =
Master Type = 1
Current = 1
Site SLA No = 01000000
Site Name = Site: 01000000
User Profile No =

Master Record 2:
MST_SQ = 6
Title = Mr
First Name = Bob
Middle Name = Charle
Last Name = Hope
Suffix =

The method for this can be found on the 'GetSpecificMasterRecordButton_Click':

```
//Here is an example of doing a search by Firstname/Lastname or Middlename
private void GetSpecificMasterRecordButton_Click(object sender, EventArgs e)
{
```

Assign 'amanoNetApi.SearchForSpecificMasterRecord' to a string:

```
String GetSpecificMasterRecord = amanoNetApi.SearchForSpecificMasterRecord(SearchCriteriaTextBox.Text);
```

Then you can use 'ParseJSON' method from the AmanoNetSDK ParseHelpers class as below:

```
for (int i = 0; i < GetSpecificMasterRecord.ParseJSON<MASTER>().Count(); i++)
{
    GetSpecificMasterRecordResponseTextBox.AppendText(
        "Master Record " + i.ToString() + ":" + Environment.NewLine +
        "    MST_SQ = " + GetSpecificMasterRecord.ParseJSON<MASTER>()[i].MST_SQ.ToString() + Environment.NewLine +
        "    Title = " + GetSpecificMasterRecord.ParseJSON<MASTER>()[i].MST_Title.ToString() + Environment.NewLine +
```

The information contained in this document is confidential to Impro technologies (Pty) Ltd and/or its suppliers and Agents. Such information is made available subject to the conditions that, 1 it may not be disclosed to third parties other than employees or consultants who need to have access thereto for the purposes of evaluating Impro Technologies (Pty) Ltd.'s response and negotiating any agreement that may arise from it. You may use the information only for such evaluation and negotiation purposes.

Dated: 07/06/2015
UP SLA: V1.0

On the MSTSQ_TextBox_LostFocus method, there is another search example. If you type in an existing MST_SQ number from the database, it will return data for you to populate fields you need:

```
//Here is an example of calling MASTER data associated with the database primary key (MST_SQ).  
//Type a number in 'Database MST_SQ' field, click anywhere else and the data will populate  
private void MSTSQ_TextBox_LostFocus(object sender, EventArgs e)  
{
```

The API method for this is 'amanoNetApi.GetMasterRecordUsingMSTSQ' and assign to a string:

```
String GetMasterRecord = amanoNetApi.GetMasterRecordUsingMSTSQ(MSTSQ_TextBox.Text);
```

Take note of how this information is parsed and used:

```
for (int i = 0; i < GetMasterRecord.ParseJSON<MASTER>().Count(); i++)  
{  
    TitleComboBox.Text = GetMasterRecord.ParseJSON<MASTER>()[i].MST_Title.ToString();  
    FirstNameTextBox.Text = GetMasterRecord.ParseJSON<MASTER>()[i].MST_FirstName.ToString();  
    MiddleNameTextBox.Text = GetMasterRecord.ParseJSON<MASTER>()[i].MST_MiddleName.ToString();  
    LastNameTextBox.Text = GetMasterRecord.ParseJSON<MASTER>()[i].MST_LastName.ToString();  
    IDNumberTextBox.Text = GetMasterRecord.ParseJSON<MASTER>()[i].MST_ID.ToString();  
    if (GetMasterRecord.ParseJSON<MASTER>()[i].MST_Gender.ToString() == "M")  
        GenderComboBox.SelectedIndex = 0;  
    else  
        GenderComboBox.SelectedIndex = 1;  
    if (GetMasterRecord.ParseJSON<MASTER>()[i].MST_Type.ToString() == "0")  
    {  
        MasterTypeComboBox.SelectedIndex = 0;  
        EmployeeNumberTextBox.Text = GetMasterRecord.ParseJSON<EMPLOYEE>()[i].EMP_EmployeeNo.ToString();  
        EmployeeEmployerTextBox.Text = GetMasterRecord.ParseJSON<EMPLOYEE>()[i].EMP_Employer.ToString();  
        EmployeePositionTextBox.Text = GetMasterRecord.ParseJSON<EMPLOYEE>()[i].EMP_Position.ToString();  
        DepartmentComboBox.Text = GetMasterRecord.ParseJSON<EMPLOYEE>()[i].DEPT_No.ToString();  
        DepartmentNameLabel.Text = GetMasterRecord.ParseJSON<DEPARTMENT>()[i].DEPT_Name.ToString();  
    }  
    else  
    {  
        MasterTypeComboBox.SelectedIndex = 1;  
        DepartmentComboBox.Text = GetMasterRecord.ParseJSON<VISITOR>()[i].DEPT_No.ToString();  
        DepartmentNameLabel.Text = GetMasterRecord.ParseJSON<DEPARTMENT>()[i].DEPT_Name.ToString();  
    }  
    if (GetMasterRecord.ParseJSON<MASTER>()[i].MST_Current.ToString() == "1")  
        MasterCurrentComboBox.SelectedIndex = 0;  
    else  
        MasterCurrentComboBox.SelectedIndex = 1;  
    MasterSiteComboBox.Text = GetMasterRecord.ParseJSON<MASTER>()[i].SITE_SLA.ToString();  
    SiteNameLabel.Text = GetMasterRecord.ParseJSON<SITE>()[i].SITE_Name.ToString();  
    if (GetMasterRecord.ParseJSON<MASTER>()[i].MT_NO.ToString() == "1")  
        ClockingTypeComboBox.SelectedIndex = 0;  
    else  
        ClockingTypeComboBox.SelectedIndex = 1;
```

Some use ParseJSON<MASTER>, some ParseJSON<EMPLOYEE>, some ParseJSON<VISITOR> and some ParseJSON<DEPARTMENT>. This is because this information comes from different database tables and what you get back is heavily dependent on the master type (e.g. Employee type =0 or Visitor type = 1). The example shows you how to decide on which to use from the information that is received.

Updating and Inserting data is shown via the 'InsertMasterDataButton_Click' method.

Depending on whether you are doing an Employee Type or Visitor Type, this will show you how to prepare the data. Before even using the API insert/update methods you need to use the API Build models methods, to ensure that the data is presented to the API correctly.

Master data is to be built regardless, this is done using the 'amanoNetApi.BuildMaster' method as shown:

```
//Build up MASTER table values, assign values to the amanoNetApi.BuildMaster method from the API.
//We do this to make sure our format is correct before doing Insert/Update...
var Master_Record = amanoNetApi.BuildMaster(int.Parse(MSTSQ_TextBox.Text), TitleComboBox.Text, FirstNameTextBox.Text, MiddleNa
null, //MST_PIN is a nullable field, but you can populate this with an integer if
MST_Type, MST_Current, MasterSiteComboBox.Text,
null, //USRPRF_NUM is a nullable field, but you can populate this with an integer
"", //because MST_CDATE is a string, it cannot be nullable from code, but you can p
MT_NO //MT_NO can be a nullable field, but it is recommended to have the clocking
);
```

Then you decide whether you are doing an employee or visitor, do build up employee use 'amanoNetApi.BuildEmployee',

```
//Build up EMPLOYEE table values using amanoNetApi.BuildEmployee
var Employee_Record = amanoNetApi.BuildEmployee(int.Parse(MSTSQ_TextBox.Text), EmployeeNumberTextBox.Text, EmployeeEmpo
```

for visitor use 'amanoNetApi.BuildVisitor'.

```
//Build up for VISITOR table values using amanoNetApi.BuildEmployee
var Visitor_Record = amanoNetApi.BuildVisitor(int.Parse(MSTSQ_TextBox.Text), VisitorCompanyTextBox.Text, 0, Convert.ToInt32(Departme
null //HOSTED_BY is a nullable field, you would need to know the MST_SQ of the host
);
```

Once this is done you can then run the relevant Insert/Update method.

For Employees its 'amanoNetApi.UpdateInsertMasterEmployee':

```
//Now do Insert/Update by calling amanoNetApi.UpdateInsertMasterEmployee and using varia
returnValue = amanoNetApi.UpdateInsertMasterEmployee(Master_Record, Employee_Record);
```

And for Employees its 'amanoNetApi.UpdateInsertMasterVisitor':

```
//Now do Insert/Update by calling amanoNetApi.UpdateInsertMasterVisitor and using variab
returnValue = amanoNetApi.UpdateInsertMasterVisitor(Master_Record, Visitor_Record);
```

'returnValue' is a string variable that will return whether you insert/update was a success or failure with the relevant database MST_SQ primary key. As seen in the response below:

The screenshot shows a web application window titled "AmanoNet API Sample". It has a "Settings" section at the top with a "Database Connection" field containing a SQL Server connection string and a "Database Setup" button. Below this is a tabbed interface with "Search for specific Masterrecord" and "Insert/Update Master Data". The "Insert/Update Master Data" tab is active, showing a "Master Table API" section. This section has a "Master Data" sub-section with fields for "Database MST_SQ" (2), "Current" (True), "Site" (01000000), "Clocking Type" (Access and Time), "Title" (Mr), "Gender" (Male), "First Name" (Bobby), "Middle Name" (Singer), "Last Name" (Naidoo), and "ID Number" (2). There is also a "Master Type" section with "Master Type" (Contractor/Visitor) and "Department" (1). Below these are fields for "Employee Number" (c123444), "Employee Employer" (F), and "Employee Position" (D). A "Contractor/Visitor" section has "Visitor Company" (Company Name), "Visitor Hosted" (False), and "Host MST_SQ" (0). A large blue button labeled "Insert Master Data to Visitor Portal" is at the bottom. The "Response" section at the very bottom shows the text: "MASTER=Success - Update Master Record MST_SQ 2 VISITOR=Success - Update Master Visitor Record MST_SQ 2".

There are lots of comments in the sample app code to help guide you if you need more understanding on the process.