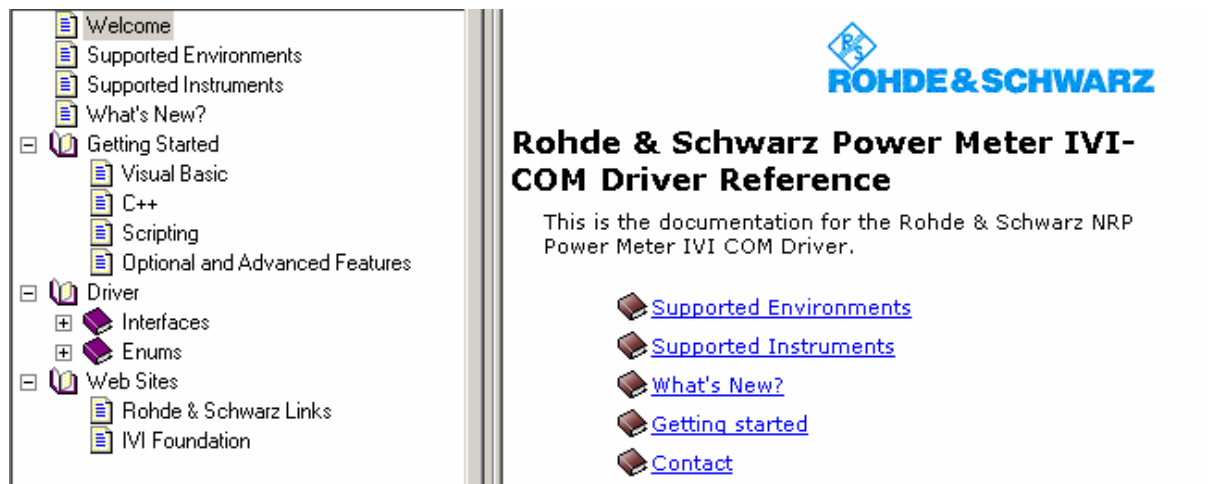# How to use the NRP IVI-COM driver

Contents

1   Overview

This document describes how to use the RsNrp IVI COM driver in various programming environments.
(The same information is also available in the online documentation, which,
in addition, also contains a detailed description of all the instrument
specific functions).

For general information about the IVI terms and features please use
the original IVI documentation, www.ivifoundation.org.
Technical data of the driver may be found in the Readme file.

2   Additional Help

In addition, a extra documentation is also included in compressed HTML format (Windows CHM help file) and stored together with the driver in the c:\Program\IVI\rsnrp.chm directory.

3   Using the IVI-COM Driver in Microsoft Visual Basic

This chapter describes step by step how to use the IVI driver in Visual Basic.

3.1   Adding a reference to the IVI COM driver objects

Create a new Visual Basic project.

Select the Visual Basic menu item "Project/References".

In the list box, locate the line "IVI RsNrp (Rohde & Schwarz) X.X Type Library" denoting the R&S Nrp IVI driver for the instrument, set the check mark in the checkbox.
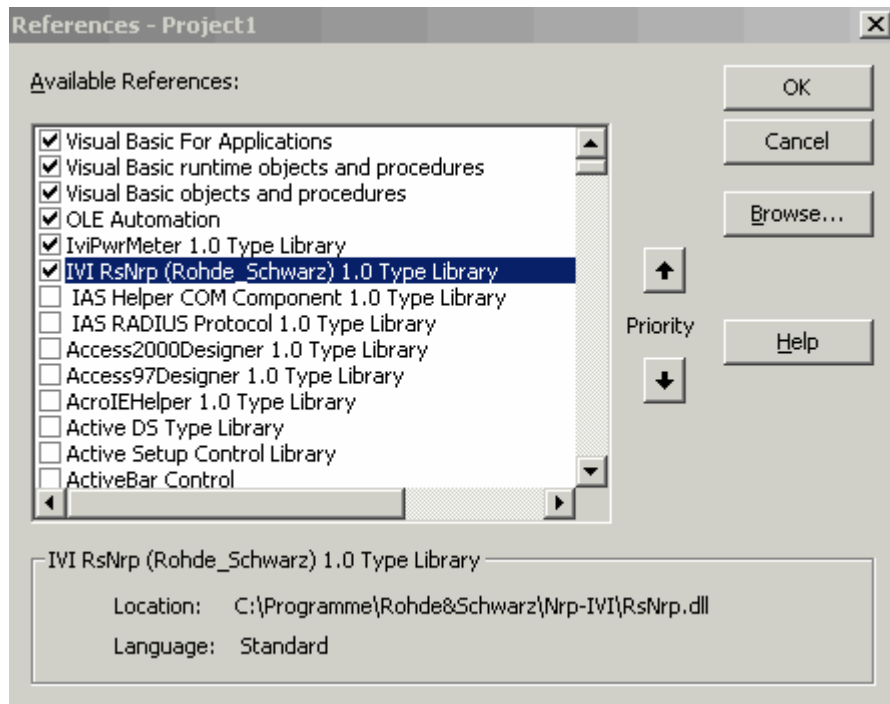
If you intend to use the *IviPwrMeter* class compliant interfaces, also check the line "IviPwrMeter X.X type library".

3.2 Creating an instance of the IVI COM driver object

At the top of the Visual Basic source code window type in

```
Dim Obj As New
```

After entering a space after "New" there appears a drop down listbox offering the available data types. Select the data type "RsNrp". The methods and properties of the NRP specific driver are now accessible via the object variable "Obj". (There are more portable methods to create a driver object instance, see below).

3.3 Using the Driver

To initialize the driver, the function "Obj.Initialize()" is to be called. This may be done in the Visual-Basic function Form_Load() or another appropriate place in the source code.
In the Visual Basic source code window type in "Call Obj.". In the drop-down box locate the "Initialize" function and press the Tab key. Then enter the opening bracket '(' for the argument list.
The first argument of Initialize() is the VISA resource descriptor, a string like "GPIB0::20::INSTR". The IdQuery and Reset arguments may both be set to "False", the option string may be empty. The source code for a minimal test program looks like this

```
Call Obj.Initialize("GPIB0::20::INSTR", False, False, "")
Dim bSuccess As Boolean
bSuccess = Obj.Initialized
Obj.Channels.Item("1").CorrectionFrequency = 987654321
...
Call Obj.Close
```

4 Using the IVI-COM Driver in C++

This chapter describes step by step how to use the IVI driver in C++.

4.1 Importing the IVI COM driver interfaces

Add the following lines at the top of the CPP file:

```
#define _WIN32_DCOM
#include <objbase.h>
#include <atlbase.h>

#import "IviDriverTypeLib.dll"
#import "IviPwrMeterTypeLib.dll"
#import "RsNrp.dll"

using namespace RsNrp_Rohde_SchwarzLib;
```

4.2 Initializing COM

After initializing COM the drivers functions may be called.
A simple test is:

```
void run()
{
    IRsNrpPtr Obj;
    HRESULT hr(Obj.CreateInstance(__uuidof(RsNrp)));
    Obj->Initialize("GPIB0::20::INSTR", VARIANT_FALSE, VARIANT_FALSE, "");
    if (Obj->GetInitialized())
    {
      IRsNrpChannelPtr NrpChannel = Obj->Channels->GetItem("1");
      NrpChannel->put_CorrectionFrequency(123456789);
    }
    Obj->Close();
}
```

4.3 Using the Driver

After initializing COM the drivers functions may be called.
A simple test is:

```
void run()
{
    IRsNrpPtr Obj;
    HRESULT hr(Obj.CreateInstance(__uuidof(RsNrp)));
    Obj->Initialize("GPIB0::20::INSTR", VARIANT_FALSE, VARIANT_FALSE, "");
    if (Obj->GetInitialized())
    {
      IRsNrpChannelPtr NrpChannel = Obj->Channels->GetItem("1");
      NrpChannel->put_CorrectionFrequency(123456789);
    }
    Obj->Close();
}
```

The essential steps are to retrieve a pointer `Obj` to a driver object, and then to `Initialize()` and finally `Close()` the driver.
The first argument of the `Initialize()` function is the VISA resource descriptor, a string like "GPIB0::20::INSTR". The *IdQuery* and *Reset* arguments may both be set to *VARIANT_FALSE*, the *Option String* may be empty.

5 Using the IVI-COM Driver in Scripting Environments

The IVI COM driver may also be used in MS Windows Scripting Host VBS (Visual Basic Script) or JS (Java Script) scripts.
This is a quick and simple way to control the instrument. The disadvantages are that functions with array arguments cannot be accessed, and that normal text editors do not support intellisense. (A script debugger can be downloaded from the Microsoft Scripting web page).
This page describes step by step how to use the IVI driver in a Visual Basic Script.

5.1 Creating the script file

Use Notepad or another ASCII editor to create a new file, add the lines

```
    Set Obj = CreateObject("RsNrp.RsNrp.1")
    Call Obj.Initialize("GPIB0::20::INSTR", False, False, "")
    If Obj.Initialized Then
        MsgBox "Success"
        MsgBox "Nrp.Frequency = " &
Obj.Channels.Item("1").CorrectionFrequency
    Else
        MsgBox "Error"
    End If
    Call Obj.Close
```

The argument string for CreateObject is the ProgId of the driver.
The first argument for `Initialize()` is the Visa resource descriptor, an example is "GPIB0::20::INSTR".
Save the file under the name *RsNrp.vbs*.

5.2 Executing the script file

To execute the script, enter the file name at the command prompt or double-click on the file symbol in the explorer.

5.3 Using the class driver interface

To use the class compliant IIviRFSigGen interface for accessing the hardware, first create a R&S SMU
To exclusively use the class compliant IIviPwrMeter interface for accessing the hardware, first create a NRP specific driver object as described above, then extract a class compliant interface object,

```
    Dim ClsObj
    Set ClsObj = Obj.IviPwrMeter()
```

The class compliant functions for the NRP can now be accessed via the IIviPwrMeter object ClsObj.

6 Optional and Advanced Features

The IVI driver architecture contains a lot of generic functionality.
The IVI *"Inherent Capabilities"* are available under the [Root Interface](#), or under the [IIviDriverOperation](#) and [IIviDriverUtility](#) interfaces.
Some features are described below. For a comprehensive description please refer to the function descriptions in this manual or to the original [IVI documentation](#).

6.1 Hardware Simulation

Simulation of the hardware can be enabled by setting a flag in the last argument of the [Initialize()](#) function (the *OptionString*).

```
[Visual Basic and VB Script]
Call Obj.Initialize("GPIB0::20::INSTR", False, False, "Simulate=True")
```

```
[C++]
Obj->Initialize("GPIB0::20::INSTR", VARIANT_FALSE, VARIANT_FALSE, "Simulate=True")
```

With an *OptionString* "Simulate=True" there won't be any accesses to the hardware and the driver will return simulated parameter values.

6.2 Range Checking and Coercion Recording

Not supported by the driver.

6.3 The IVI Configuration Store

The first argument of the [Initialize()](#) function need not be a *VISA* resource string. Instead it may also be an *IVI Logical Name* identifying an *IVI Driver Session* in the *IVI Configuration Store* (the central IVI data repository, an XML file).
Every *IVI Driver Session* record in the Configuration Store contains values for the *VISA* resource string and the other `Initialize()` arguments.
In that case the values from the Configuration Store are used in the call of the [Initialize()](#) function. The purpose of this construct is to reconfigure the application without touching the applications source code. Using the IVI Configuration Store requires that the *IVI Shared Components* be installed.

6.4 The IVI Class Factory

Another step to make the application code more generic would be to use the *IVI Class Factory*.
The purpose of the *IVI Class Factory* is to create a driver object from an *IVI Driver Session* name.
Assume that the *IVI Configuration Store* contains a *IVI Driver Session* record with logical name "RsNrp.DriverSession". Then the following code is an alternative to creating the object directly (as described above):

```
[Visual Basic]
Dim Factory As New IviSessionFactory
Dim Obj2 As RsNrp
Set Obj2 = Factory.CreateDriver("RsNrp.DriverSession")
```

```
[C++]
#import "IviSessionFactory.dll"
using namespace IVISESSIONFACTORYLib;
IIviSessionFactoryPtr ObjFac;
HRESULT hr(ObjFac.CreateInstance(__uuidof(IviSessionFactory)));
IRsNrpPtr Obj = ObjFac->CreateDriver("RsNrp.DriverSession");
```

(Using the session factory with Visual Basic requires a reference to *"IviSessioFactory X.X Type Library"* under the menu item /Project/References).

A more useful application of the *IVI Class Factory* would be to use it for creating an *IIviPwrMeter* class driver object:

```
Dim Factory As New IviSessionFactory
Dim ClsObj2 As IIviPwrMeter
Set ClsObj2 = Factory.CreateDriver("RsNrp.DriverSession")
```