R&S®RTO, R&S®RTP
R&S®ScopeSuite Automation
Manual

Manual

Version 02

ROHDE&SCHWARZ

This document describes the following R&S options:

- R&S®RTO K99 (1326.4419.02)
- R&S®RTP K99 (1326.4425.02)

The software contained in this product uses several valuable open source software packages. For information, see the "Open Source Acknowledgment" document, which is available for download from the R&S RTO/RTP product page at http://www.rohde-schwarz.com/product/rto.html > "Software".
Rohde & Schwarz would like to thank the open source community for their valuable contribution to embedded computing.

Mühldorfstr. 15, 81671 München, Germany
Phone: +49 89 41 29 - 0
Fax: +49 89 41 29 12 164
Email: info@rohde-schwarz.com
Internet: www.rohde-schwarz.com

1178.8959.02 | Version 02 | R&S®RTO, R&S®RTP

The following abbreviations are used throughout this manual: R&S®RTO is abbreviated as R&S RTO, R&S®RTP is abbreviated as R&S RTP, and R&S®ScopeSuite is abbreviated as R&S ScopeSuite.

Contents

1 Programming Interface Overview

R&S ScopeSuite provides .NET remote programming interface by using Windows Communication Foundation (WCF) technology. It allows the client program to control the R&S ScopeSuite and perform compliance tests while R&S ScopeSuite is running on a local or a remote machine (oscilloscope or a PC).

`ScopeSuiteService` is hosted by the R&S ScopeSuite. By following WCF unified programming model and duplex communication design patterns, the client creates a proxy of the service and connects to the service via WCF framework. It is then able to control the R&S ScopeSuite by consuming the service.

`ScopeSuiteService` works with .NET 4.5 in Windows 7/10 platform.

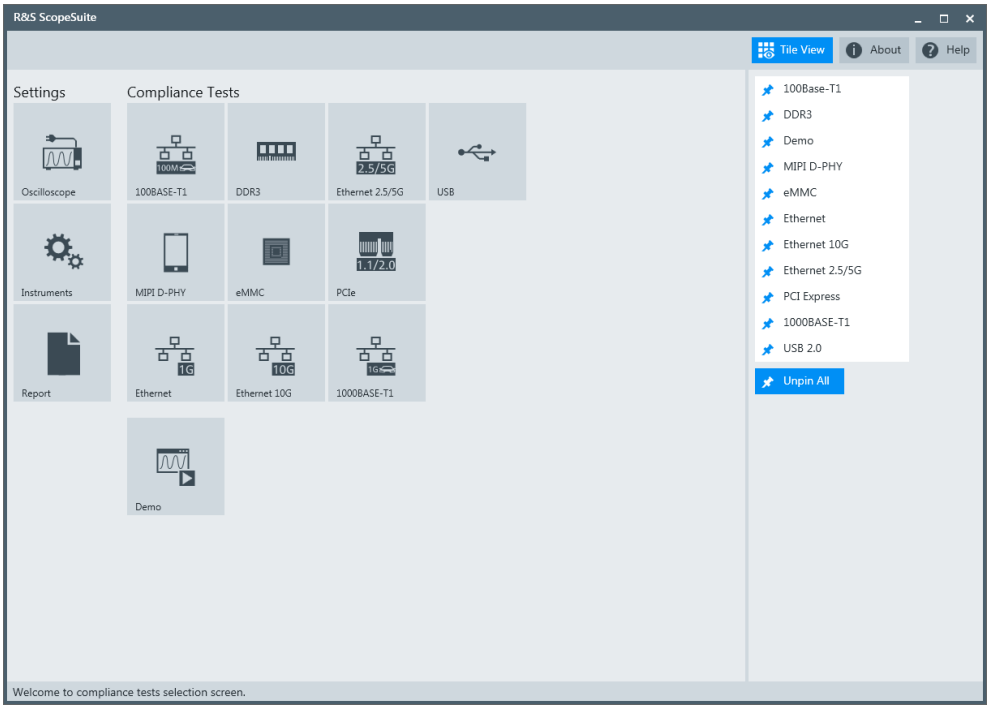
2 System Requirements for Remote Access and Automation

To control the R&S ScopeSuite running on a remote machine (oscilloscope or PC) via a remote programming interface, the following system configurations and requirements should be met:

- Software option R&S RTO/RTP-K99 for remote programming interface.
- R&S ScopeSuite with firmware version 3.8.0 and above must be installed on the machine.
- The oscilloscope must be fully licensed for the targeted compliance test options.
- The client computer must be able to access the machine (oscilloscope or a PC) via the network.
- The Windows Firewall on the machine may need be configured to allow the communication. For example:
  Start-> Control Panel -> Windows Firewall -> Allow programs to communicate through Windows Firewall -> Windows Communication Foundation
- The IP address of the machine should be known. It is part of the address for the `NetTcpBinding` configured in the client program.
- 8734 is the default port number used by `ScopeSuiteService` on the machine. It is part of the address for `NetTcpBInding` configured in the client program as well. Alternatively, if you need to use a different port number, simply drop it in `"C:\portForService.txt"` on the machine on which the R&S ScopeSuite is running.

Starting the Automation Option

1. Launch the R&S ScopeSuite. The `ScopeSuiteService` turns on automatically and ready for connection.



2. If needed, click "Oscilloscope" and configure the settings.
3. If needed, click "Instruments" and configure the settings.
4. If you want to automate any VNA test cases, pre-calibration procedure is required. More specific, in R&S ScopeSuite, execute the VNA test cases without expert mode so that the VNA calibration is performed and the settings are saved. VNA test automation uses the saved pre-calibration data with expert mode turned on.
5. Run your client program.

3 Requirements for Client Development

For developing the client program, `ScopeSuiteService.dll` need to be referenced. The client can be an application written in .NET C# or other .NET languages.

A simple client executable and sample code in C# are provided.

4 Client Programming with Duplex Communication

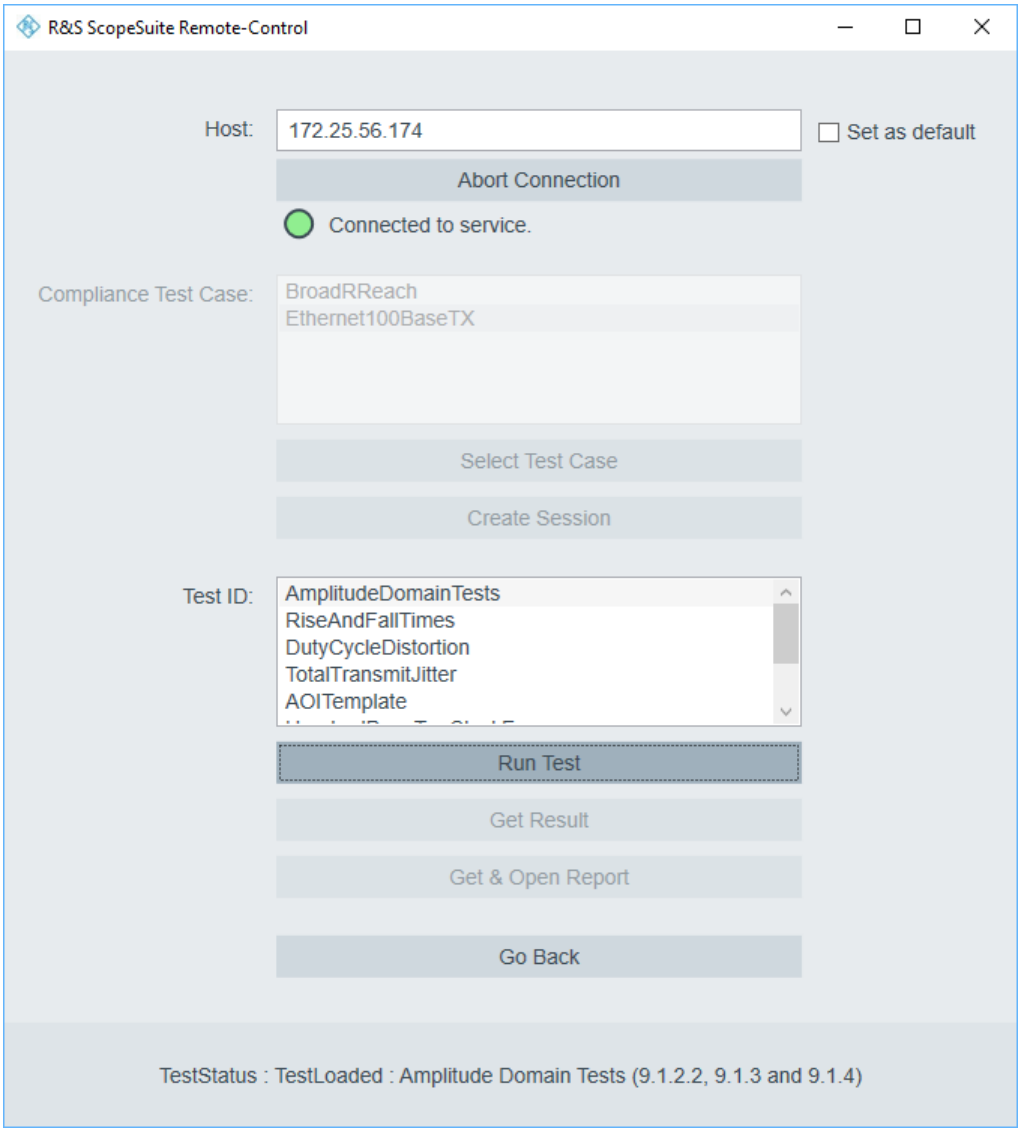
The client in general does the following to consume `ScopeSuiteService` and perform server-side operations remotely.

- Adds `RSScopeSuiteService.dll` as reference
- Implements `ICallback` to establish the connection with service and handle callback messages.
- Call APIs defined in `IService` to control the R&S ScopeSuite remotely. For most of the API operations, the service will callback to notify the client when the targeted operation is complete or any state information needs an update.

Refer to sample code and RSServiceInterface.chm for details.

5 Remote Control Client

The RemoteControlClient can be used to test the `ScopeSuiteService` and the client-service connection. It is also an example of a C# based remote client with a source code provided.



6 Client Program API Workflow and R&S ScopeSuite States

The client program connects to the service and controls the R&S ScopeSuite to perform tests based on session management.

There are three `ScopeSuiteState`:

- `HomePage`:
- `SessionSelectionPage`:
- `SessionManagementPage`:

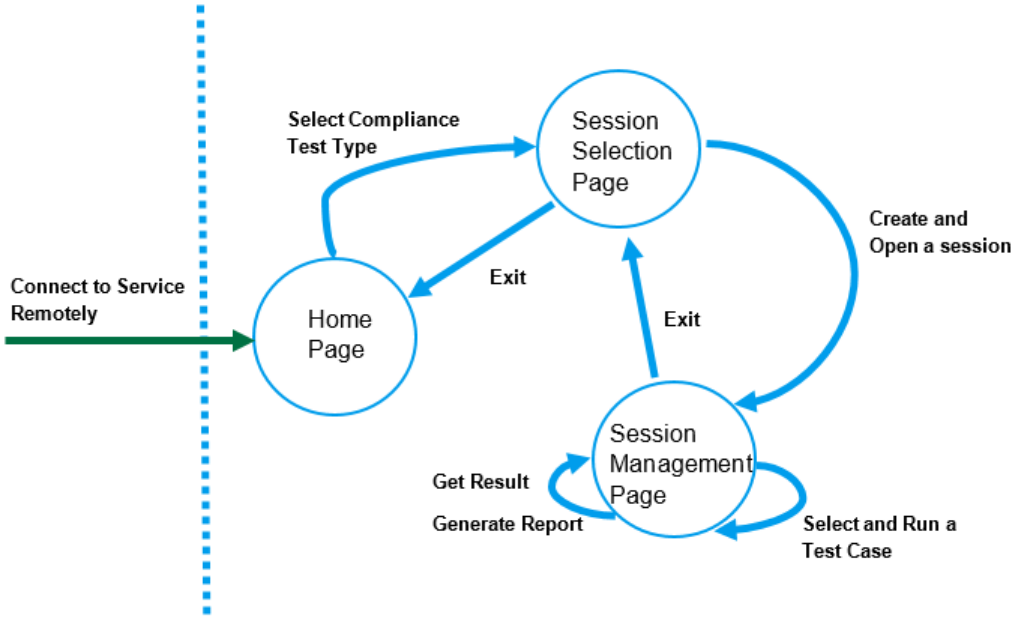


Figure 6-1: R&S ScopeSuite States

An example of the client program with API workflow can look as follows:

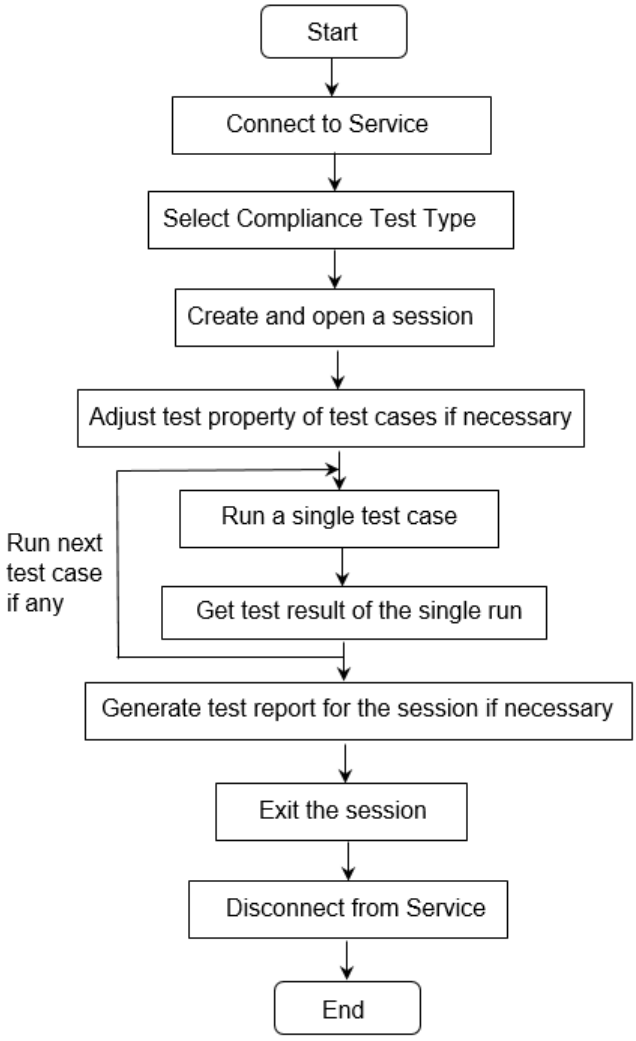


Figure 6-2: Example of an API workflow

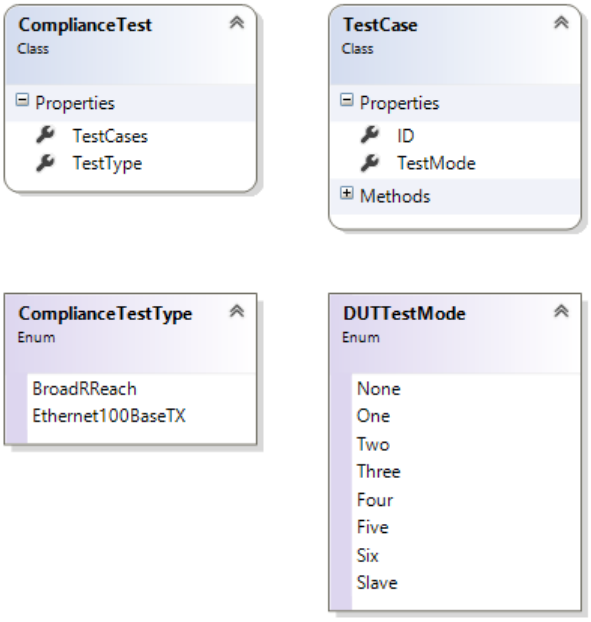
7 Programming Interface and Sample Code

`RSScopeSuiteService` has three components exposed to the client as programming interfaces: `TestData`, `IService` and `ICallback`

7.1 RSScopeSuiteService.TestData

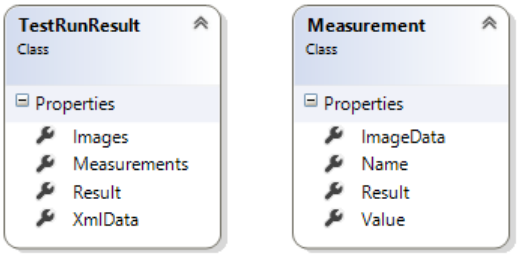
7.1.1 Compliance Test Options and Test Cases

The client may call `IService.GetComplianceTests` to get the full lists of compliance test options and test cases supported by remote control in test automation.



7.1.2 Test Result

The client may call `IService.GetSingleTestRunResult` to get the test result from the latest single execution of a test case.

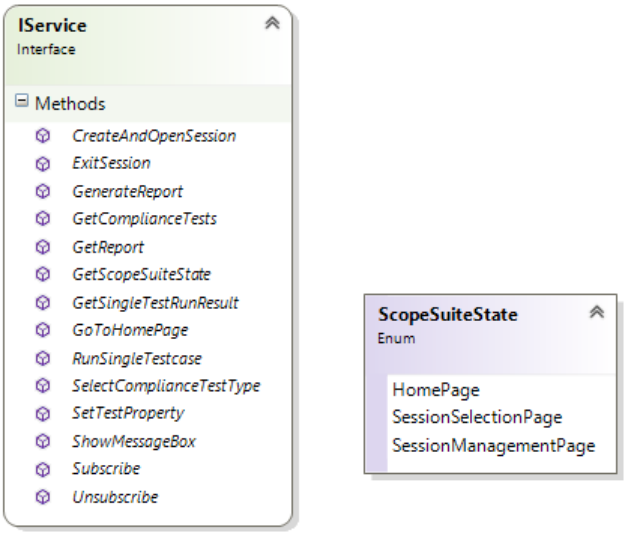


`TestRunResult.Result` gives the overall pass/fail result from the latest single test case execution.

The client may get the list of Measurement from the latest single test case execution via `TestRunResult.Measurements`. Instead, the client may parse the test result from XmlData and Images, which are consistent with the test result information in the session folder.

7.2 RSScopeSuiteService.IService

`IService` exposes the following APIs to the client:



- `CreateAndOpenSession`
  Creates and opens a new session from `SessionSelectionPage` of the R&S ScopeSuite.
  The `complianceType` must match the string of `TestData.ComplianceTest.ComplianceTestType`.
- `ExitSession`
  Exits from the current session. The R&S ScopeSuite returns to the `SessionSelectionPage`.
- `GenerateReport`

Generates a report from the `SessionManagementPage` page of the R&S ScopeSuite.
While the report is generating, service callbacks with messages that start with `"ReportStatus : "`.

- `GetComplianceTest`
  Gets a list of compliance test options and test cases supported by remote control in test automation.
- `GetReport`
  Gets a report from the current session, if the report has been generated successfully.
- `GetScopeSuiteState`
  Gets the state of the R&S ScopeSuite as a `ScopeSuiteState`.
- `GetSingleTestRunResult`
  Get `TestResult` from a single run of a single test case execution.
- `GoToHomePage`
  Goes back to the `HomePage` from the `SessionSelectionPage`.
- `RunSingleTestcase`
  Selects and run a single test case from `SessionManagementPage` of the R&S ScopeSuite.
  `Testcase` must match `TestData.TestCase.ID`.
  When the test case is running, service callbacks with messages that start with `"TestStaus : "`.
- `SelectComplianceTestType`
  Selects a compliance test type from the `HomePage` of the R&S ScopeSuite.
- `SetTestProperty`
  Sets the test property of the specific test case in the `SessionManagementPage`.
  Below is an example in the property file of a test case and it can be found in session folder. The property name and value to be passed in to the API must match those in the property file. The limits are available as well.

```
<TestPropertiesConfig>
 <HdMode showinreport="true" display="HD Mode" unit="">true
</HdMode>
 <ExpertMode showinreport="true" display="Expert Mode"
unit="" type="B">false
</ExpertMode>
</TestPropertiesConfig>
```

- `ShowMessageBox`
  Enables or disables the message box to pop up. It is disabled when the client is connected to the service and enabled when it is disconnected, automatically by the R&S ScopeSuite.
- `Subscribe`
  Establishes the connection to the service after a channel is created.
- `Unsubscribe`
  Closes the connection to the service.

7.3 RSScopeSuiteService.ICallback

7.3.1 Client Implements ICallback

`ICallback` is the callback contract to `IService`. The client must implement `ICallback` and initialize an instance of System.ServiceModel.DuplexChannelFactory with the remote address and configurations. The channel proxy of `IService` is then created. Below is an example of implementation:

```
[CallbackBehavior()]
public class ScopeSuiteCallback : ICallback
{
  public IService Proxy = null;

  public bool Connect()
  {
    var binding = new NetTcpBinding("UserServiceBinding");
    var factoryBack = new DuplexChannelFactory <IService>
    (this, binding, $"net.tcp://{host}:{port}/User");
    Proxy = factoryBack.CreateChannel();
    return (Proxy.Subscribe());
  }

  public void CallClient(string message)
  {
    // Handle callback message
  }
}
```

Below is an example of the client configuration. Instead, it can be configured in the program.

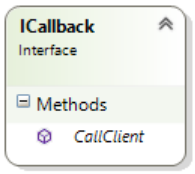
```
<system.serviceModel>
 <bindings>
  <netTcpBinding>
   <binding name="UserServiceBinding" closeTimeout="10:00:00" openTimeout="10:00:00"
                receiveTimeout="10:00:00" sendTimeout="10:00:00"
                maxBufferPoolSize="2147483647" maxBufferSize="2147483647"
                maxReceivedMessageSize="2147483647">
    <security mode="None"></security>
   </binding>
  </netTcpBinding>
 </bindings>
 <client>
  <endpoint binding="netTcpBinding" bindingConfiguration="UserServiceBinding"
                    contract="RSScopeSuiteService.IService">
  </endpoint>
 </client>
</system.serviceModel>
```

The proxy is used by the client to connect to the service and control the R&S ScopeSuite via `IService` APIs. Below is the sample code continued.

```
Public ScopeSuiteCallback callbackProxy = new ScopeSuiteCallback ();
bool ret = callbackProxy.Connect(); // connects to service
callbackProxy.Proxy.CreateAndOpenSession();
// calls ScopeSuite to create a session and open it
```

7.3.2 ICallback.CallClient

`ICallback.CallClient` is the callback operation for the service to send messages to the client. The client handles the messages received in this operation.



During a test case execution, the messages could be `TestStatus` like:

- `TestReset`
- `TestLoading`
- `TestLoaded`
- `TestInProgress`
- `TestLaunchError`
- `TestFinished`
- `TestMessageReceived`
- `TestWarningReceived`
- `TestErrorReceived`
- `TestMeasurementsFailed`
- `TestAborted`
- etc.

During a change of a R&S ScopeSuite state, the message could also be `ScopeSuiteState`.

During report generation, the messages could be `ReportStatus` like:

- `InProgress`
- `Successful`
- `Failed`
- etc.