SCPI-Recorder

Test Automation at Your Fingertips

Application Note

Products:

- R&S®SMW200A
- R&S®SMA100B

This application note briefly summarizes the history of SCPI and outlines in which fields of application a SCPI based test system software has still advantages over a driver (e.g. IVI) based approach.

Further the unique SCPI related support features of the high-end vector signal generator R&S®SMW200A and the performance leading RF/Microwave analog signal generator R&S®SMA100B are introduced and operating guidelines are provided

Note:

Please find the most up-to-date document on our homepage

ROHDE&SCHWARZ

Application Note

C. Neuhaeusler
10.2017-1GP98_2E

Table of Contents

The following abbreviations are used in this application note for Rohde & Schwarz products:

- The R&S®SMW200A vector signal generator is referred to as SMW.
- The R&S®SMA100B RF and MW analog signal generator is referred to as SMA.

1 Introduction

1.1 History of Test Automation

With the growing use of remote controlled test equipment and test systems in the industry in the 1980s it became evident that the required control software would substantially increase the total costs of the test systems. It was recognized that the proprietary software interface in between the controller and the test instrument which often even differed amongst the devices of one manufacturer, mainly influenced the development and service efforts and consequently the costs of the test solution.

In order to overcome this, the Standard Commands for Programmable Instrumentation (SCPI) consortium specified a software interface language and adopted it as part of the IEEE-488.2 (GBIP) standard in 1990 [6]. During the following years SCPI was increasingly accepted in the market and therefore was specified for additional test interfaces. Today SCPI is available for GPIB, RS-232, VXIbus, Ethernet LANs, and USB. SCPI is not only almost independent from the hardware, but is also supported by all common programming languages and Integrated Development Environments (IDE) used for test automation software development.
SCPI encompasses both 'common commands' supported by all SCPI compatible devices and 'instrument control commands' that are in line with the SCPI syntax but are defined manufacturer- and/or device-specific.

The original advantage of SCPI, its ASCII based syntax, eventually slowed down the development of test software as the test devices became more and more complex and the number of command sequences grew nearly exponentially (e.g. >6000 commands for a high-end vector signal generator).
Each command required to trigger a certain reaction of the device first had to be selected from the programming manual of the respective device. In a second step it had to be transferred without any support from the IDE in use, into the respective programming language applying the correct syntax and semantics.

In 2001, the IVI foundation was established in order to resolve this problem. Both manufacturers for test- and measurement equipment, software, and test systems, and test system integrators and also end users are members of the foundation. R&S is one of the three sponsor members of the foundation and is actively contributing to the standardisation process. Since 2003, the SCPI consortium has been integrated into the IVI foundation as well.
The foundation's objective is to develop a specification of a driver-based software interface that supports the interchange ability of test devices independent of the manufacturer. In addition, the implemented driver functionality (e.g. simulation mode, range check, state caching and auto complete) shall significantly reduce the development and service costs for remote control applications.
Like R&S, all other main manufacturers of test instruments offer so-called IVI drivers. Therefore it is possible to exchange devices belonging to one of the currently defined 13 device classes (e.g. RF signal generator: IviRFSigGen class) independently of the manufacturer, if:

- only IVI class drivers with base class capabilities are used and
- all used / required functions of the device are realised in the respective driver

1.2 Why SCPI is still alive

Unfortunately the conditions for the exclusive operation of IVI drivers described above are not applicable for more complex test scenarios (e.g. digital standards like LTE and 5G NR). The main obstacle of these scenarios are the variety of the test instrument parameters and functions that have to be implemented which will be done in most cases manufacturer and device specific. In this case one of the following options must be chosen:

- an 'IVI Class Compliant Specific Driver' (base class capabilities e.g. IviRFSigGen and class extension capabilities),
- an 'IVI Custom Specific Driver' (only not standardized instrument specific capabilities),
- use of a combination of an IVI driver with base class capabilities and a supplementary SCPI driver module
- a solution based exclusively on SCPI commands

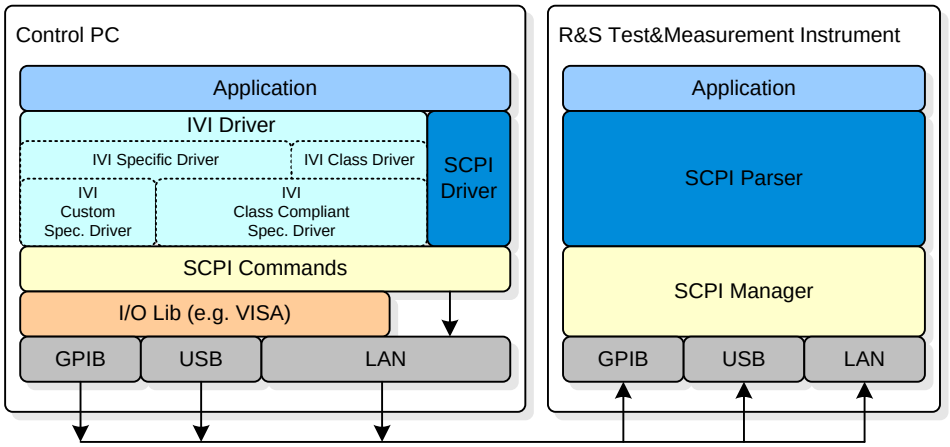


Figure 1: Test Automation Software Stack

Users often choose a solution based exclusively on SCPI commands in this situation. This allows the unlimited access to all functions of the device via one software interface and avoids possible interface conflicts by using two parallel driver solutions. Likewise, the additional software layer of the IVI driver is avoided for test applications in the manufacturing environment as the minimization of the testing time (processing speed is very critical) is of major importance there.

When extending the functionality of test instruments the SCPI commands have to be defined and implemented before transforming them into IVI driver functions. Therefore, SCPI based software solutions are preferred in most cases for control software used in test systems for technologies that have short innovation cycles. This allows the conduct of automated tests already in early stages of the development.

This would bring the users back to the strenuous hunting around the manual to find the right SCPI command and exact syntax and to the time consuming search for semantic or syntactic errors in the code…. **unless they use the SMW or the SMA**!

2 R&S knows Your Test Automation Needs

2.1 Overview

R&S realized early that SCPI based control software will also in the future be indispensable for complex and/or latency critical test scenarios like those mentioned in the previous chapter. Therefore, adequate tools will have to be provided to allow an easy and efficient work with SCPI commands within the most popular IDEs and programming languages.

As a first step, R&S equipped its test and measurement instruments with the 'Help Key'. This key allows requesting background information for each configuration parameter that can be modified manually. Additionally, the parameter related SCPI command and its syntax is shown. Thus the test system developers are enabled to simply transfer each manually performed configuration step directly into SCPI based source code.

Beyond that R&S' point of view is to provide means that allow:

- to find required SCPI commands and the related indices and parameters as quickly as possible,
- an easy access to background information on certain configuration parameters and related SCPI commands,
- the recording and export of SCPI lists,
- to generate source code for the most common programming languages and IDEs based on the recorded SCPI lists.

The **SMW, a high-end vector signal generator** of the most recent generation and the **SMA, the performance leading RF/microwave analog signal generator** include all these unique features that allow an increase in efficiency of the SCPI based test system software development. This achievement has been further extended through the touch screen based concept of operation.

The following figure provides an overview of the integrated SCPI related support features and the used name conventions:

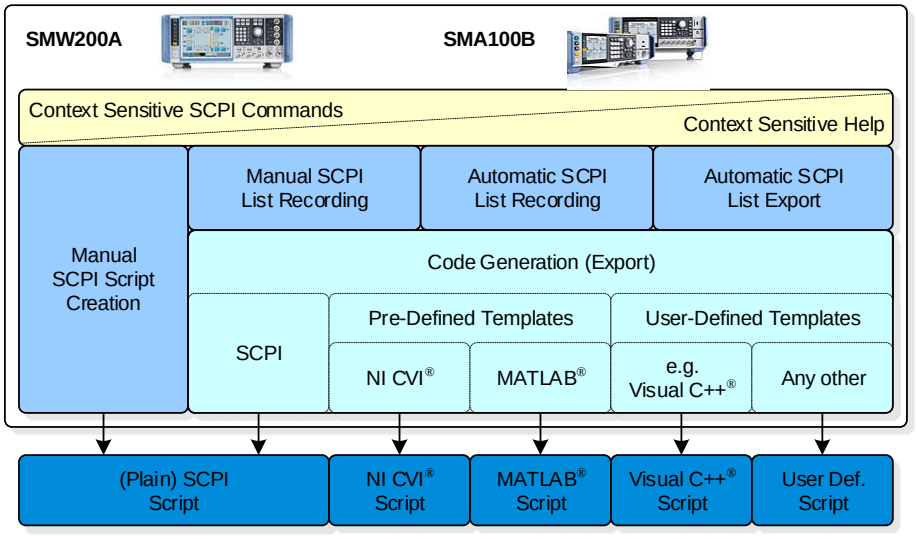


Figure 2: SMW and SMA integrated SCPI related Support Functionality

1. **Find and show SCPI Commands**
   A significant extension of the 'Help Key' functionality was achieved through the touch screen based concept of operation.

   - **Context sensitive Help Function:**
     The background information on certain functions and parameters can be shown directly via a context sensitive menu on the GUI.
   - **Context sensitive SCPI Commands:**
     Additionally, the SCPI command syntax including the indices and parameter values can be requested via the context sensitive menu for each GUI parameter. Thus the time consuming hunting for appropriate SCPI commands belongs definitely to the past. The shown SCPI command string can directly be used in control software without further extensions and/or parameterization.

   For details, see chapter 3 on page 11.

2. **SCPI List and Script Generation**
   Additional functions are integrated in signal generators that allow generating SCPI lists for any signal generator configuration. These functions are tailored for different modes of operation.
   In the context of SCPI recording, a SCPI list may consist of:

   - several SCPI commands in their chronological order of execution
   - all SCPI commands needed for a certain functional setup of the signal generator but not necessarily in the advisable order of their execution.

   These different types of SCPI lists can be generated as follows:

   - **Manual SCPI List Recording:**
     The signal generators are supporting the manual recording of chronological SCPI lists consisting of SCPI commands comprising any number of parameter variations. In this recording mode, the user has to decide deliberately for which parameter the related SCPI command shall be added to the SCPI list. The recorded list can be exported in a final step as a script file in various formats.
     This mode of recording is extremely helpful if a certain configuration has to be worked out and thus not any parameter variation or keystroke but only the final and proper settings should be recorded.
   - **Automatic SCPI List Recording:**
     To further facilitate the generation of SCPI lists, the SCPI-Recorder also supports an automatic mode of operation. In this mode, the user starts the recording process in advance of a certain configuration task and stops it after finalization of all necessary configuration steps. The SCPI commands related to the parameter variations done in between are added to the list automatically in a chronological order without any further user activity. The recorded list can be exported in a final step as a script file in various formats. This recording mode is normally the best choice for fast and convenient SCPI list recording.

- **SCPI List Export:**
  Some users do not require a chronological SCPI list but only a list of SCPI commands related to those configuration parameters differing from the default settings of the signal generator (PRESET). These commands can be identified at any time, compiled into a list and exported in various formats as a script file.
  This functionality is especially helpful if extensive manual configurations have been made without prior activation of the automatic SCPI list recording.
- **Manual SCPI Script Creation:**
  Those users of the signal generators which want to manually create SCPI script files on the fly without using the integrated recording and export functionality will profit from the valuable functions to find and show SCPI commands.

For details, see chapter 4 on page 13.

3. SCPI Script Export and Code Generation

The generated SCPI lists can be either exported directly as an ‘Plain’ (ASCII based) SCPI script or can be converted into a source code script file by using the integrated code generator.

- **Export of Plain SCPI Scripts:**
  In case only the SCPI command strings are needed for the following test automation tasks the recorded SCPI lists can easily be exported as ‘Plain’ (ASCII based) SCPI script files.
- **Export of Source Code Scripts:**
  The SCPI lists can be integrated very easily into source code modules of nearly any programming language with the help of the integrated code generator.
  Pre-defined code templates used to control the code generation process are available in the signal generators for the programming languages most commonly used for test automation software.
  Even in case source code in a programming language not directly supported by the pre-defined code templates is needed the code generator can be used. In this case, a user-defined code template incorporating the coding rules and remote control functions of the respective programming language can be set up and loaded.

For details, see chapter 5 on page 19.

All the introduced functions, which together cover all aspects of SCPI based test automation software development are selected, activated and configured via the touch screen, based GUI. **Test automation will be done at your fingertips!**

With this approach, you will be able to:

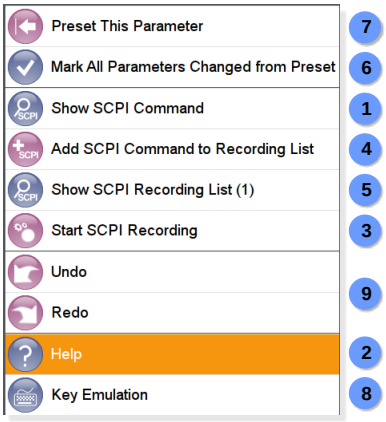
- Speed up your development of test automation systems and thus avoid high costs for developing and maintaining your test system software.
- Keep track with any rapidly evolving technology.

All the tools and functions offered by the SMW/SMA and their operation will be illustrated in detail in the following chapters.

The descriptions will be based on a SMW only since the overall functionality is nearly identical for SMW and SMA.

2.2 Concept of Operations

2.2.1 Context Sensitive Menu



The SMW provides a context sensitive menu, which allows a quite simple access to the auxiliary functions available for all GUI elements.
The context sensitive menu is either directly activated via a touch and hold operation (for at least one second) on the functional blocks or parameter entry fields on the SMW's touch screen based GUI or as usual through a right mouse click after appropriately positioning the mouse pointer.
The number and the kind of auxiliary functions offered depend on the selected GUI element. The auxiliary functions shown by *Figure 3* are for example accessible via the context sensitive menu after selecting a GUI parameter entry field.

Figure 3: GUI-Parameter related Context Sensitive Menu

Amongst others the SCPI commands of the selected GUI parameter and additional parameter specific background information can be requested via the menu:
① Shows the SCPI command to control the selected GUI parameter.
② Activates the Help Menu, which provides in depths information about the selected GUI parameter and the related SCPI command.

In addition, the SCPI recording will be controlled via this menu. This includes the start and stop of a recording task as well as the monitoring of running recordings and the review of already available SCPI recording lists:
③ Starts the automatic recording of all SCPI commands related to manual SMW configuration steps performed thereafter.
④ Adds the SCPI command required to set/alter the selected parameter to the SCPI recording list.
⑤ Shows the current SCPI list, collected during the last manual or automatic recording run or the SCPI list of the currently running recording task.

Another auxiliary function was implemented allowing the user to keep track of all configuration parameters differing from their default value. This is especially helpful in case of complex and extensive SMW configurations:
⑥ 'Mark All Parameters Changed from Preset' will show all those parameters and functional blocks highlighted in orange whose actual value differs from the default value set after the latest SMW preset.
⑦ Allows a selective preset of the selected GUI parameter.
⑧ Opens the key emulation dialog, which is indispensable in case SMW remote operation via VNC.

3 Find and show SCPI Commands

A significant extension of the ‘Help Key’ functionality is available through the touch screen based concept of operation.

3.1 Context sensitive Help Function

The background information on certain functions and parameters can be shown directly via a context sensitive menu on the GUI.
The following example demonstrates based on the 'EUTRA/LTE Frame Configuration' dialog offered by the 'Baseband' blocks how easily background information about certain parameters can be requested and displayed.

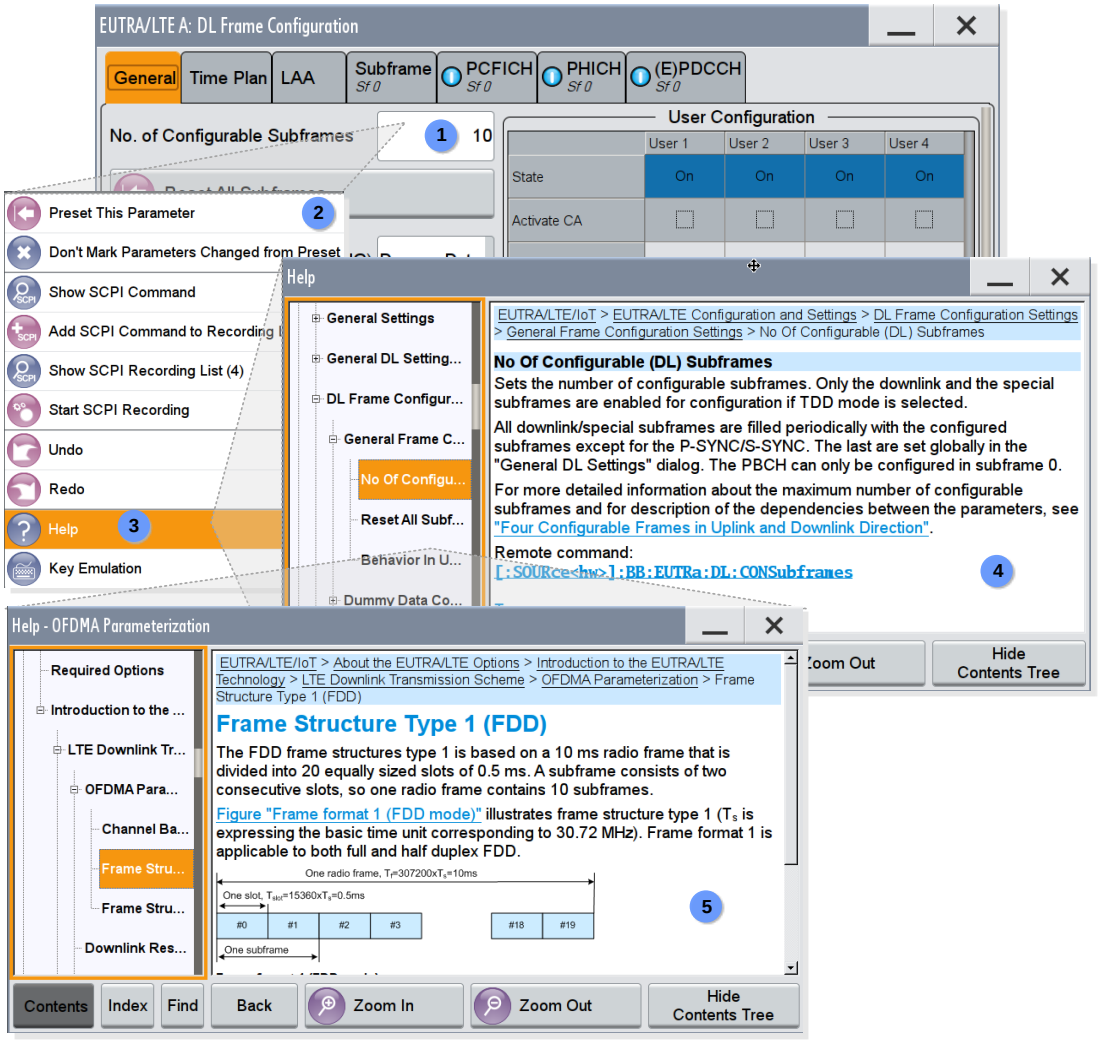


Figure 4: Show Help Information for certain Parameter

① Select the GUI parameter for which background information has to be requested.
② Open the context sensitive menu.

③ Request information via the ‘Help’ function.
④ Apart from the requested background information on a certain GUI parameter the SCPI command syntax is also shown. See chapter 3.2 on page 12 if certain indices and parameter values are needed in addition to the SCPI command syntax.
⑤ Not only parameter related information but also technology (e.g. EUTRA/LTE) specific details can be requested that may have an impact on the overall SMW configuration.

3.2 Context sensitive SCPI Commands

Additionally, the SCPI command syntax including the indices and parameter values can be requested via the context sensitive menu for each GUI parameter.
Thus, the time consuming hunting for appropriate SCPI commands belongs definitely to the past. The shown SCPI command string can directly be used in control software without further extensions and/or parameterization.
The following example shows how to request the SCPI command to set the trigger mode for the ‘Custom Digital Modulation’ in ‘Baseband’ block A to single mode:

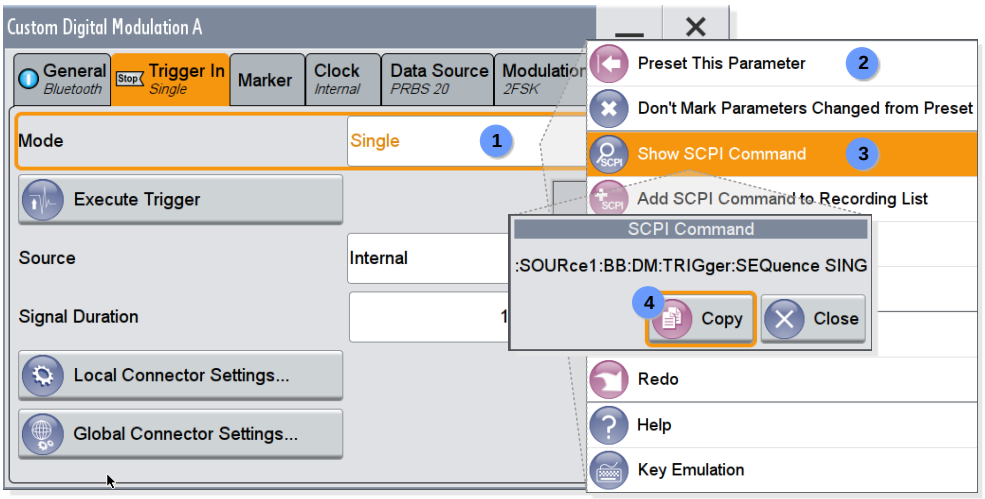


Figure 5: Show SCPI Command for certain Parameter

① Choose the parameter for the required SCPI command.
② Open the parameter specific context sensitive menu.
③ Request the SCPI command.

The displayed SCPI command can be taken over manually into the test automation software or may be used to check the correctness of an already implemented code.

④ To further improve the programming comfort and convenience one may just copy the SCPI command to the clipboard, which may be accessed via a VNC viewer with enabled clipboard synchronization.
This feature makes any SCPI command available on the fly by a simple copy/paste operation!

However it is advisable to use the SCPI command recording functions explained in detail in chapter 4 (starting at page 13), in case a SMW configuration has to be defined through numerous SCPI commands.

4 SCPI List and Script Generation

Additional functions are integrated in the SMW, which allow generating SCPI lists for any signal generator configuration. These functions are tailored for different modes of operation.

In the context of SCPI recording, a SCPI list may consist of:

- several SCPI commands in their chronological order of execution
- all SCPI commands needed for a certain functional setup of the signal generator but not necessarily in the advisable order of their execution.

The following paragraphs provide detailed information how to generate these different kind of SCPI lists and finally how to export them as a SCPI script.

4.1 Manual SCPI List Recording

The SMW allows to manually record SCPI commands comprising any number of parameter variations.
In this recording mode, the user has to decide deliberately for which parameter modification performed the related SCPI command shall be added to the SCPI list.
The recorded list can be exported finally as a script file in various formats.

This mode of recording is extremely helpful if a certain SMW configuration has to be worked out and thus not any parameter variation or key stroke but only the final and proper settings should be recorded.
The following figure shows how the manual SCPI recording is used:

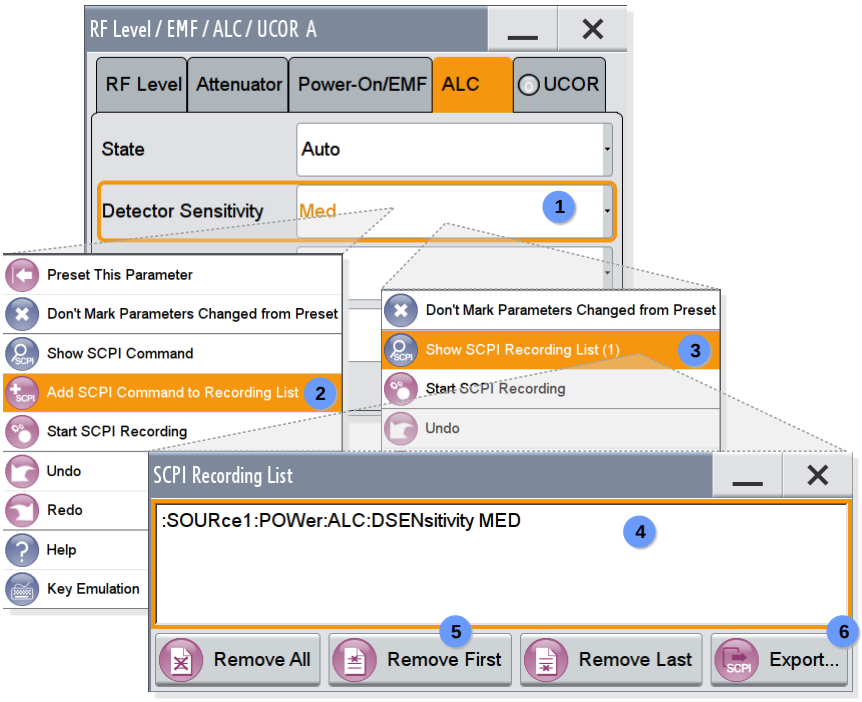


Figure 6: Add dedicated SCPI Command manually to a SCPI List

① Select a parameter and perform the necessary value adjustment.

In this example, the ‘ALC’ detector sensitivity was changed from the default value ‘Auto’ to ‘Med’. This modification is indicated by displaying the new parameter value in orange (only if the function ‘Mark all parameters changed from preset’ has been activated via the context menu).

② Add the SCPI command required to perform this parameter adjustment to the SCPI list via the context menu.

Repeat steps ① and ② for all parameter settings, which have to be included in the SCPI list.

③ Show the content of the currently recorded SCPI list at any time via the context menu.

④ The ‘SCPI Recording List’ dialog provides a chronological overview of the already recorded SCPI commands.

⑤ It further allows removing the first, the last or all command entries.

⑥ Finally, it allows to initiate the export of the SCPI list as a ‘Plain’ SCPI- or source code script by pressing the ‘Export’ key.

For details about the script export, see chapter 5 on page 19.

4.2 Automatic SCPI List Recording

To further facilitate the generation of SCPI lists, the SCPI-Recorder also supports an automatic mode of operation. In this mode the user starts the recording process in advance of a certain configuration sequence and stops it after finalization of all necessary configuration steps. The SCPI commands related to the parameter variations done in between are added to the list automatically without any further user activity. The recorded list can be exported in a final step as a script file in various formats.
This recording mode is normally the best choice for fast and convenient SCPI list recording.
The following example shows how to activate the automatic SCPI list recording and how all subsequent manual configurations are added automatically to the SCPI command list:

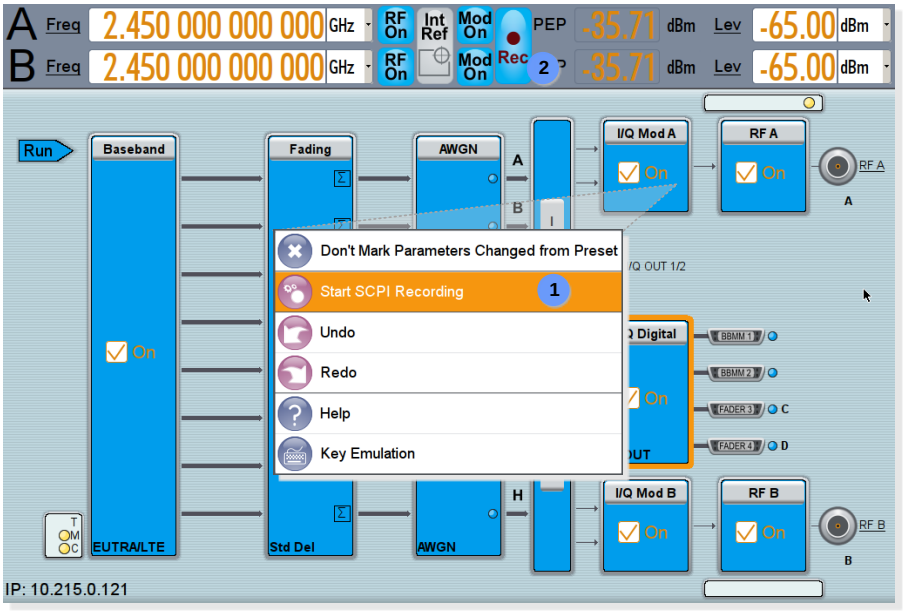


Figure 7: Start automatic SCPI Recording

① Start the automatic SCPI recording via the context menu.
It has to be mentioned that in contrast to some other context menu functions, the start functionality is not only shown if the menu is opened via the selection of a functional block or parameter field but also when opened anywhere on the GUI.
② The running recording process is indicated in the top center of the GUI block diagram.

After activation of the automatic recording process, each modification of a signal generator parameter will result in an extension of the SCPI list.

③ If, for example, the 'System Bandwidth' of the AWGN impairments applied to the 8x8 MIMO channels has to be adjusted, the 'AWGN/IMP' block has to be configured accordingly. As mentioned above this parameter variation automatically results in an extension of the SCPI list without any additional manual interaction. Additionally, this modification may be indicated by displaying the new parameter value in orange (only if the function 'Mark all parameters changed from preset' was activated via the context menu).
Repeat step ③ for all parameter settings to be included in the SCPI list.

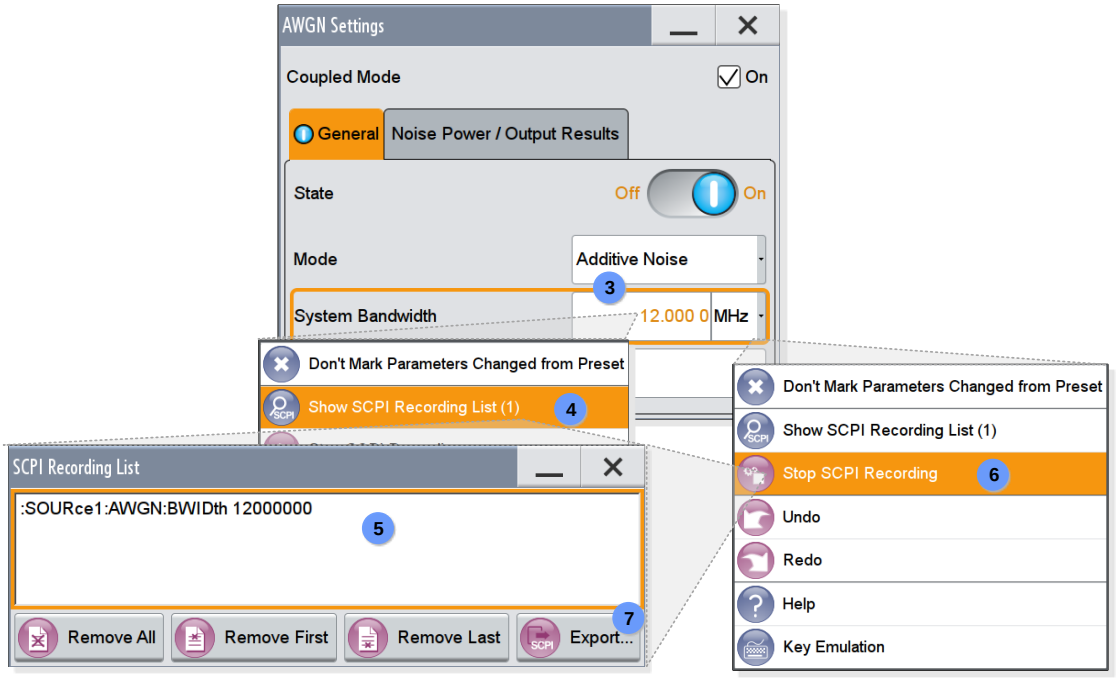


Figure 8: Add SCPI Commands automatically to SCPI List

④ Show the content of the currently recorded SCPI list at any time via the context menu. The 'SCPI Recording List' dialog provides a chronological overview of the already recorded SCPI commands.
⑤ It further allows removing the first, the last or all command entries.
⑥ Upon completion of the whole configuration sequence, stop the recording process via the context menu. Immediately after the recording has been stopped, the 'SCPI Recording List' dialog pops up and displays the final SCPI list content.
⑦ Finally, start the export of the SCPI list as a 'Plain' SCPI or source code script by pressing the 'Export' key.
For details about the script export see chapter 5 on page 19.

4.3 SCPI List Export

Some users do not need a chronological SCPI list but only a list of SCPI commands related to those configuration parameters differing from the default settings of the SMW (after an SMW preset).
These commands can be identified (highlighted) at any time, and can be compiled into a list and exported in various formats as a script file.
This functionality is especially helpful if extensive manual configurations have been made without prior activation of the automatic SCPI recording.

To get an overview about the GUI parameters modified during the manual configuration process and thus differing from their default values (after an SMW preset) they can be highlighted via a function offered by the context sensitive menu:

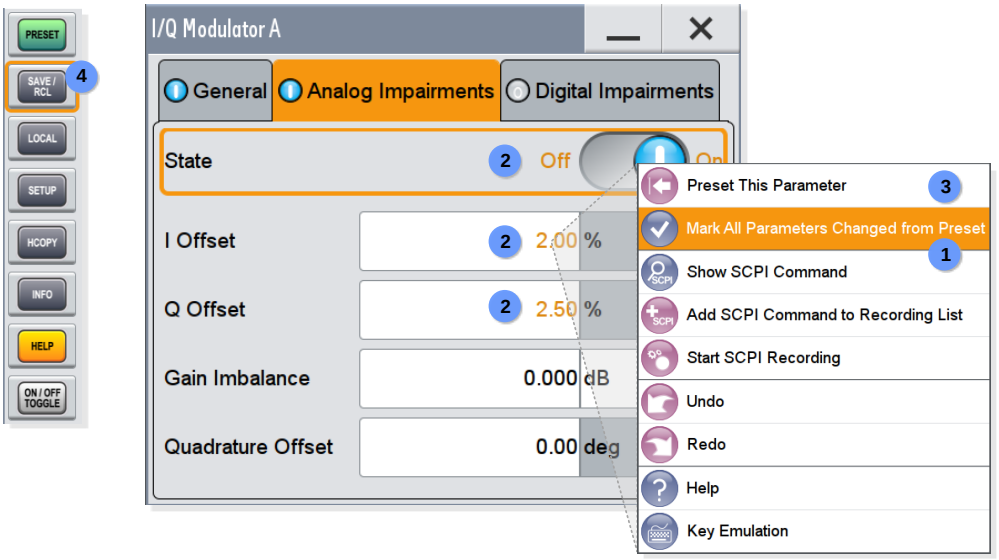


Figure 9: Mark all Parameters changed from PRESET

① Activate the functionality to show all modified parameters via the context sensitive menu.
② All modified parameters and functional blocks are highlighted in orange.
③ If a certain parameter has to be reset to its default value this parameter can be selected and reset via the context sensitive menu without any impact on any other configured parameter.

When a certain SMW configuration is considered adequate for a specific measurement task, the SCPI list that is maintained by the SMW can be exported.

④ The export and the related conversion of the SMW internal SCPI list into a SCPI script file are started via the ‘Save/Recall’ hard key and the related dialog.
Alternatively, the corresponding soft key offered through the ‘Key Emulation’ activated via the context sensitive menu may be used. This is also the method of choice in case of remote SMW operation via VNC.
For details about the script export see chapter 5 on page 19.

4.4 Manual SCPI Script Creation

Those users of the SMW who want to manually create SCPI script files without using the integrated recording and export functionality will profit from the functions that find and show SCPI commands.
For the completely manual creation of a SCPI script file it is recommended to use the ‘Context sensitive Help Function‘, described in chapter 3.1 on page 11.
In addition the ‘Context sensitive SCPI Commands’ function described in chapter 3.2 on page 12 allows an easy and straight forward determination of the index and parameter values needed to parameterize the SCPI commands.

5 SCPI Script Export and Code Generation

The generated SCPI lists can either be exported directly as a ‘Plain’ (ASCII based) SCPI script or can be converted into a source code script file by using the integrated code generator.

5.1 Export of Plain SCPI Scripts

In case only the SCPI command strings are needed for the following test automation tasks the recorded SCPI lists can easily be exported as ‘Plain’ SCPI script files.

As discussed in chapter 4 a SCPI list may consist of:

- several SCPI commands in their chronological order of execution
- all SCPI commands needed for a certain functional setup of the signal generator but not necessarily in the advisable order of their execution.

None-chronological SCPI lists are exported via the ‘Save/Recall‘ dialog whereas the ‘SCPI Recording List’ dialog has to be used to export chronological SCPI lists.

The steps required to create a ‘Plain’ SCPI script file based on SCPI lists are described in the following examples.

Export based on a none-chronological SCPI list:
When a certain SMW configuration is considered adequate for a specific measurement task the export of the automatically generated SCPI list is started via the ‘Save/Recall’ hard key and the related dialog:

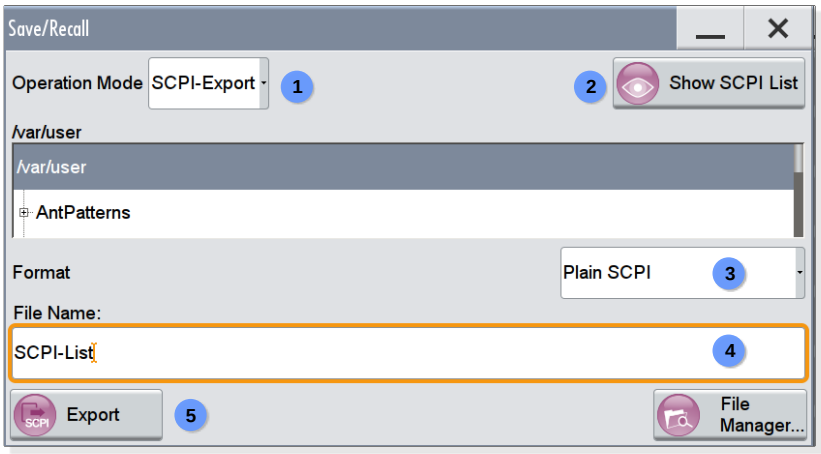


Figure 10:* Export of ‘Plain’ SCPI Scripts *(none-chronological SCPI List)

① Since the ‘Save/Recall’ dialog is not only used to control the SCPI list export but also the internal storage of SMW settings the operation mode ‘SCPI Export’ has to be selected first.
② When the ‘SCPI Export’ mode is activated the ‘Show SCPI List’ key is displayed. This key may be used to request the current content of the SMW internal SCPI list.
③ Choose the output format ‘Plain SCPI’.
④ Provide a name for the SCPI script. The script file extension is automatically set to ‘.txt’.
⑤ Press the ‘Export’ key to start the preparation of the SCPI script file.

The SCPI script comprises all SCPI commands that have to be sent to the SMW to restore the current SMW configuration status after a SMW preset.
It has to be recognized however that all SCPI commands are given in an alphabetical order in the SMW internal SCPI list and thus also in the SCPI script file. Consequently, it cannot be guaranteed that the commands are in the right order to allow a smooth and time-wise optimal reconfiguration of the SMW.
Therefore it is recommended to perform a manual check of the command list and to reorder certain commands prior to the integration of the script into a source code module, if necessary. For example, the activation of a certain functional block (e.g. a digital standard) of the SMW should not be done before all other parameters of this block are properly configured. This prevents repeated restarts of the signal generation which may lead to substantial delays in the overall configuration.

Export based on a chronological SCPI list:
The 'SCPI Recording List' dialog pops up immediately after a manual or automatic SCPI recording was finished/stopped.

① This dialog allows to remove all, the first or the last recorded SCPI command(s).

② The export of a SCPI list is started by pressing the 'Export' key:

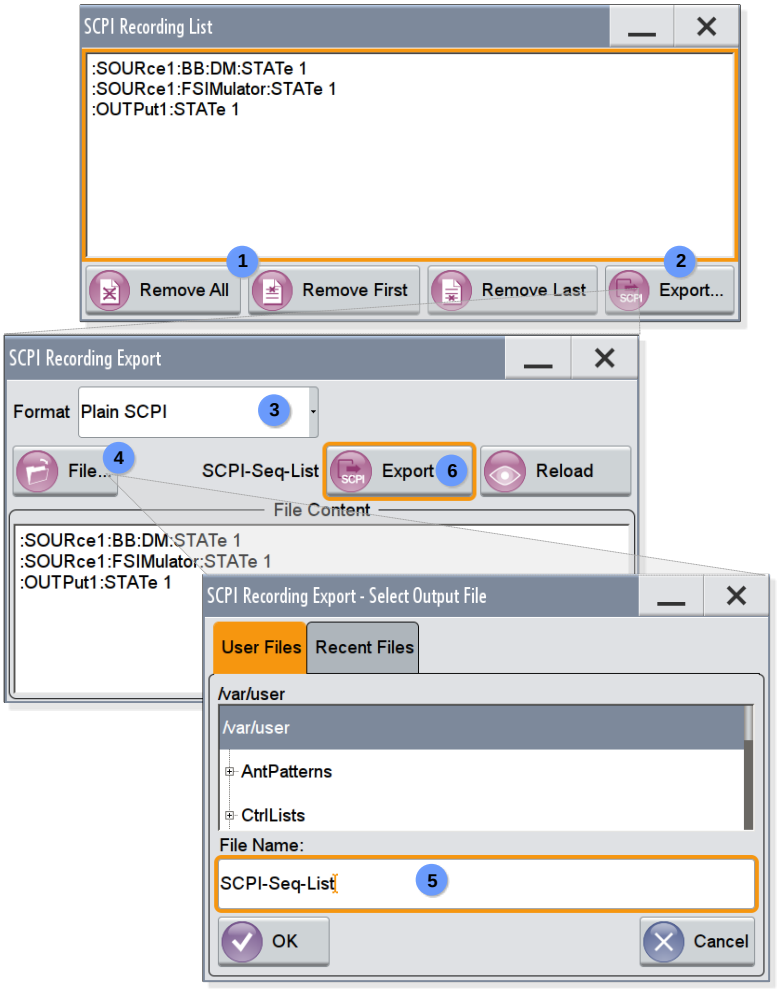


Figure 11: **Export of 'Plain' SCPI Scripts** ***(chronological SCPI List)***

③ Select the export format ‘Plain SCPI‘
④ Activate the ‘Select Output File‘ dialog by striking the ‘File‘ key.
⑤ This dialog is used to specify the name of the script file and the mass memory to be used. The script file extension is automatically set to ‘.txt’.
The choice of the mass memory for the script file (SMW internal file system or optionally attached USB mass memory storage device) that allows the most simple and fast transfer of the created file to the PC running the IDE software depends on the SMWs operating infrastructure. See chapter 6 starting on page 31 for details about the available file transfer opportunities supported by SMW.
⑥ Press the ‘Export‘ key to start the export and thus the conversion of the SMW internal SCPI list. The file content is automatically displayed afterwards.

5.2 Export of Source Code Scripts

As an alternative to the export as ‘Plain’ SCPI script files the SCPI lists can also be integrated very easily into source code modules of nearly any programming language using the integrated code generator. The code generator employs pre-defined (integrated) and user-defined code templates to control the code generation process. The steps required to create and export source code script files is shown solely based on the ‘SCPI Recording List‘ dialog.

5.2.1 Code Template Structure and Keywords

Following keywords were defined to enable the code generator to parameterize the applied template:

Code Template Keywords	
Keyword	**Function Description**
`%HOSTNAME`	This keyword has to be used to indicate the position of the SMW’s host address, which will be inserted by the code generator. It will normally be used in the `#INIT_CODE` block to open the TCP/IP based remote control session.
`%COMMAND`	Has to be used to indicate the position of the SCPI command string within the `#COMMAND_CODE` block.

Table 1: Code Template Keywords

Each code template, whether pre-defined or user-defined, has to comply with a well-established structure. It consists of the following functional blocks/sections:

Code Template Structure	
Functional Block	**Function Description**
`#EXTENSION_START` `#EXTENSION_END`	Defines the file extension to be used for the generated source code script file.
`#INIT_CODE_START` `#INIT_CODE_END`	Contains source code blocks to be located at the beginning of the generated source code script file and to be run only once during the initialization of the remote control session between SMW and IDE/Test Controller. It thus contains for example defines, includes, constants, variable definitions and initializations but also function definitions. Apart from these code parts the programming language specific code required to open a remote control session (e.g. via NI-VISA) has to be given in this functional block.
`#COMMAND_CODE_START` `#COMMAND_CODE_END`	This block contains code which has to be run to send a certain SCPI command (derived from the SCPI list) to the SMW. The SMW code generator will thus repeat it for each SCPI command contained in the provided SCPI list.
`#EXIT_CODE_START` `#EXIT_CODE_END`	The last code part written only once to the source code script file has to be written into this block of the template. It is normally used to close the remote control session. For some programming languages it is also useful/necessary to finalize a code section (e.g. main function) started in the `#INIT_CODE` block of the code template.

Table 2: Code Template Structure

Each template file has to use the file extension: `.expcodetempl` in order to enable the SMW to properly recognize a code template file.

5.2.2 Using Pre-Defined Code Templates

Pre-defined code templates for the control of the code generation process are available in the SMW for the programming languages most commonly used for test automation software.
Currently, the following pre-defined templates are integrated and thus the following IDEs and programming languages are supported:

Supported IDEs and Programming Languages (Pre-Defined Templates)		
SMW Code Template	**IDE**	**Language**
MATLAB	MATLAB	MATLAB
NI CVI	NI LabWindows/CVI	ANSI C

Table 3: Supported IDEs and Programming Languages (Pre-defined Templates)

The process of code generation is shown exemplarily using the pre-defined 'NI CVI' code template.

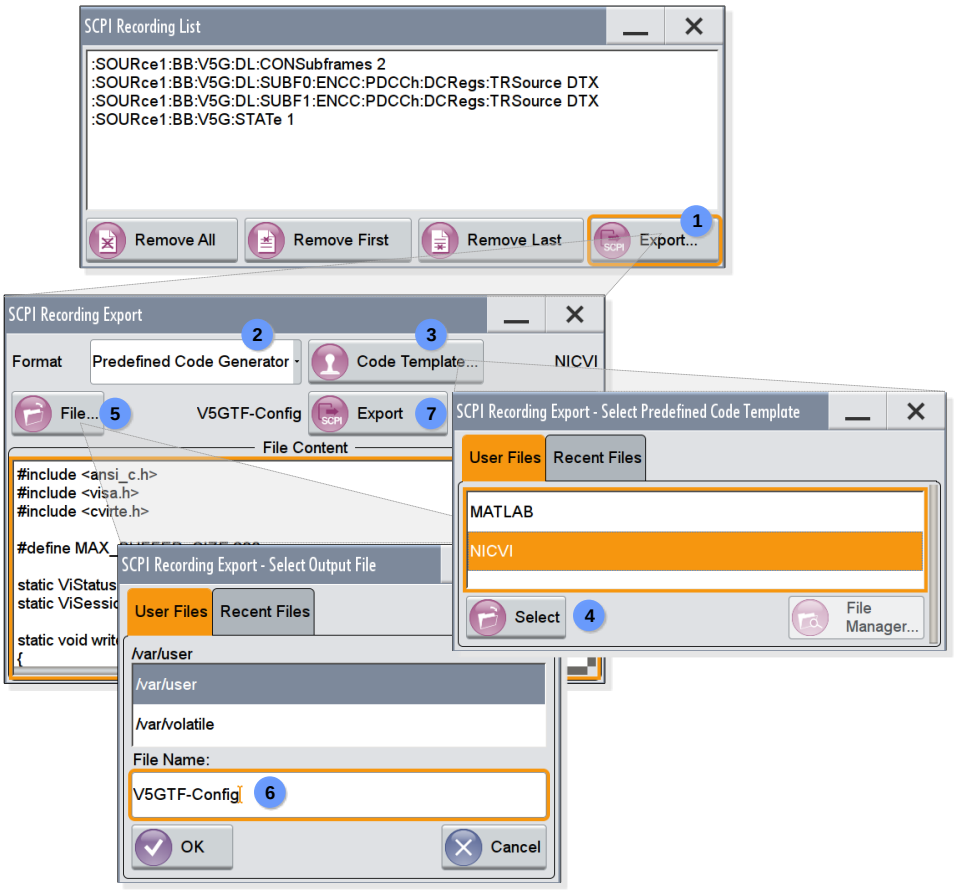


Figure 12: Source Code Generation based on NI LabWindows/CVI Template

① Press the 'Export' key in the 'SCPI Recording List' dialog automatically shown upon finalization of a recording process, to activate the 'SCPI Recording Export' dialog.
② Select the export format 'Predefined Code Generator'.
③ Press the 'Select Code Template' key to open the dialog showing the pre-defined templates.
④ Select the required template (see *Table 3*).
⑤ Open the 'Select Output File' dialog via the 'File' key.
⑥ Specify the name of the script file and the mass memory to be used for storage. The script file extension is set by the code generator in accordance with the string defined in the `#EXTENSION` block of the used code template (see *Table 2*).
The choice of the mass memory (SMW internal file system or optionally attached USB mass memory storage device) that allows the most simple and fast transfer of the created file to the PC running the IDE software depends on the SMW's operating infrastructure. See chapter 6 starting on page 31 for details about the available file transfer opportunities supported by the SMW.
⑦ Press the 'Export' key to start the export and thus the conversion of the SMW internal SCPI list. The file content is automatically displayed afterwards.

The following paragraphs summarize the minimum requirements to be fulfilled by an PC to integrate, run and/or compile the source code generated through any of the pre-defined templates.

5.2.2.1 NI LabWindows/CVI Source Code

The ANSI C source code generated by SMW can be run within a NI LabWindows/CVI IDE. The following table summarizes the hardware and software requirements.

System Requirements – NI LabWindows/CVI[1]
Hardware
NI LabWindows/CVI PC requirements (CPU, memory, hard drive...)
100Mbit or 1Gbit LAN
Software
Microsoft Windows 7
NI LabWindows/CVI 2010
NI VISA V5.4.1

1) Functionality was tested by R&S with following hardware and software components

Table 4: System Requirements to run NI LabWindows/CVI Source Code

The content of the Pre-Defined 'NI CVI' template is shown as an example for the generation of User-Defined code templates in chapter 10.1 on page 41.

5.2.2.2 MATLAB Source Code

The generated MATLAB source code can be run within a MATLAB IDE after extension with some functions of the 'R&S MATLAB Toolkit for R&S Signal Generators' [4].
The following table summarizes the hardware and software requirements in detail.

System Requirements – MATLAB[1]
Hardware
MATLAB PC requirements (CPU, memory, hard drive...)
100Mbit or 1Gbit LAN
Software
Microsoft Windows 7
MATLAB R2016a (32 or 64bit)
MATLAB Instrument Control Toolbox
R&S MATLAB Toolkit for R&S Signal Generators [4]
NI VISA V5.4.1

1) Functionality was tested by R&S with following hardware and software components

Table 5: System Requirements to run MATLAB Source Code

The ‘R&S MATLAB Toolkit for R&S Signal Generators‘ [4] has to be installed to extend the MATLAB functionality. It provides generic functions (m-files) for the remote control of R&S signal generators. Thus, it is useful not only for running SMW MATLAB script files but also for many other interactions between MATLAB and R&S signal generators (e.g. generation of waveform files). The following toolkit functions are called by the generated MATLAB source code:

Used MATLAB Toolkit Functions	
Function	**Description**
rs_connect	Sets up the connection to the signal generator and performs an initial link check. It thus operates as an additional software layer above the MATLAB Instrument Control Toolbox which simplifies the remote control of R&S signal generators. The function returns an object handle to the connected signal generator which has to be used for any further interaction with the instrument. Even though this function supports several remote control connection types and remote control interface vendors the SMW code generator only uses the VISA based TCP/IP connection type from National Instruments. If any other connection type is needed the function parameters have to be adapted accordingly (see [4] for details).
rs_send_query	Sends a single SCPI query to the signal generator specified by the provided object handle. The query returns the OPC status.

Table 6: Used R&S MATLAB Toolkit Functions

The content of the Pre-Defined ‘MATLAB’ template is shown in chapter 10.2 on page 42 as an example for the generation of User-Defined code templates.

5.2.3 Using User-Defined Code Templates

The code generator can be used even in case a source code in a programming language not directly supported by the pre-defined code templates is needed. In this case, a user-defined code template incorporating the coding rules and the remote control functions of the respective programming language can be set up and loaded into the SMW.
The code generation process based on a user-defined template (VSC++VXI-OPC, see chapter 5.2.3.1 for details) is shown in the following example:

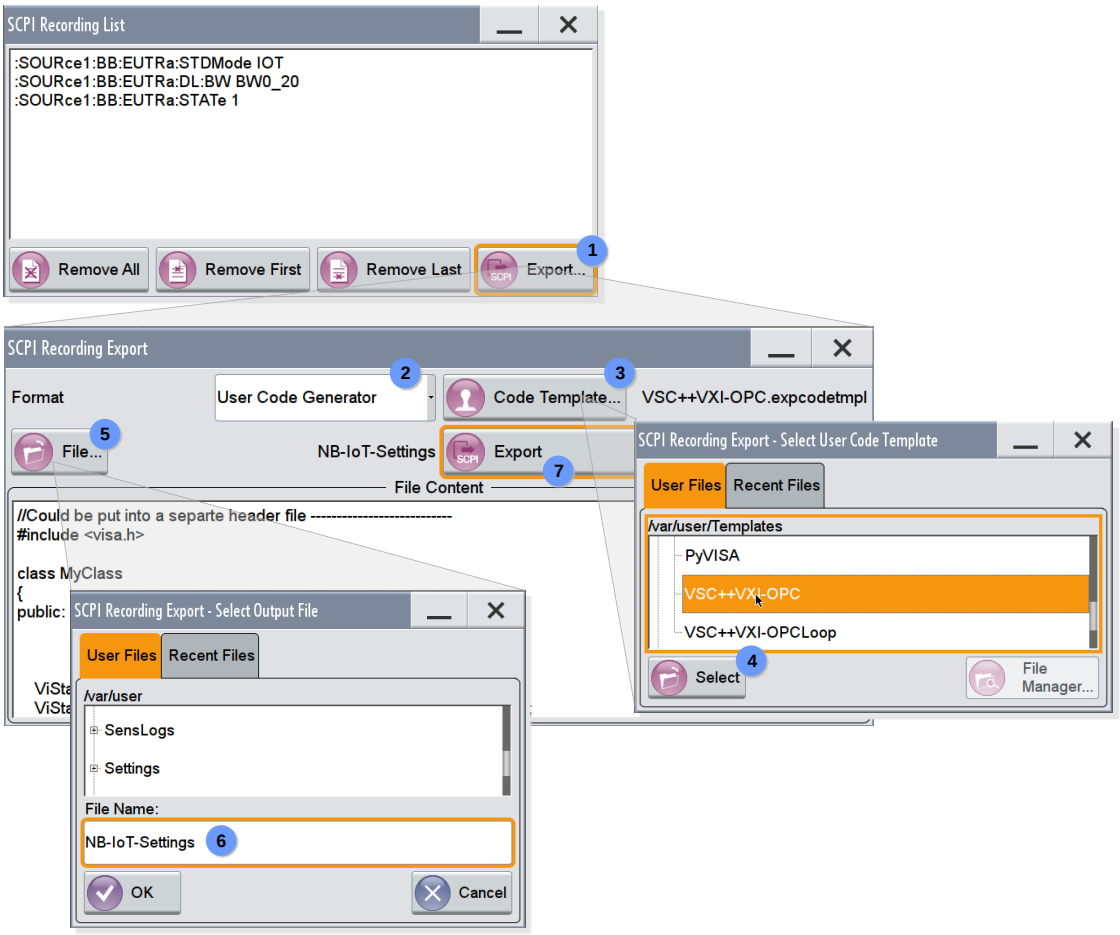


Figure 13: Source Code Generation based on User-Defined Template

① Press the ‘Export‘ key in the ‘SCPI Recording List‘ dialog automatically displayed upon finalization of a recording process, to activate the ‘SCPI Recording Export‘ dialog.
② Select the export format ‘User Code Generator‘.
③ Open the file dialog to select the user-defined template via the ‘Code Template‘ key.
④ Select the desired user-defined template.
⑤ Open the ‘Select Output File’ dialog via the ‘File’ key.
⑥ Specify the name of the script file and the mass memory to be used for the storage.
The script file extension is set by the code generator in accordance with the string defined in the `#EXTENSION` block of the used code template (see *Table 2*).
The choice of the mass memory (SMW internal file system or optionally attached USB mass memory storage device) that allows the most simple and fast transfer of the

created file to the PC running the IDE software depends on the SMW's operating infrastructure. See chapter 6 starting on page 31 for details about the available file transfer opportunities supported by the SMW.

⑦ Press the 'Export' key to start the export and thus the conversion of the SMW internal SCPI list. The file content is automatically displayed afterwards.

As the use of this code generation functionality requires an appropriate user-defined code template to be available, the following paragraph shows the principle how to create a code template for any IDE and/or programming language.

5.2.3.1 Microsoft Visual C++ User-Defined Code Template

The following user-defined template allows the code generation for an application whose architecture is based on the usage of VXIplug&play drivers. In parallel, the advantages of the SCPI recording can be fully utilized. This impressively demonstrates how flexible the template based code generation can be used in the context of test automation software development.

In detail, a Microsoft Visual C++ class is created whose methods make use of VXIplug&play R&S RF signal generator driver functionality for the remote access to the SMW. Any SCPI command of the recorded SCPI list is transmitted transparently to the SMW by a driver function.

Supported IDE and Programming Language (User-Defined Template)		
SMW Code Template	**IDE**	**Language**
VSC++VXI-OPC	Microsoft Visual Studio 2013	Microsoft Visual C++

Table 7: Supported IDE and Programming Language (User-defined Template)

The code template is based on the structure introduced in chapter 5.2.1 on page 21 and has the following content:

```
#EXTENSION_START
   .cpp
#EXTENSION_END
#INIT_CODE_START
   //Could be put into a separate header file -------------------------
   #include <visa.h>                                                  (1)

   class MyClass    (2)
   {
   public:
                MyClass();
                ~MyClass();

       ViStatus configSMW();
       ViStatus writeCommand(ViSession instrHandle, ViString command);
   };
   //---------------------------------------------------------------------
```

```
#include "Windows.h"
#include "rsrfsiggen.h"
```
3

```
MyClass::MyClass()
{
    //Add application specific constructor code if applicable
}
```
4

```
MyClass::~MyClass()
{
    //Add application specific destructor code if applicable
}
```

```
ViStatus MyClass::writeCommand(ViSession instrHandle,
                               ViString command)
{
    //Write SCPI command to instrument
    ViStatus status = rsrfsiggen_WriteInstrData(instrHandle,
                                                command);

    //Query OPC
    ViInt32 OPCState;
    status |= rsrfsiggen_QueryOPC(instrHandle, &OPCState);

    return(status);
}
```
5

```
ViStatus MyClass::configSMW()
{
    ViStatus  status;
    ViBoolean performReset = VI_FALSE;
    ViBoolean performIDQuery = VI_FALSE;
    ViSession instrHandle;

    //Initialize the VXIplug&Play driver and establish connection
    status = rsrfsiggen_init((ViRsrc)"TCPIP::%HOSTNAME::INSTR",
                             performIDQuery, performReset,
                             &instrHandle);
    if(status < 0)
    {
        return(status);
    }

    while(1)
    {
#INIT_CODE_END
#COMMAND_CODE_START
        //Write SCPI command to instrument
        status = writeCommand(instrHandle, "%COMMAND");
        if(status < 0)
            break;
#COMMAND_CODE_END
#EXIT_CODE_START
        break;
    }
    //Close connection
    rsrfsiggen_close(instrHandle);
    return(status);
}
#EXIT_CODE_END
```
6
7
8

① The header file of the VISA library is included and the class is declared in the `#INIT_CODE` block.

② The class is named `MyClass` in the template and later on has to be renamed in the generated source code as desired. The generated code up to the end of the class declaration can be swapped to a related header file.

③ The header file of the VXIplug&play driver library is included at the beginning of the class definition.

④ The class constructor and destructor are part of the code template. Both of them can be adapted application-specific after code generation.

⑤ The method `writeCommand` is an auxiliary function used to transmit any SCPI command to the SMW followed by an OPeration Complete (OPC) query. This approach guarantees that the next command can only be sent after the SMW has finished the internal command processing. See [5] for further information regarding an optimized design of remote control applications.

⑥ The `configSMW` method encapsulates the whole SMW configuration process. It reaches from the `#INIT_CODE` block, whose code is run only once, over the `#COMMAND_CODE` block, which is repeated for each SCPI command, down to the `#EXIT_CODE` block, where the method ends.
First the VXIplug&play driver is initialized within the `#INIT_CODE` block and the LAN (TCP/IP) based connection to the SMW is established. The keyword `%HOSTNAME` used in the template is replaced with the hostname configured on the SMW by the code generator.

⑦ The individual SCPI commands of a recorded list are sent to the SMW using the auxiliary function `writeCommand` within the `#COMMAND_CODE` block. For this purpose, the code generator repeats this code segment for each SCPI command of the list and the keyword `%COMMAND` is replaced by the SCPI command string.

⑧ The while-loop, which is laid around the #COMMAND_CODE block to allow a simple interruption of the SMW configuration in case of erroneous writeCommand calls, is closed in the `#EXIT_CODE` block. Finally, the remote connection to the SMW is closed.

5.2.3.2 Microsoft Visual C++ Source Code

The generated Visual C++ source code can be run/compiled within a Microsoft Visual Studio IDE. The following table summarizes the hardware and software requirements.

System Requirements – Visual C++ (with VXIplug&play driver)[1]
Hardware
Microsoft Visual Studio 2013 PC requirements (CPU, memory, hard drive...)
100Mbit or 1Gbit LAN
Software
Microsoft Windows 7
Microsoft Visual Studio 2013
Microsoft Visual C++ 2013
R&S rsrfsiggen VXIplug&play driver (32-bit)
NI VISA V5.4.1

1) Functionality was tested by R&S with following hardware and software components

Table 8: System Requirements to run Visual C++ Source Code with VXI Driver

Prior to the integration of the generated script file into a Visual Studio project it is recommended to split up the file according to common C++ coding practice into a header- and a source-file (see chapter 5.2.3.1, ②). Furthermore, the created class which is named `MyClass` by the template should be renamed as desired.
To bring the class to life only an instance has to be created followed by a call of the `configSMW` method.

Apart from the source code related activities, it has to be ensured that the included paths and library directories of the VXIplug&play driver and of the VISA library are added properly to the project properties.

6 Script File Transfer

There are several different methods available to easily transfer the generated script files from the SMW to the PC running the IDE software for further processing and integration in a test automation project. Which way to select depends on the SMW's field of application and the available infrastructure:

- Access of the SMW file system via ftp
- Mapping of the SMW file system on the PC running the IDE software
- Mounting of a shared directory on the SMW
- Storage on an USB mass memory device (Memory stick)

The following paragraphs provide an overview on how to use these methods.

6.1 File Transfer via File Transfer Protocol (ftp)

A PC can directly access the SMW's file system via ftp if the SMW is integrated into a LAN.
In case the ftp access functionality is disabled (it is enabled when supplied to the customer) it may be enabled as follows:

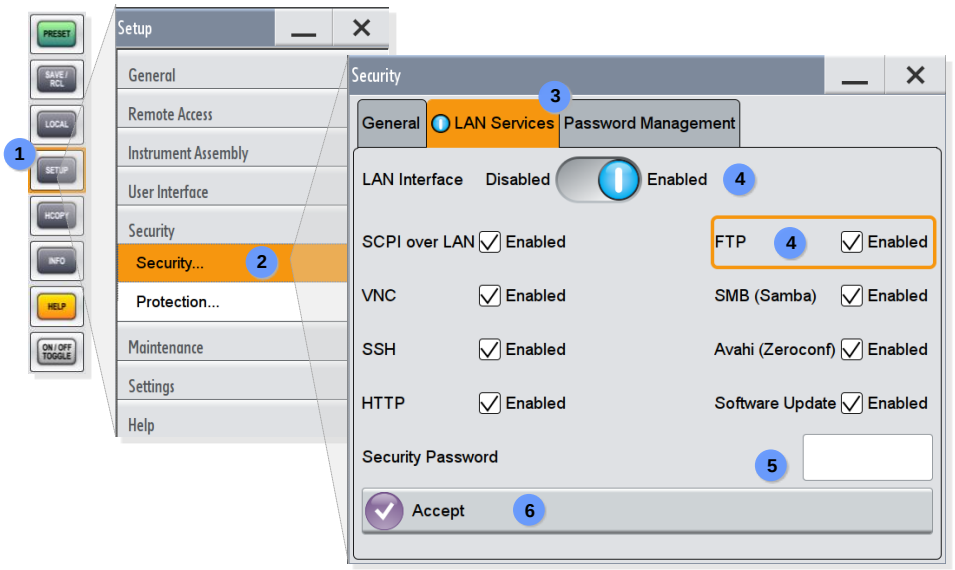


Figure 14: Enabling of ftp-Access to SMW File System

① Open the 'Setup' menu either via the 'Setup' hard key (or alternatively via the corresponding soft key offered by the 'Key Emulation' which can be activated via the context sensitive menu).
② Select the menu item 'Security' to open the related 'Security' dialog.
③ Activate the 'LAN Services' tab.
④ Within this dialog the 'LAN Interface' itself and the 'FTP' protocol have to be enabled.
⑤ Finally, the 'Security Password' (default: 123456) has to be provided.
⑥ Confirm the changes by pressing the 'Accept' key.

The ftp access to the SMW file system can be established via an internet browser or a file explorer available on the PC.
The following example shows the file explorer access procedure to be followed in case of a Windows 7 based PC:

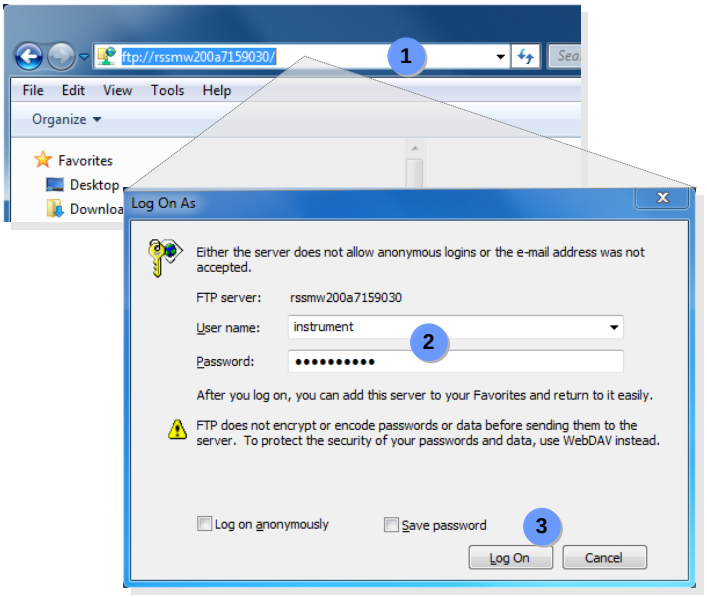


Figure 15: ftp-Access to SMW File System via Windows 7 based PC

① First the desired ftp server has to be specified. In case of the SMW the following syntax and data have to be used:

ftp://<Host Name> or ftp://<SMW IP Address>

The parameter <Hostname> has to be replaced with the host name configured on the particular SMW. By default the SMW host name is set as follows:

SMW200A-<Serial Number>

Running an older SMW FW (< 3.20.200.19), the default host name is set as follows:

rssmw200a<Serial Number>

The parameter <Serial Number> has to be replaced with the serial number of the SMW to be accessed (see label on the rear of the SMW).
② In advance of granting ftp access to its file system the SMW requests the 'User name' and the related 'Password'. Both parameters are set to 'instrument' by default. The assigned password can be changed via the SMW 'Security' menu (see *Figure 14*, step ②, Password Management).
③ The ftp connection will be established by pressing the 'Log On' key.

In case of the host name or the IP address are unknown (e.g. host name not set to the default value) both of them can be found via the menu item ‘Remote Access, Network’ provided in the ‘Setup’ menu (see *Figure 16*).

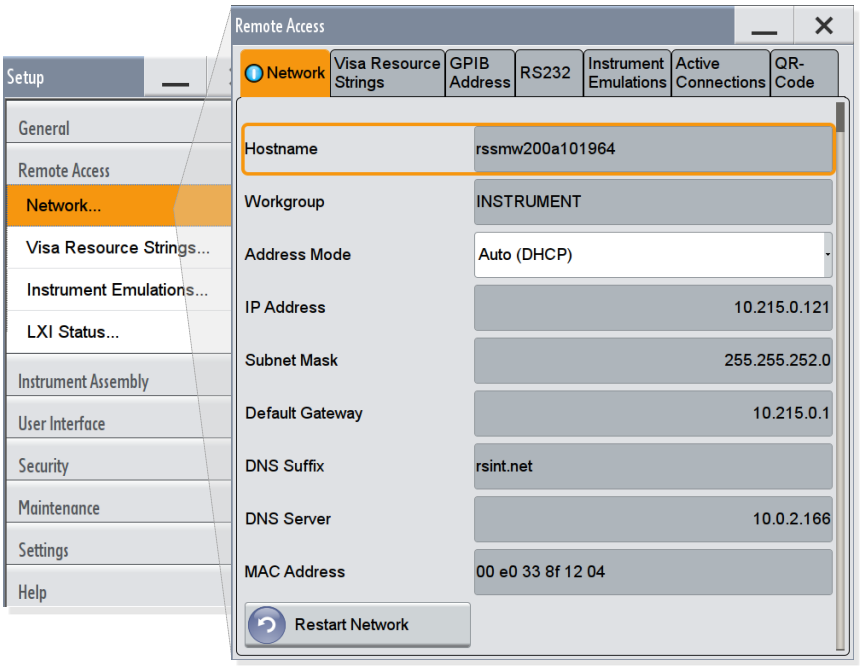


Figure 16: Remote Access – Network Setup Menu

6.2 File Transfer via File System Mapping

As an alternative to the ftp access, the file system of the SMW can be mapped as a network drive to the PC running the IDE software. In case the file system mapping functionality is disabled (it is enabled when supplied to the customer) it may be enabled as follows:

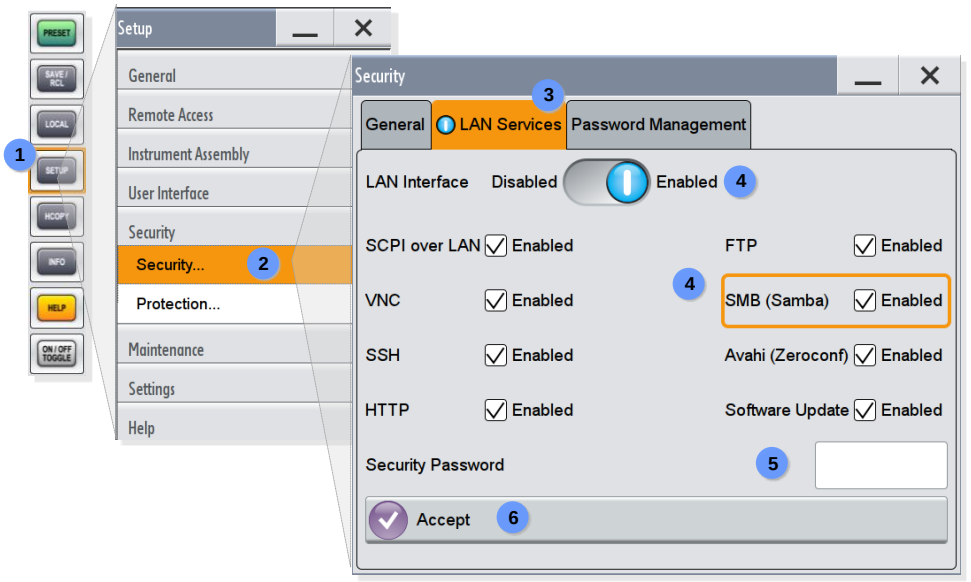


Figure 17: Enabling of SMW File System mapping

① Open the ‘Setup’ menu either via the ‘Setup‘ hard key (or alternatively via the corresponding soft key offered by the ‘Key Emulation’ which can be activated via the context sensitive menu).
② Select the menu item ‘Security‘ to open the related ‘Security‘ dialog.
③ Activate the ‘LAN Services’ tab.
④ Within this dialog the ‘LAN Interface‘ itself and the ‘Server Message Block (SMB)‘ protocol have to be enabled.
⑤ Finally, the ‘Security Password‘ (default: 123456) has to be provided.
⑥ Confirm the changes by pressing the ‘Accept‘ key.

After that, the SMW file system can be mapped as a network drive to a PC.

The following example shows the file explorer mapping procedure to be followed in case of a Windows 7 based PC:

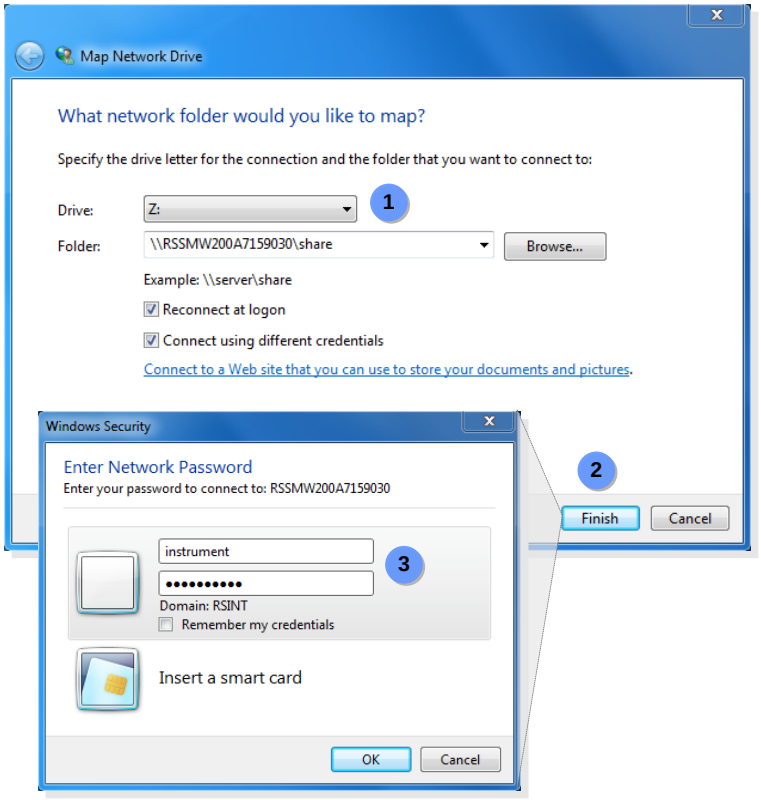


Figure 18: Mapping of SMW File System on a Windows 7 based PC

① First the desired network drive letter has to be specified in the mapping dialog accessible via the file explorer menu bar. Thereafter the folder to be mapped has to be defined. In case of the SMW the following syntax and data have to be used:

\\<Host Name>\share or \\<SMW IP Address>\share

The parameter <Hostname> has to be replaced with the host name configured on the particular SMW. By default the SMW host name is set as follows:

SMW200A-<Serial Number>

Running an older SMW FW (< 3.20.200.19), the default host name is set as follows:

rssmw200a<Serial Number>

The parameter <Serial Number> has to be replaced with the serial number of the SMW to be accessed (see label on the rear of the SMW).

② The mapping process is started by pressing the 'Finish' key.

③ Before enabling the user to map the file system the SMW requests the 'User name' and the related 'Password'. Both parameters are set to 'instrument' by default. The assigned password can be changed via the SMW 'Security' menu (see *Figure 17*, step ②, Password Management).

In case of the host name or the IP address are not known (e.g. host name not set to the default value) both of them can be found via the menu item 'Remote Access, Network' provided in the 'Setup' menu (see *Figure 16*).

6.3 File Transfer via Shared Folder

Besides the file transfer methods described above, which are based on the granted access to the SMW's file system, the SMW is also able to mount 'Shared Folders' of a connected PC (file server). This feature enables the SMW to save generated script files directly on the PC running the IDE system.

To share folders/directories of a PC file system with other network resources (e.g. the SMW) operating system dependent settings are required. The following example shows, which settings have to be made on a Windows 7, based PC to share a certain folder with a specific user group:

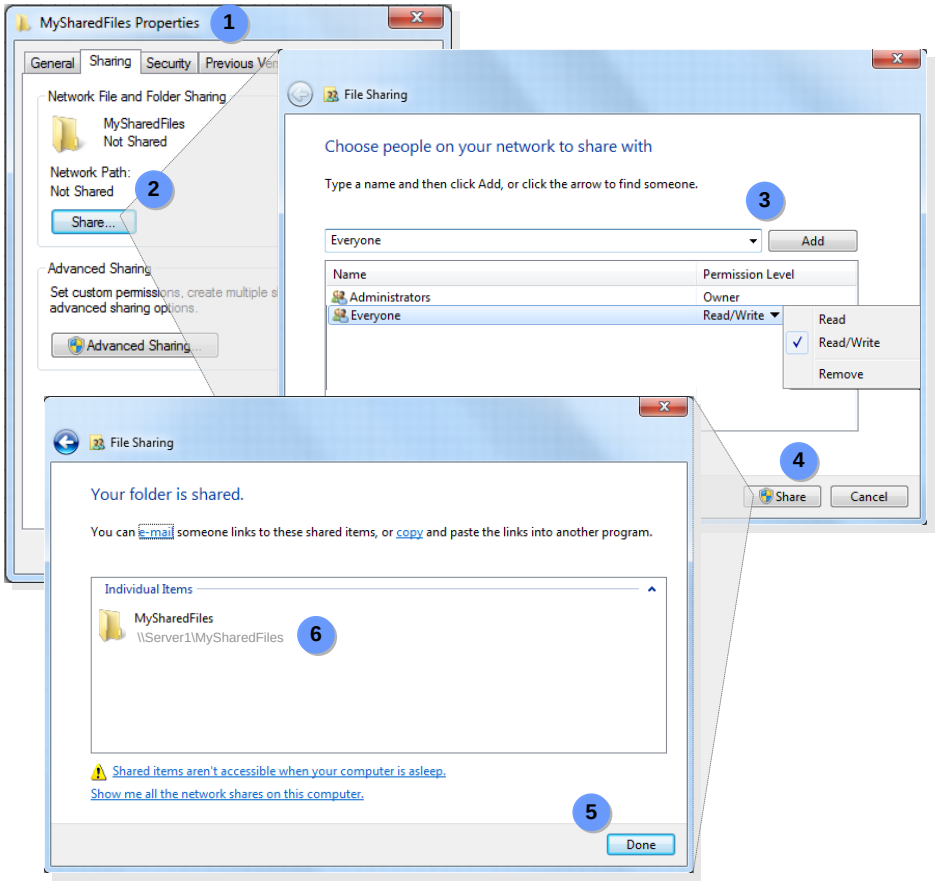


Figure 19: Sharing of a Folder/Directory on a Windows 7 based PC

① By using the file explorer first open the properties dialog of a certain folder (e.g. MySharedFiles) which has to be shared.
② Select the 'Sharing' tab and press the 'Share' button to open the File Sharing dialog.
③ Add a user group (e.g. Everyone) and provided the 'Permission Level' Read/Write.
④ Press 'Share' to activate the folder share.
⑤ Finalize the process by pressing 'Done'.
⑥ Now the shared folder can be accessed via the network path of the shared folder (e.g. \\Server1\MySharedFiles) by any device which has LAN access to the PC (e.g. a SMW).

To mount a shared folder on a SMW following steps are required:

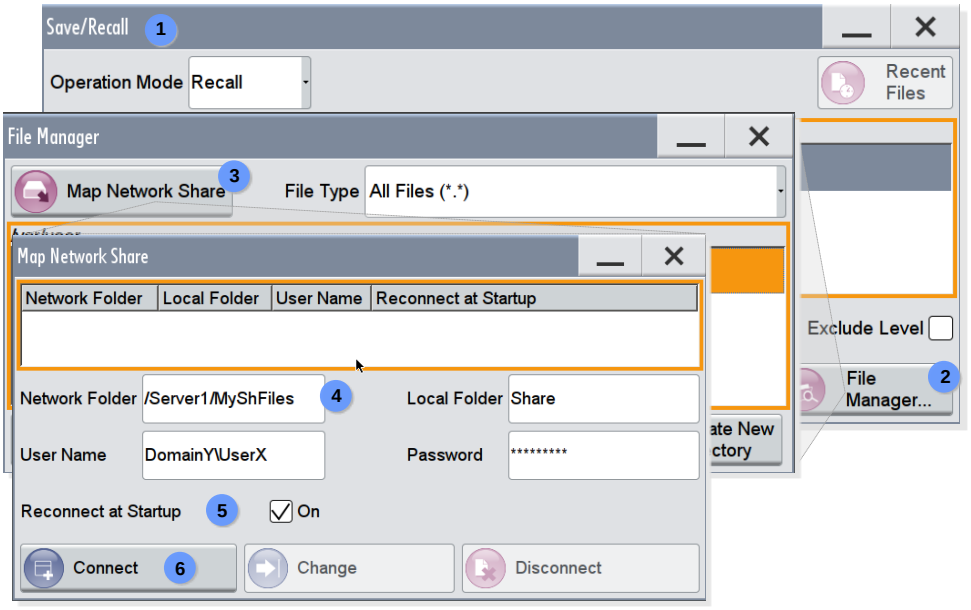


Figure 20: Mount a Shared Folder on a SMW

① Open the 'Save/Recall' dialog via the 'SAVE/RCL' hard key (or alternatively via the corresponding soft key offered by the 'Key Emulation' which can be activated via the context sensitive menu).
② Open the 'File Manager'.
③ Open the 'Map Network Share' dialog
④ and provide following parameters:

Network Folder: The network path of the shared folder (e.g. ***//***Server1***/***MyShFiles). Take care that the backslash has to be replaced by a slash!
Local Folder: The name used by the SMW as a synonym for the shared network folder.
User Name: The used user name (incl. the domain) on the PC offering the shared folder.
Password: The related user password.

⑤ Specify if the shared folder shall be reconnected at any SMW restart.
⑥ Mount the shared folder by pressing the 'Connect' key.

After that the SMW has full read/write access to any file located within the shared folder.

6.4 File Transfer via USB Mass Memory

Some SMW operation areas do not provide any means to access the SMW's file system directly via a LAN. The reason could be a stand-alone operation of the SMW or company specific security guidelines, which do not allow this kind of access.
In these cases, the transfer of script files has to be managed via a USB mass memory storage device (e.g. memory stick). The memory stick only has to be connected to one of the USB ports offered by the SMW to be then available for all mass memory operations.
In case the USB mass memory storage functionality is disabled, (it is enabled when supplied to the customer) it may be enabled as follows:

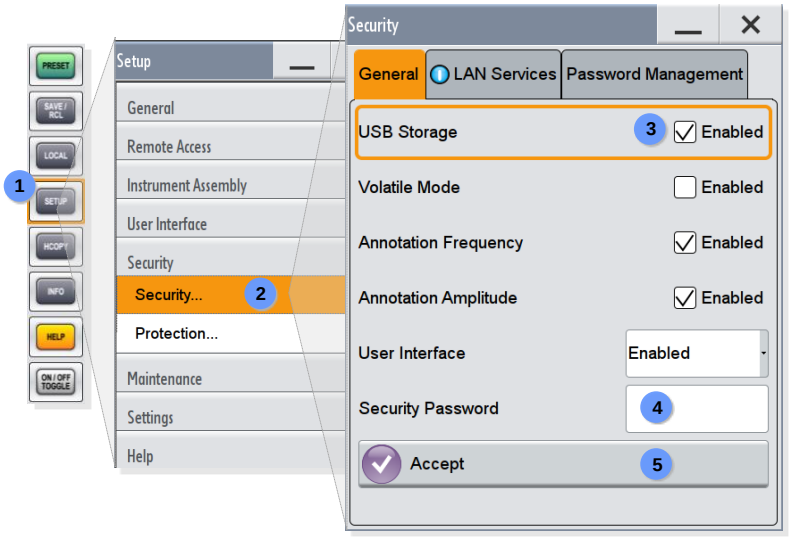


Figure 21: Enabling of USB Mass Memory Storage

① Open the 'Setup' menu either via the ‚Setup' hard key (or alternatively via the corresponding soft key offered by the 'Key Emulation' which can be activated via the context sensitive menu).
② Select the menu item 'Security' to open the related 'Security' dialog.
③ Activate the 'General' tab and enable 'USB Storage'.
④ Provide the 'Security Password' (default: 123456).
⑤ Confirm the modification by pressing the 'Accept' key.

7 SCPI Script Import

SCPI scripts can not only be exported from the SMW but can also be imported to allow a quick and easy re-configuration of the SMW.

7.1 SCPI Script Import via USER Key

Each ‘Plain’ SCPI list saved within a SCPI script file can be assigned to the ‘USER‘ Key. Thus, any SMW configuration can be repeated by a simple key stroke.

The following example shows how a ‘Plain’ SCPI script file is imported from an attached USB stick and how the SCPI list is assigned to the ‘USER‘ key:

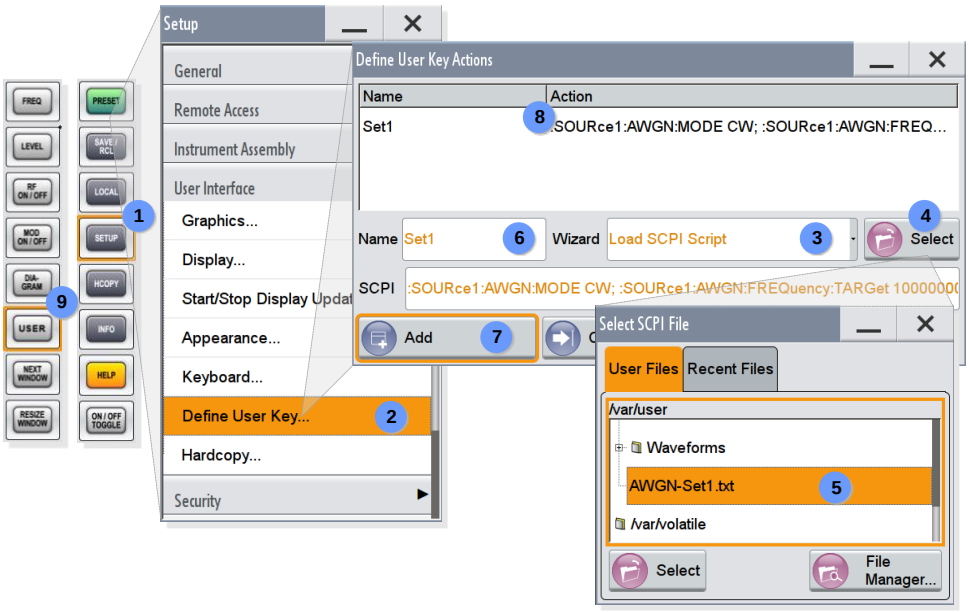


Figure 22: SCPI Script assignment to USER Key

① Open the ‘Setup’ menu either via the ‘Setup‘ hard key or alternatively via the corresponding soft key offered by the ‘Key Emulation’ which can be activated via the context sensitive menu.
② Select ‘Define User Key‘ to open the dialog for the definition of user key actions.
③ Choose the wizard ‘Load SCPI script’
④ Open a file dialog to select the desired SCPI script file by pressing the ‘Select‘ key.
⑤ The selection of a certain script file is confirmed by pressing the ‘Select‘ key. Then the complete SCPI list contained in the script file is assigned to the ‘SCPI’ parameter entry of the ‘Define User Key Actions‘ dialog.
⑥ Optionally a certain ‘Action name’ may be assigned to the SCPI sequence. This is helpful to differentiate between the assigned key actions especially in case several sequences are assigned to the ‘USER’ key.
⑦ Press the ‘Add’ key to finally activate the SCPI sequence.
⑧ After this activation step the sequence appears in the ‘Action’ window which indicates the successful assignment to the ‘USER’ key.
In case several script files have to be assigned to the ‘USER’ key to allow huge

configurations to be done with one key stroke, steps ③ to ⑦ have to be repeated for each script file.
⑨ The SCPI sequence(s) are now ready to be replayed by pressing the ‘USER‘ key.

8 Abbreviations

IDE	Integrated Development Environment
IVI	Interchangeable Virtual Instrument
OPC	Operation Complete
SCPI	Standard Commands for Programmable Instrumentation
VME	Versa Module Eurocard
VISA	Virtual Instrument Software Architecture
VNC	Virtual Network Computing
VXI	VME Extensions for Instrumentation

9 References

Manuals:

[1] R&S, SMW200A Operating Manual

Application Notes:

[2] R&S, 1GP72, Connectivity of Rohde & Schwarz Signal Generators
[3] R&S, 1MA153, Development Hints and Best Practices for Using Instrument Drivers
[4] R&S, 1GP60, MATLAB Toolkit for R&S Signal Generators
[5] R&S, 1GP79, Top Ten SCPI Programming Tips for Signal Generators

Books:

[6] Pieper John M.: Automatic Measurement Control-A tutorial on SCPI and IEEE488.2, Rohde & Schwarz, 2007

10 Appendix

10.1 NI CVI Template

```
#EXTENSION_START
   .c
#EXTENSION_END
#INIT_CODE_START
   #include <ansi_c.h>
   #include <visa.h>
   #include <cvirte.h>

   #define MAX_BUFFER_SIZE 200

   static ViStatus status;
   static ViSession defaultRM, handle;

   static void write_command(char *command)
   {
       char   writeBuffer[MAX_BUFFER_SIZE];
       char   readBuffer[MAX_BUFFER_SIZE];
       int    length;
       int    readCount;

       strcpy(writeBuffer, command);
       //Append "*OPC?" to sync
       strcat(writeBuffer, ";*OPC?");
       length = strlen(writeBuffer);
       writeBuffer[length]='\n';
       length = length+1;
       viWrite(handle, writeBuffer, length, VI_NULL);
       //Read result
       viRead(handle, readBuffer, 100, &readCount);
   }

   int main(int argc, char *argv[])
   {
       if(InitCVIRTE(0, argv, 0) == 0)
          return -1; //Out of memory
       //Create a VISA session and return a handle to it
       viOpenDefaultRM(&defaultRM);
       //Create a VISA session via LAN (TCPIP) to instrument and return
       //a handle to it
       viOpen(defaultRM, (ViRsrc)"TCPIP::%HOSTNAME::INSTR", VI_NULL,
              VI_NULL, &handle);
#INIT_CODE_END
#COMMAND_CODE_START
       write_command("%COMMAND");
#COMMAND_CODE_END
#EXIT_CODE_START
       viClose (handle);
       viClose (defaultRM);
       return 0;
   }
#EXIT_CODE_END
```

10.2 MATLAB Template

```
#EXTENSION_START
   .m
#EXTENSION_END
#INIT_CODE_START
   [status, instrObject] = rs_connect('visa', 'ni',
                                      'TCPIP::%HOSTNAME::INSTR');
   if(~status)
       disp(['Instrument connection failed: ' num2str(status)]);
       return;
   end

   while(1)
#INIT_CODE_END
#COMMAND_CODE_START
       [status, result] = rs_send_query(InstrObject,
                                        '%COMMAND; *OPC?');
       if(~status), break, end;
#COMMAND_CODE_END
#EXIT_CODE_START
       break;
   end

   delete(instrObject);
   clear;
   return;
#EXIT_CODE_END
```

11 Ordering Information

Please visit the Rohde & Schwarz product websites at www.rohde-schwarz.com for ordering information on the following Rohde & Schwarz products:

- R&S®SMW200A vector signal generator
- R&S®SMA100B vector signal generator

Rohde & Schwarz

The Rohde & Schwarz electronics group offers innovative solutions in the following business fields: test and measurement, broadcast and media, secure communications, cybersecurity, radiomonitoring and radiolocation. Founded more than 80 years ago, this independent company has an extensive sales and service network and is present in more than 70 countries.

The electronics group is among the world market leaders in its established business fields. The company is headquartered in Munich, Germany. It also has regional headquarters in Singapore and Columbia, Maryland, USA, to manage its operations in these regions.

Regional contact

Europe, Africa, Middle East
+49 89 4129 12345
customersupport@rohde-schwarz.com

North America
1 888 TEST RSA (1 888 837 87 72)
customer.support@rsa.rohde-schwarz.com

Latin America
+1 410 910 79 88
customersupport.la@rohde-schwarz.com

Asia Pacific
+65 65 13 04 88
customersupport.asia@rohde-schwarz.com

China
+86 800 810 82 28 |+86 400 650 58 96
customersupport.china@rohde-schwarz.com

Sustainable product design

- Environmental compatibility and eco-footprint
- Energy efficiency and low emissions
- Longevity and optimized total cost of ownership

Certified Quality Management ISO 9001

Certified Environmental Management ISO 14001

Rohde & Schwarz GmbH & Co. KG
Mühldorfstraße 15 | D - 81671 München
Phone + 49 89 4129 - 0 | Fax + 49 89 4129 – 13777

www.rohde-schwarz.com