



AHEAD OF WHAT'S POSSIBLE™

Analog Devices, Inc.

[www.analog.com](http://www.analog.com)

# A2B Scripting Guide

<b>Document Status:</b>	<b>Approved</b>
<b>Approved By:</b>	<b>ASH</b>

**Revision List****Table 1: Revision List**

<b>Revision</b>	<b>Date</b>	<b>Description</b>
0.1	4-May-18	Draft Version for the Script Detailed Design
0.2	16-May-18	Updated documents based on the feedback from PM/PL
0.3	5-Jun-18	Minor typo corrections
1.0	6-June-18	Baselined for Rel19.0.0
1.1	22-Aug-19	Updated comments for all API, Functions
1.2	22-Aug-19	Added navigation page and minor typo corrections.
1.3	30-Aug-19	Minor corrections
2.0	30-Aug-19	Approved and Baselined for Rel 19.3.0
2.1	20-Apr-22	Added Export APIs for BCF, NCF and Command list
3.0	25-Apr-22	Approved and Baselined for Rel 19.4.3

## **Copyright, Disclaimer Statements**

### **Copyright Information**

Copyright (c) 2009-2019 Analog Devices, Inc. All Rights Reserved. This software is proprietary and confidential to Analog Devices, Inc. and its licensors. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

### **Disclaimer**

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

## Table of Contents

<b>Revision List.....</b>	<b>2</b>
<b>Copyright, Disclaimer Statements .....</b>	<b>3</b>
<b>Table of Contents .....</b>	<b>4</b>
<b>List of Figures .....</b>	<b>6</b>
<b>List of Tables .....</b>	<b>6</b>
<b>List of Equations .....</b>	<b>6</b>
<b>1 Introduction .....</b>	<b>7</b>
1.1 Purpose .....	7
1.2 Scope .....	7
<b>2 Scripting Interfaces.....</b>	<b>8</b>
2.1 IScripted Interface.....	8
2.2 SigmaStudio Server .....	8
2.2.1 Environment setup for win32com Clients.....	8
2.2.1.1 Using it as a win32com Client.....	8
2.2.1.2 Pre-condition: - .....	9
2.2.2 Using ActivePython to access the server APIs .....	9
2.2.2.1 Environment setup for Python.....	9
2.2.3 Other Supported win32com Clients.....	10
2.2.3.1 Matlab .....	10
2.2.3.2 LabView .....	10
<b>3 Function Definitions.....</b>	<b>11</b>
3.1 General SigmaStudio Functions.....	11
3.1.1 Return Type .....	11
3.1.2 Script Run .....	11
3.1.3 Script Delay.....	11
3.1.4 Enable Logging Mode .....	11
3.2 Project File Interface .....	12
3.2.1 Create .....	12
3.2.2 Open .....	12
3.2.3 Save/Save As .....	12
3.2.4 Close.....	13
3.2.5 Set Project as active .....	13
3.2.6 Link .....	13
3.3 A2BScript API definitions .....	14
3.3.1 A2BMasterICRegisterWrite .....	15

3.3.2 A2BMasterICRegisterRead .....	15
3.3.3 A2BSlaveICRegisterWrite .....	16
3.3.4 A2BSlaveICRegisterRead .....	16
3.3.5 A2BSlavePerilICRegisterWrite .....	17
3.3.6 A2BSlavePerilICRegisterRead .....	18
3.3.7 A2BGetNetworkDiscoveryStatus .....	18
3.3.8 A2BGetSlaveNodeActiveStatus .....	19
3.3.9 A2BGetNetworkLineFaultStatus .....	19
3.3.10 A2BUserCommand .....	20
3.3.11 A2BStartBERTest .....	20
3.3.12 A2BStopBERTest .....	20
3.3.13 A2BGetBERCount .....	20
3.3.14 A2BResetNetwork .....	20
3.3.15 A2BResetSlaveNode .....	21
3.3.16 A2BGetNetworkBandwidthUsage .....	21
3.3.17 A2BGetNodeBandwidthUsage .....	21
3.3.18 A2BExportBusConfigCFile .....	21
3.3.19 A2BExportBusConfigDatFile .....	22
3.3.20 A2BExportBusConfigXMLFile .....	22
3.3.21 A2BExportCommandListXMLFile .....	22
3.3.22 A2BExportCommandListHFile .....	22
3.3.23 A2BExportMasterNodeConfigXMLFile .....	23
3.3.24 A2BExportSlaveNodeConfigXMLFile .....	23
3.4 SigmaStudio Server APIs .....	23
3.4.1 A2B_MASTER_REGISTER_WRITE .....	23
3.4.2 A2B_MASTER_REGISTER_READ .....	23
3.4.3 A2B_SLAVE_REGISTER_WRITE .....	23
3.4.4 A2B_SLAVE_REGISTER_READ .....	24
3.4.5 A2B_SLAVE_PERIREGISTER_WRITE .....	24
3.4.6 A2B_SLAVE_PERIREGISTER_READ .....	25
3.4.7 A2B_GET_NETWORK_DISCOVERY_STATUS .....	25
3.4.8 A2B_GET_SLAVE_NODE_ACTIVE_STATUS .....	25
3.4.9 A2B_GET_NETWORK_LINEFAULT_CODE .....	25
3.4.10 A2B_GET_NETWORK_LINEFAULT_STATUS .....	26
3.4.11 A2B_USER_COMMAND .....	26
3.4.12 A2B_START_BER_TEST .....	26
3.4.13 A2B_STOP_BER_TEST .....	27
3.4.14 A2B_GET_BER_COUNT .....	27
3.4.15 A2B_RESET_NETWORK .....	27
3.4.16 A2B_RESET_SLAVE_NODE .....	27
3.4.17 A2B_GET_NETWORK_BANDWIDTH_USAGE .....	27
3.4.18 A2B_GET_NODE_BANDWIDTH_USAGE .....	28

3.4.19 A2B_EXPORT_BUSCONFIG_C_FILE.....	28
3.4.20 A2B_EXPORT_BUSCONFIG_DAT_FILE .....	28
3.4.21 A2B_EXPORT_BUSCONFIG_XML_FILE .....	29
3.4.22 A2B_EXPORT_COMMANDLIST_XML_FILE .....	29
3.4.23 A2B_EXPORT_COMMANDLIST_H_FILE.....	29
3.4.24 A2B_EXPORT_MAIN_NODE_CONFIG_XML_FILE .....	29
3.4.25 A2B_EXPORT_SLAVE_NODE_CONFIG_XML_FILE .....	30
<b>4 Accessing from LabView .....</b>	<b>31</b>
<b>Terminology.....</b>	<b>35</b>
<b>References.....</b>	<b>35</b>
<b>List of Figures</b>	
Figure 1: A2B Scripting Interface Overview .....	7
Figure 2: Tool palette in LabView .....	31
Figure 3: Server selection from .Net constructor .....	32
Figure 4: Methods exposed in SigmaServer .....	33
Figure 5: Example for Open Project VI .....	33
Figure 6: Example of A2B node discovery status check.....	34
Figure 7: Running Iscript from LabView .....	34
<b>List of Tables</b>	
Table 1: Revision List .....	2
Table 2: Terminology .....	35
Table 3: References .....	35
<b>List of Equations</b>	
<b>No table of figures entries found.</b>	

# 1 Introduction

A2B schematics are designed and configured over SigmaStudio graphical development environment. SigmaStudio has a highly intuitive user interface for A2B network design and Audio system development.

SigmaStudio defines a Microsoft .NET functional interface, *IScripted*, which provides control over the most common elements of SigmaStudio. Script files can be created and reused from within the SigmaStudio development environment. *SigmaStudioServer* is a software automation interface to SigmaStudio that allows external client applications to control and automate SigmaStudio functions from external software.

SigmaStudio is available for download at [1].

The overall system and its components are illustrated in Figure 1.

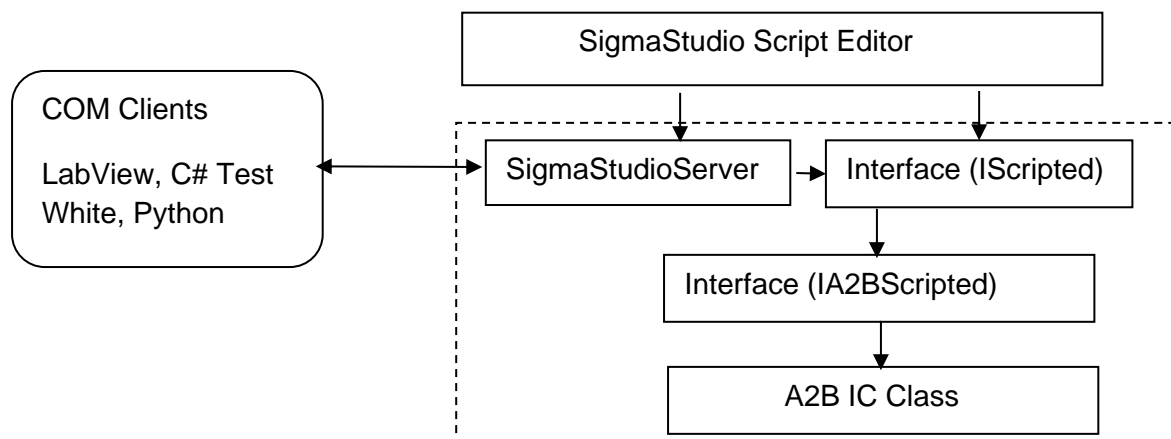


Figure 1: A2B Scripting Interface Overview

## 1.1 Purpose

The document provides information on the access methods to A2B scripting APIs and detailed design which enable diagnostics and automation using Script Editor, the Scripting interface and the SigmaStudio automation server (which enables LabView, Matlab, C# and Python).

## 1.2 Scope

This document is intended to assist A2B testing engineers and advanced design engineers. This document provides an overview of A2B SigmaStudio scripting interface and the access to SigmaStudio automation server for designing automated test cases and for diagnostics.

For generic SigmaStudio script APIs, usage and examples are available at

<https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting>

## 2 Scripting Interfaces

### 2.1 IScripted Interface

*Analog.SStudioScripting.IScripted* is contained in a .NET assembly, *BaseLib.dll*, installed in the SigmaStudio folder. This allows script files to be created and reused from within the SigmaStudio development environment.

Refer to the link below to understand how to create and run scripts using IScripted Editor built in SigmaStudio IDE.

<https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting/create>

The scripts developed and run from IScripted editor which is built in IDE has access to all reflection properties of .Net and rich with syntax features.

These scripts shall bear an extension “\*.sss”. All APIs in *SigmaStudioServer.dll* might not be exposed to win32 com interface. Still, these APIs can be accessed by calling the using the *RUN\_SCRIPT()* method which accesses these APIs internally.

For using via LabView refer the example Refer Section 4

For using via Python refer the example script Refer Section 2.2.2

### 2.2 SigmaStudio Server

An automation client can be used to access the objects, properties, methods, and events associated with the *SigmaStudioServer* interface. *SigmaStudioServer* is a .NET server as well as an ActiveX server.

To access the server interface, *SStudio.exe* and the client application should be launched on the same PC, and then the client application should point to the *Analog.SigmaStudioServer.dll* which is installed alongside *SStudio.exe* in the SigmaStudio program folder.

Refer to the link below to understand how to create and run scripts using SigmaServer which supports win32com Client communications to SigmaStudio IDE.

<https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting/server>

#### 2.2.1 Environment setup for win32com Clients

##### 2.2.1.1 Using it as a win32com Client

To use Python or LabView/Matlab as a SigmaStudio server client, both client and SigmaStudio [Server] shall be installed and running on the same machine.

### 2.2.1.2 Pre-condition: -

Launch SigmaStudio [Server] before running Client [Python or LabView/Matlab] If not it shall show There was no endpoint listening at net.pipe//localhost/SigmaServerPipe.

Result will be True if compile is successful.

```
import win32com
import win32com.client
server = win32com.client.dynamic.Dispatch("Analog.SigmaStudioServer.SigmaStudioServer")
projectFile="C:\\Work\\schematics\\Example_schematic.dspproj" #Just a dummy example
server.OPEN_PROJECT(projectFile)
Result = server.compile)
```

## 2.2.2 Using ActivePython to access the server APIs

Perform the following steps to access SigmaStudio server scripts using ActivePython.

### 2.2.2.1 Environment setup for Python

1. Install ActivePython (version 2.7.2 for Python 2.7).
2. Install PythonNet (version 2.0 for Python 2.7).
3. Set PYTHONPATH=SIGMASTUDIO\_INSTALLED\_FOLDER.
4. Launch python.exe from 'C:\\Python27\\pythonnet'.

Create a SigmaStudioServer instance and run APIs via it.

```
import clr
clr.AddReference('Analog.SigmaStudioServer')
from Analog.SigmaStudioServer import SigmaStudioServer
server=SigmaStudioServer()
server.OPEN_PROJECT(r'C: \Work\schematics\Example_schematic.dspproj')
server.RUN_SCRIPT_FILE( r'C: \Work\schematics\Example_schematic.sss');
```

## **2.2.3 Other Supported win32com Clients**

### **2.2.3.1 Matlab**

In this configuration, MATLAB is a COM automation client and SigmaStudio (via SigmaStudioServer) becomes a COM automation server. For COM operation, the SigmaStudioServer must first be registered as a COM object. Refer the below link for detailed information.

<https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting/matlab>

### **2.2.3.2 LabView**

To use SigmaStudio as a LabVIEW automation client, both applications must be installed and running on the same machine. In this configuration, LabVIEW is a .NET automation client and SigmaStudio (via SigmaStudioServer) becomes a .NET automation server. Refer the below link for detailed information.

<https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting/labview>

For step by step guide to test A2B automation APIs using LabView is mentioned in Section 4 .

## 3 Function Definitions

### 3.1 General SigmaStudio Functions

A list of general functions and their prototypes are given below.

#### 3.1.1 Return Type

The interface defines an integer return type, “HResult”, as follows:

HResult.S\_OK = 0

HResult.E\_FAILED = 1

HResult.E\_INVALID\_ARGS = 2

HResult.E\_EXCEPTION = 3

#### 3.1.2 Script Run

Run script

HResult ScriptRun( string script );

"script" = script code as a System.String

Run script file

HResult ScriptRunFile( string scriptFilename );

"scriptFilename" = The fully qualified script file name

#### 3.1.3 Script Delay

Pause script execution for delay Milliseconds

HResult ScriptDelay( int delayMilliseconds );

"delayMilliseconds" = Amount of delay in milliseconds

#### 3.1.4 Enable Logging Mode

Sets SigmaStudio in logging mode and disables all message pop-ups during the execution of the script.

HResult EnableLoggingMode(bool enabled);

"enabled" = Flag indicating whether to enable (on true) or disable (on false) logging mode.

## 3.2 Project File Interface

The following functions can be used to interface with a project file.

### 3.2.1 Create

Create a new project file

```
HResult ProjectNew();
```

The function takes no parameter

### 3.2.2 Open

Open a project file from disk

```
HResult ProjectOpen( string filename );
```

"filename" = A fully qualified file path

### 3.2.3 Save/Save As

Save the Active project file

```
HResult ProjectSave();
```

The function takes no parameter

Save a specific project file

```
HResult ProjectSave( string projectName );
```

"projectName" = An open project file's name or fully qualified path

Save as the Active project file

```
HResult ProjectSaveAs( string saveAsFilename );
```

"saveAsFilename" = A new file name or fully qualified path

Save as a specific project file

**HResult** ProjectSaveAs( **string** projectName, **string** saveAsFilename );

"projectName" = An open project file's name or fully qualified path

"saveAsFilename" = The new file name or fully qualified path

### 3.2.4 Close

Close the Active project file

**HResult** ProjectClose();

Close a specific project file

**HResult** ProjectClose( **string** projectName );

"projectName" = An open project file's name or fully qualified path

### 3.2.5 Set Project as active

Set a project as the active project

**HResult** ProjectSetActive( **string** projectName );

"projectName" = An open project file's name or fully qualified path

### 3.2.6 Link

Link the active Schematic

**HResult** ProjectLink();

The function takes no parameter

Link and compile the active Schematic

**HResult** ProjectLinkCompile();

The function takes no parameter

Link and compile the active Schematic

**HResult** ProjectLinkCompile( **string** projectName );

"projectName" = An open project file's name or fully qualified path

Link, compile and download the active Schematic

```
HResult ProjectLinkCompileDownload();
```

The function takes no parameter

Link, compile and download a specific project file

```
HResult ProjectLinkCompileDownload( string projectName );
```

"projectName" = An open project file's name or fully qualified path

For general queries on SigmaStudio usage and sigmastudio script APIs refer [2]

### 3.3 A2BScript API definitions

Parameter configuration examples:

<param name="ICName"> Name of A2B IC </param>

Ex: From the below figure in the SigmaStudio schematic, hardware configuration tab find the IC name

ICName = "IC 1"

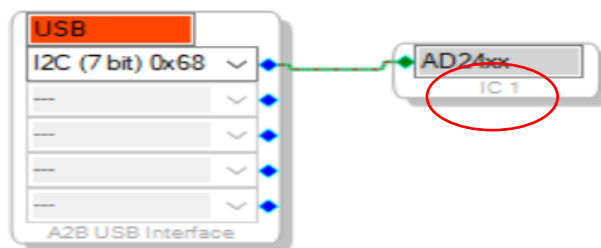


Figure 2 A2B-USBi

<param name="writeAddress">The register address to write</param>

<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>

Ex: masterAddress: 0x68, busAddress: 0x69

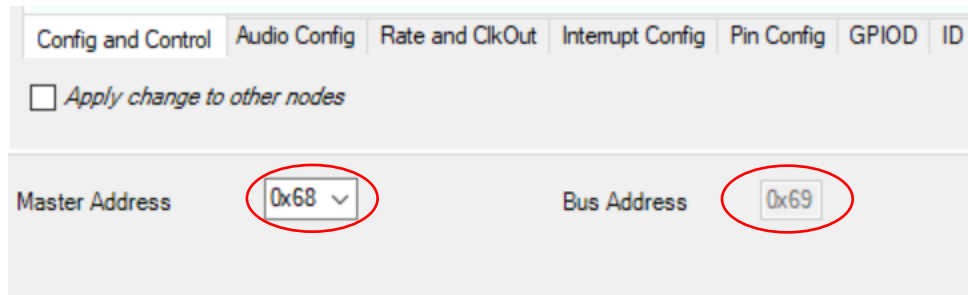


Figure 3 Master and Bus Address

**Note:** Call ProjectLinkCompile API before calling Export APIs for BCF, Command List and NCF.

### 3.3.1 A2BMasterICRegisterWrite

```
<summary> Write data to an A2B Master Transceiver register</summary>
<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="writeAddress">The register address to write</param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the transceiver</param>
<param name="dataToWrite"> Array of data bytes to write </param>
HRESULT A2BMasterICRegisterWrite(string ICName, int masterAddress, int writeAddress, int writeNumberBytes, byte[] dataToWrite);
```

```
<summary> Write data to an A2B Master Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="writeAddress">The register address to write</param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the transceiver</param>
<param name="dataToWrite"> data bytes to write packed as long </param>
HRESULT A2BMasterICRegisterWrite(string ICName, int masterAddress, int writeAddress, int writeNumberBytes, long dataToWrite);
```

### 3.3.2 A2BMasterICRegisterRead

```
<summary> Read data from an A2B Master Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="readAddress">The register address to read</param>
<param name="readNumberBytes">Number of bytes to read from the transceiver</param>
<param name="bytesRead">Return data read from the transceiver if successful</param>
HRESULT A2BMasterICRegisterRead(string ICName, int masterAddress, int readAddress, int readNumberBytes, ref byte[] bytesRead);
```

```

<summary> Read data from an A2B Master Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="readAddress">The register address to read</param>
<param name="readNumberBytes">Number of bytes to read from the transceiver</param>
<param name="bytesRead">Return data read from the transceiver if successful</param>
HResult A2BMasterICRegisterRead(string ICName, int masterAddress, int readAddress, int
readNumberBytes, out long bytesRead);

```

### 3.3.3 A2BSlaveICRegisterWrite

```

<summary> Write data to an A2B Slave Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="writeAddress">The register address to write </param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the transceiver
</param>
<param name="dataToWrite"> Array of data bytes to write </param>
HResult A2BSlaveICRegisterWrite(string ICName, int masterAddress, int busAddress, int nodeID,
int writeAddress, int writeNumberBytes, byte[] dataToWrite);

```

```

<summary> Write data to an A2B Slave Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="writeAddress">The register address to write </param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the transceiver
</param>
<param name="dataToWrite"> data bytes to write packed as long </param>
HResult A2BSlaveICRegisterWrite(string ICName, int masterAddress, int busAddress, int nodeID,
int writeAddress, int writeNumberBytes, long dataToWrite);

```

### 3.3.4 A2BSlaveICRegisterRead

```

<summary> Read data from an A2B Slave Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress">A2B Master Transceiver I2C address (7-bit) </param>

```

```

<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name="nodeID"> Slave Node position in the schematic [Linear schematic] 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="readAddress">The register address to read</param>
<param name="readNumberBytes">Number of bytes to read from the transceiver</param>
<param name="bytesRead">Return data read from the transceiver if successful</param>
HResult A2BSlaveICRegisterRead(string ICName, int masterAddress, int busAddress, int nodeID,
int readAddress, int readNumberBytes, ref byte[] bytesRead);

```

```

<summary> Read data from an A2B Slave Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress">A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name="nodeID"> Slave Node position in the schematic [Linear schematic] 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="readAddress">The register address to read</param>
<param name="readNumberBytes">Number of bytes to read from the transceiver</param>
<param name="dataRead">Return data read from the transceiver if successful</param>
HResult A2BSlaveICRegisterRead(string ICName, int masterAddress, int busAddress, int nodeID,
int readAddress, int readNumberBytes, out long dataRead);

```

### 3.3.5 A2BSlavePerilCRegisterWrite

```

<summary> Write data to register of a peripheral connected to an A2B Slave
Transceiver</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name="nodeID"> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
< param name="chipAddress"> set device address (7-bit) for peripheral (CHIP) in Slave</param>
<param name="writeAddress">The register address to write</param>
<addrWidth name="addrWidth"> Address width of peripheral IC</param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the
peripheral</param>
<param name="dataToWrite"> Array of data bytes to write </param>
HResult A2BSlavePerilCRegisterWrite(string ICName, int masterAddress, int busAddress, int
nodeID, int chipAddress, int writeAddress, int addrWidth, int writeNumberBytes, byte[] dataToWrite);

```

```

<summary> Write data to register of a peripheral connected to an A2B Slave
Transceiver</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>

```

```

< param name="nodeID"> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
< param name="chipAddress"> set device address(7-bit) for peripheral (CHIP) in Slave</param>
<param name="writeAddress">The register address to write</param>
< param name="addrWidth"> Address width of peripheral IC</param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the
peripheral</param>
<param name="dataToWrite"> data bytes to write packed as long </param>
HResult A2BSlavePerilCRegisterWrite(string ICName, int masterAddress, int busAddress, int
nodeID, int chipAddress, int writeAddress, int addrWidth, int writeNumberBytes, long dataToWrite);

```

### 3.3.6 A2BSlavePerilCRegisterRead

```

<summary>Read data from a register of a peripheral connected to an A2B Slave
Transceiver</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
< param name=" chipAddress "> set device address (7-bit) for peripheral (CHIP) in Slave</param>
<param name="readAddress">The register address to read</param>
< param name=" addrWidth "> Address width of peripheral IC</param>
<param name="readNumberBytes">Number of bytes to read from the peripheral</param>
<param name="bytesRead">Return data read from the peripheral if successful</param>
HResult A2BSlavePerilCRegisterRead(string ICName, int masterAddress, int busAddress, int
nodeID, int chipAddress, int readAddress, int addrWidth, int readNumberBytes, ref byte[]
bytesRead);

```

```

<summary>Read data from a register of a peripheral connected to an A2B Slave
Transceiver</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
< param name=" chipAddress "> set device address (7-bit) for peripheral (CHIP) in Slave</param>
<param name="readAddress">The register address to read</param>
< param name=" addrWidth "> Address width of peripheral IC</param>
<param name="readNumberBytes">Number of bytes to read from the peripheral</param>
<param name=" dataRead ">Return data read from the peripheral if successful</param>
HResult A2BSlavePerilCRegisterRead(string ICName, int masterAddress, int busAddress, int
nodeID, int chipAddress, int readAddress, int addrWidth, int readNumberBytes, out long dataRead);

```

### 3.3.7 A2BGetNetworkDiscoveryStatus

```

<summary>Get A2B network discovery status</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="discStatus">Return status: True if the network if successful else False</param>
<param name="numDiscSlaves">Return number of nodes discovered in the network</param>
HResult A2BGetNetworkDiscoveryStatus(string ICName, int masterAddress, out bool discStatus,
out int numDiscSlaves);

```

### 3.3.8 A2BGetSlaveNodeActiveStatus

```

<summary>Get slave node discovery status</summary>
<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="bActiveStatus"> Node status: True if active, False if inactive </param>
<returns></returns>
HResult A2BGetSlaveNodeActiveStatus(string ICName, int masterAddress, int busAddress, int
nodeID, out bool bActiveStatus);

```

### 3.3.9 A2BGetNetworkLineFaultStatus

```

<summary>Get network line fault status returns fault code, fault node type and fault node
position</summary>
<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="faultCode">Fault code from Interrupt Type Register</param>
<param name="faultNodeType">Fault node type : 0-Master, 1-Slave </param>
<param name="faultNodePosition">Fault Node Position : 0 for Master, 0 for 1st Slave, 1 for 2nd
Slave etc </param>
<returns></returns>
HResult A2BGetNetworkLineFaultStatus(string ICName, int masterAddress, out int faultCode, out
int faultNodeType, out int faultNodePosition);

```

```

<summary> Get network line fault status as string, contains fault code, fault node type and fault
node position</summary>
<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="faultDescription">Return fault description that contains the below as string
"faultCode">Fault code from Interrupt Type Register
"faultNodeType">Fault node type : 0-Master, 1-Slave
"faultNodePosition">Fault Node Position: 0 for Master, 0 for 1st Slave, 1 for 2nd Slave etc </param>
<returns></returns>
HResult A2BGetNetworkLineFaultStatus(string ICName, int masterAddress, out string
faultDescription);

```

### 3.3.10 A2BUserCommand

```
<summary>Generic command to perform write for A2B</summary>
<param name="ICName">Name of A2B IC </param>
<param name="cmd">Command string</param>
<param name="ParamIn">The data to write, input parameters</param>
<param name="ResOut">Return result out if command execution is successful</param>
<returns></returns>
HRESULT A2BUserCommand(string ICName, string cmd, byte[] paramIn, ref byte[] resOut);
```

### 3.3.11 A2BStartBERTest

```
<summary>Start BER Test</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="BERMode">BER Mode: 0-PRBS, 1-Audio; For PRBS, node-to-node checking is enabled by default; For Audio, all bit errors in BECCTL are enabled by default</param>
HRESULT A2BStartBERTest(string ICName, int masterAddress, int BERMode);
```

### 3.3.12 A2BStopBERTest

```
<summary>Stop BER Test</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
HRESULT A2BStopBERTest(string ICName, int masterAddress);
```

### 3.3.13 A2BGetBERCount

```
<summary>Read BER Count</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="BERMode">BER Mode: 0-PRBS, 1-Audio; For PRBS, node-to-node checking is enabled by default; For Audio, all bit errors in BECCTL are enabled by default</param>
<param name="BERCount"> Returns BER count based on the mode configured if successful. [0]- Master Node error count, [1] – slave node 0 error count, [2] slave node 1 error count and so on. </param>
<param name="nNodeCount">Returns the number of active nodes (including Master) for which the error count is returned in BERCount</param>
HRESULT A2BGetBERCount(string ICName, int masterAddress, int BERMode, out uint[] BERCount, out int nNodeCount);
```

### 3.3.14 A2BResetNetwork

```
<summary>Issue Soft Reset at Master which will reset whole network</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<returns></returns>
HRESULT A2BResetNetwork(string ICName, int masterAddress);
```

### 3.3.15 A2BResetSlaveNode

```
<summary>Issue Soft Reset to an individual slave</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic], 0 for 1st slave
and 1 for 2nd slave etc </param>
HRESULT A2BResetSlaveNode(string ICName, int masterAddress, int busAddress, int nodeID);
```

### 3.3.16 A2BGetNetworkBandwidthUsage

```
<summary> Get bandwidth usage at Master node</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="usage"></param>returns network bandwidth usage of at master node. [0]- 'B' side
downstream BW, [1] – 'B' side upstream BW.
<returns></returns>
HRESULT A2BGetNetworkBandwidthUsage(string ICName, int masterAddress, out double[] usage);
```

### 3.3.17 A2BGetNodeBandwidthUsage

```
<summary>Get Bandwidth at individual slave node </summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic] 0 for 1st
Slave, 1 for 2nd Slave etc </param>
<param name="usage"></param>returns the node bandwidth usage of individual slave. [0] – 'B'
side downstream BW, [1] – 'B' side upstream BW, [2] – 'A' side downstream BW and [3] – 'A' side
upstream BW.
<returns></returns>
HRESULT A2BGetNodeBandwidthUsage(string ICName, int masterAddress, int busAddress, int
nodeID, out double[] usage);
```

### 3.3.18 A2BExportBusConfigCFile

```
<summary>Export API for Bus Configuration C file</summary>
<param name="bcfCFileName"> Bus configuration File name(*.C) with path</param>
<param name="bcfVersion"> BCF File Version</param>
<param name="optimizeForMemory"> Flag to enabled/disabled memory optimization in BCF
during export</param>
<param name="bcfCompressed"> Flag to enabled/disabled the compressed bcf during export
</param>
<returns></returns>
bool A2BExportBusConfigCFile(string bcfCFileName, string bcfVersion, bool optimizeForMemory,
bool bcfCompressed);
```

**Note:** Call ProjectLinkCompile API before calling Export APIs

### 3.3.19 A2BExportBusConfigDatFile

```
<summary>Export API for Bus Configuration DAT file</summary>
<param name="bcfDatFileName">Bus configuration File(*.DAT) name with path</param>
<param name="streamInfoEnable"> Flag to export the Stream Information to DAT File</param>
<returns></returns>
bool A2BExportBusConfigDatFile(string bcfDatFileName, bool streamInfoEnable);
```

### 3.3.20 A2BExportBusConfigXMLFile

```
<summary>Export API for Bus Configuration XML file</summary>
<param name="bcfXMLFileName">Bus configuration File(*.XML) name with path</param>
<param name="bcfVersion">BCF File Version</param>
<param name="optimizeForMemory">Flag to enabled/disabled memory optimization in BCF during export</param>
<param name="bcfCompressed">Flag to enabled/disabled the compressed bcf during export</param>
<returns></returns>
bool A2BExportBusConfigXMLFile(string bcfXMLFileName, string bcfVersion, bool optimizeForMemory, bool bcfCompressed);
```

### 3.3.21 A2BExportCommandListXMLFile

```
<summary>Export API for Command List XML file</summary>
<param name="cmdListXMLFileName">Command List File(*.XML) name with path</param>
<param name="sVersion">Command List Version</param>
<param name="includeModifiedRegister"> Flag to include modified registers only </param>
<param name="includePeriphConfigData"> Flag to include peripheral configuration data</param>
<param name="optimizeForMemory">Flag to enabled/disabled memory optimization in Command List during export</param>
<returns></returns>
bool A2BExportCommandListXMLFile(string cmdListXMLFileName, string sVersion, bool includeModifiedRegister, bool includePeriphConfigData, bool optimizeForMemory);
```

### 3.3.22 A2BExportCommandListHFile

```
<summary>Export API for Command List H file</summary>
<param name="cmdListXMLFileName">Command List File(*.H) name with path</param>
<param name="sVersion">Command List Version</param>
<param name="includeModifiedRegister"> Flag to include modified registers only </param>
<param name="includePeriphConfigData"> Flag to include peripheral configuration data</param>
<param name="optimizeForMemory">Flag to enabled/disabled memory optimization in Command List during export</param>
<returns></returns>
bool A2BExportCommandListHFile(string cmdListHFileName, string sVersion, bool includeModifiedRegister, bool includePeriphConfigData, bool optimizeForMemory);
```

### 3.3.23 A2BExportMasterNodeConfigXMLFile

```
<summary>Export API for Main Node Configuration File(NCF.xml)</summary>
<param name="sNodeConfigXMLName"> Node Configuration file name(*.xml) with path</param>
<returns></returns>
bool A2BExportMasterNodeConfigXMLFile(string sNodeConfigXMLName);
```

### 3.3.24 A2BExportSlaveNodeConfigXMLFile

```
<summary>Export API for Sub Node Configuration File(NCF.xml)</summary>
<param name="sNodeConfigXMLName">Node Configuration file name(*.xml) with path</param>
<param name="nSlaveNodeId"> Sub node ID</param>
<returns></returns>
bool A2BExportSlaveNodeConfigXMLFile(string sNodeConfigXMLName, ushort nSlaveNodeId);
```

## 3.4 SigmaStudio Server APIs

### 3.4.1 A2B\_MASTER\_REGISTER\_WRITE

```
<summary> Write data to an A2B Master Transceiver register</summary>
<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="writeAddress">The register address to write</param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the transceiver</param>
<param name="dataToWrite"> Array of data bytes to write</param>
public bool A2B_MASTER_REGISTER_WRITE(string ICName, int masterAddress, int writeAddress, int writeNumberBytes, byte[] dataToWrite)
```

### 3.4.2 A2B\_MASTER\_REGISTER\_READ

```
<summary> Read data from an A2B Master Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="readAddress">The register address to read</param>
<param name="readNumberBytes">Number of bytes to read from the transceiver</param>
<param name="bytesRead">Return data read from the transceiver if successful</param>
public bool A2B_MASTER_REGISTER_READ(string ICName, int masterAddress, int readAddress, int readNumberBytes, ref byte[] bytesRead)
```

### 3.4.3 A2B\_SLAVE\_REGISTER\_WRITE

```
<summary> Write data to an A2B Slave Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
```

```

< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="writeAddress">The register address to write </param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the transceiver
</param>
<param name="dataToWrite"> Array of data bytes to write </param>
public bool A2B_SLAVE_REGISTER_WRITE(string ICName, int masterAddress, int busAddress,
int nodeID, int writeAddress, int writeNumberBytes, byte[] dataToWrite)

```

### 3.4.4 A2B\_SLAVE\_REGISTER\_READ

```

<summary> Read data from an A2B Slave Transceiver register</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress">A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name="nodeID"> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
<param name="readAddress">The register address to read</param>
<param name="readNumberBytes">Number of bytes to read from the transceiver</param>
<param name="bytesRead">Return data read from the transceiver if successful</param>
public bool A2B_SLAVE_REGISTER_READ(string ICName, int masterAddress, int busAddress,
int nodeID, int readAddress, int readNumberBytes, ref byte[] bytesRead)

```

### 3.4.5 A2B\_SLAVE\_PERIREGISTER\_WRITE

```

<summary> Write data to register of a peripheral connected to an A2B Slave
Transceiver</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name="nodeID"> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave,
1 for 2nd Slave etc </param>
< param name="chipAddress"> set device address (7-bit) for peripheral (CHIP) in Slave</param>
<param name="writeAddress">The register address to write</param>
<addrWidth name="addrWidth"> Address width of peripheral IC</param>
<param name="writeNumberBytes">Number of bytes in 'dataToWrite' to write to the
peripheral</param>
<param name="dataToWrite"> Array with the data bytes to write </param>
public bool A2B_SLAVE_PERIREGISTER_WRITE(string ICName, int masterAddress, int
busAddress, int nodeID, int chipAddress, int writeAddress, int addrwidth, int writeNumberBytes,
byte[] dataToWrite)

```

### 3.4.6 A2B\_SLAVE\_PERIREGISTER\_READ

<summary>Read data from a register of a peripheral connected to an A2B Slave Transceiver</summary>

<param name="ICName">Name of A2B IC </param>

<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>

<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>

< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave, 1 for 2<sup>nd</sup> Slave etc </param>

< param name=" chipAddress "> set device address (7-bit) for peripheral (CHIP) in Slave</param>

<param name="readAddress">The register address to read</param>

< param name=" addrWidth "> Address width of peripheral IC</param>

<param name="readNumberBytes">Number of bytes to read from the peripheral</param>

<param name="bytesRead">Return data read from the peripheral if successful</param>

```
public bool A2B_SLAVE_PERIREGISTER_READ(string ICName, int masterAddress, int
busAddress, int nodeID, int chipAddress, int readAddress, int addrwidth, int readNumberBytes, ref
byte[] bytesRead)
```

### 3.4.7 A2B\_GET\_NETWORK\_DISCOVERY\_STATUS

<summary>Get A2B network discovery status</summary>

<param name="ICName">Name of A2B IC </param>

<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>

<param name="discStatus">Return status: True if the network if successful else False</param>

<param name="numDiscSlaves">Return number of nodes successfully discovered in the network</param>

```
public bool A2B_GET_NETWORK_DISCOVERY_STATUS(string ICName, int masterAddress, out
bool discStatus, out int numDiscSlaves)
```

### 3.4.8 A2B\_GET\_SLAVE\_NODE\_ACTIVE\_STATUS

<summary>Get slave node discovery status</summary>

<param name="ICName"> Name of A2B IC </param>

<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>

<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>

< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st Slave, 1 for 2<sup>nd</sup> Slave etc </param>

<param name="bActiveStatus"> Node status: True if active, False if inactive </param>

<returns></returns>

```
public bool A2B_GET_SLAVE_NODE_ACTIVE_STATUS(string ICName, int masterAddress, int
busAddress, int nodeID, out bool bActiveStatus)
```

### 3.4.9 A2B\_GET\_NETWORK\_LINEFAULT\_CODE

<summary>Get network line fault status returns 3 parameters: fault code, fault node type and fault node position</summary>

```

<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="faultCode">Fault code from Interrupt Type Register</param>
<param name="faultNodeType">Fault node type : 0-Master, 1-Slave </param>
<param name="faultNodePosition">Fault Node Position : 0 for Master, 0 for 1st Slave, 1 for 2nd
Slave etc </param>
<returns></returns>
public bool A2B_GET_NETWORK_LINEFAULT_CODE(string ICName, int masterAddress, out int
faultCode, out int faultNodeType, out int faultNodePosition)

```

### 3.4.10 A2B\_GET\_NETWORK\_LINEFAULT\_STATUS

```

<summary> Get network line fault status as string, String contains description of fault code, fault
node type and fault node position</summary>
<param name="ICName"> Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="faultDescription">Return fault description that contains the below as string
"faultCode">Fault code from Interrupt Type Register
"faultNodeType">Fault node type : 0-Master, 1-Slave
"faultNodePosition">Fault Node Position: 0 for Master, 0 for 1st Slave, 1 for 2nd Slave etc </param>
<returns></returns>
public bool A2B_GET_NETWORK_LINEFAULT_STATUS(string ICName, int masterAddress, out
string faultDescription)

```

### 3.4.11 A2B\_USER\_COMMAND

```

<summary>Generic command to perform write for A2B</summary>
<param name="ICName">Name of A2B IC </param>
<param name="cmd">Command string</param>
<param name="ParamIn">The data to write, input parameters</param>
<param name="ResOut">Return result out if command execution is successful</param>
<returns></returns>
public bool A2B_USER_COMMAND(string ICName, string cmd, byte[] paramIn, ref byte[] resOut)

```

### 3.4.12 A2B\_START\_BER\_TEST

```

<summary>Start BER Test</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="BERMode">BER Mode: 0-PRBS, 1-Audio; For PRBS, node-to-node checking is
enabled by default; For Audio, all bit errors in BECCTL are enabled by default</param>
public bool A2B_START_BER_TEST(string ICName, int masterAddress, int BERMode)

```

### 3.4.13 A2B\_STOP\_BER\_TEST

```
<summary>Stop BER Test</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
public bool A2B_STOP_BER_TEST(string ICName, int masterAddress)
```

### 3.4.14 A2B\_GET\_BER\_COUNT

```
<summary>Read BER Count</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="BERMode">BER Mode: 0-PRBS, 1-Audio; For PRBS, node-to-node checking is
enabled by default; For Audio, all bit errors in BECCTL are enabled by default</param>
<param name="BERCount"> Returns BER count based on the mode configured if successful. [0]-
Master Node error count, [1] – slave node 0 error count, [2] slave node 1 error count and so on.
</param>
<param name="nNodeCount">Returns the number of active nodes (including Master) for which the
error count is returned in BERCount</param>
public bool A2B_GET_BER_COUNT(string ICName, int masterAddress, int BERMode, out uint[]
BERCount, out int nNodeCount)
```

### 3.4.15 A2B\_RESET\_NETWORK

```
<summary>Issue Soft Reset at Master which will reset whole network</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<returns></returns>
public bool A2B_RESET_NETWORK(string ICName, int masterAddress)
```

### 3.4.16 A2B\_RESET\_SLAVE\_NODE

```
<summary>Issue Soft Reset to an individual slave</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic], 0 for 1st slave
and 1 for 2nd slave etc </param>
public bool A2B_RESET_SLAVE_NODE(string ICName, int masterAddress, int busAddress, int
nodeID)
```

### 3.4.17 A2B\_GET\_NETWORK\_BANDWIDTH\_USAGE

```
<summary>Get Network bandwidth usage</summary>
```

```

<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st
Slave, 1 for 2nd Slave etc </param>
<param name="usage"></param>returns the node bandwidth usage of individual slave. [0]- 'B'
side downstream BW, [1] – 'B' side upstream BW.

<returns></returns>
public bool A2B_GET_NETWORK_BANDWIDTH_USAGE(string ICName, int masterAddress, out
double[] usage)

```

### 3.4.18 A2B\_GET\_NODE\_BANDWIDTH\_USAGE

```

<summary>Get Node bandwidth at individual slave</summary>
<param name="ICName">Name of A2B IC </param>
<param name="masterAddress"> A2B Master Transceiver I2C address (7-bit) </param>
<param name="busAddress"> A2B Bus/Slave Transceiver I2C address (7-bit) </param>
< param name=" nodeID "> Slave Node position in the schematic [Linear schematic]. 0 for 1st
Slave, 1 for 2nd Slave etc </param>
<param name="usage"></param>returns the node bandwidth usage of individual slave. [0] – 'B'
side downstream BW, [1] – 'B' side upstream BW, [2] – 'A' side downstream BW and [3] – 'A' side
upstream BW.
<returns></returns>
public bool A2B_GET_NODE_BANDWIDTH_USAGE(string ICName, int masterAddress, int
busAddress, int nodeID, out double[] usage)

```

### 3.4.19 A2B\_EXPORT\_BUSCONFIG\_C\_FILE

```

<summary>Export API for Bus Configuration C file</summary>
<param name="sBCFCFileName"> Bus configuration File name(*.C) with path</param>
<param name="bcfVersion"> BCF File Version</param>
<param name="optimizeForMemory"> Flag to enabled/disabled memory optimization in BCF
during export</param>
<param name="bcfCompressed"> Flag to enabled/disabled the compressed bcf during export
</param>
<returns></returns>
public bool A2B_EXPORT_BUSCONFIG_C_FILE(string sBCFCFileName, string bcfVersion, bool
optimizeForMemory, bool bcfCompressed)

```

### 3.4.20 A2B\_EXPORT\_BUSCONFIG\_DAT\_FILE

```

<summary>Export API for Bus Configuration DAT file</summary>
<param name="sBCFDatFileName">Bus configuration File(*.DAT) name with path</param>
<param name="streamInfoEnable"> Flag to export the Stream Information to DAT File</param>
<returns></returns>
public bool A2B_EXPORT_BUSCONFIG_DAT_FILE(string sBCFDatFileName, bool
streamInfoEnable)

```

### 3.4.21 A2B\_EXPORT\_BUSCONFIG\_XML\_FILE

```
<summary>Export API for Bus Configuration XML file</summary>
<param name="sBCFXMLFileName">Bus configuration File(*.XML) name with path</param>
<param name="bcfVersion">BCF File Version</param>
<param name="optimizeForMemory">Flag to enabled/disabled memory optimization in BCF during export</param>
<param name="bcfCompressed">Flag to enabled/disabled the compressed bcf during export</param>
<returns></returns>
public bool A2B_EXPORT_BUSCONFIG_XML_FILE(string sBCFXMLFileName, string bcfVersion,
bool optimizeForMemory, bool bcfCompressed)
```

### 3.4.22 A2B\_EXPORT\_COMMANDLIST\_XML\_FILE

```
<summary>Export API for Command List XML file</summary>
<param name="sCmdListXMLFileName">Command List File(*.XML) name with path</param>
<param name="sVersion">Command List Version</param>
<param name="includeModifiedRegister"> Flag to include modified registers only </param>
<param name="includePeriphConfigData"> Flag to include peripheral configuration data</param>
<param name="optimizeForMemory">Flag to enabled/disabled memory optimization in Command List during export</param>
<returns></returns>
public bool A2B_EXPORT_COMMANDLIST_XML_FILE(string sCmdListXMLFileName, string
sVersion, bool includeModifiedRegister, bool includePeriphConfigData, bool optimizeForMemory)
```

### 3.4.23 A2B\_EXPORT\_COMMANDLIST\_H\_FILE

```
<summary>Export API for Command List H file</summary>
<param name="sCmdListHFileName">Command List File(*.H) name with path</param>
<param name="sVersion">Command List Version</param>
<param name="includeModifiedRegister"> Flag to include modified registers only </param>
<param name="includePeriphConfigData"> Flag to include peripheral configuration data</param>
<param name="optimizeForMemory">Flag to enabled/disabled memory optimization in Command List during export</param>
<returns></returns>
public bool A2B_EXPORT_COMMANDLIST_H_FILE(string sCmdListHFileName, string sVersion,
bool includeModifiedRegister, bool includePeriphConfigData, bool optimizeForMemory)
```

### 3.4.24 A2B\_EXPORT\_MAIN\_NODE\_CONFIG\_XML\_FILE

```
<summary>Export API for Main Node Configuration File(NCF.xml)</summary>
<param name="sNodeConfigXMLFileName"> Node Configuration file name(*.xml) with path</param>
<returns></returns>
public bool A2B_EXPORT_MAIN_NODE_CONFIG_XML_FILE(string sNodeConfigXMLFileName)
```

### 3.4.25 A2B\_EXPORT\_SLAVE\_NODE\_CONFIG\_XML\_FILE

```
<summary>Export API for Sub Node Configuration File(NCF.xml)</summary>
<param name="sNodeConfigXMLFileName">Node Configuration file name(*.xml) with
path</param>
<param name="nSlaveNodeId"> Sub node ID</param>
<returns></returns>
public bool A2B_EXPORT_SLAVE_NODE_CONFIG_XML_FILE(string
sNodeConfigXMLFileName, ushort nSlaveNodeId)
```

## 4 Accessing from LabVIEW

To use SigmaStudio as a LabVIEW automation client, both applications must be installed and running on the same machine. In this configuration, LabVIEW is a .NET automation client and SigmaStudio (via SigmaStudioServer) becomes a .NET automation server.

To create a virtual instrument to control SigmaStudio, follow the steps given below:

Step 1. Launch LabVIEW and SigmaStudio on the same machine, and then open a new VI file in LabVIEW.

Step 2. Open the .NET palette to access the .NET objects.

Step 3. Insert a Constructor Node which will open the Select .NET Constructor dialog box.

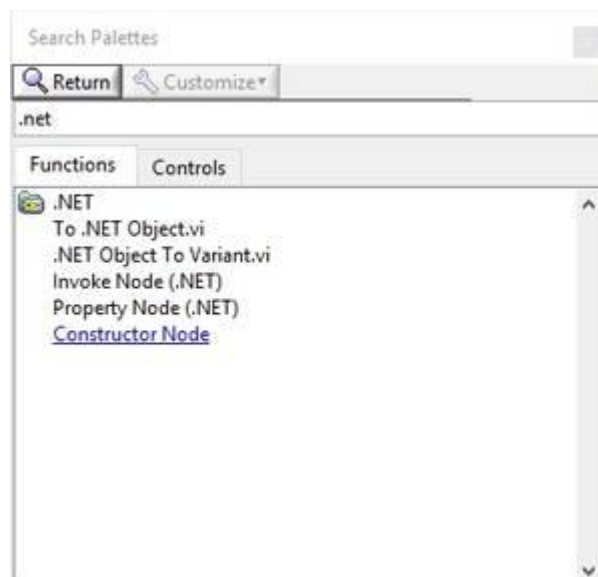
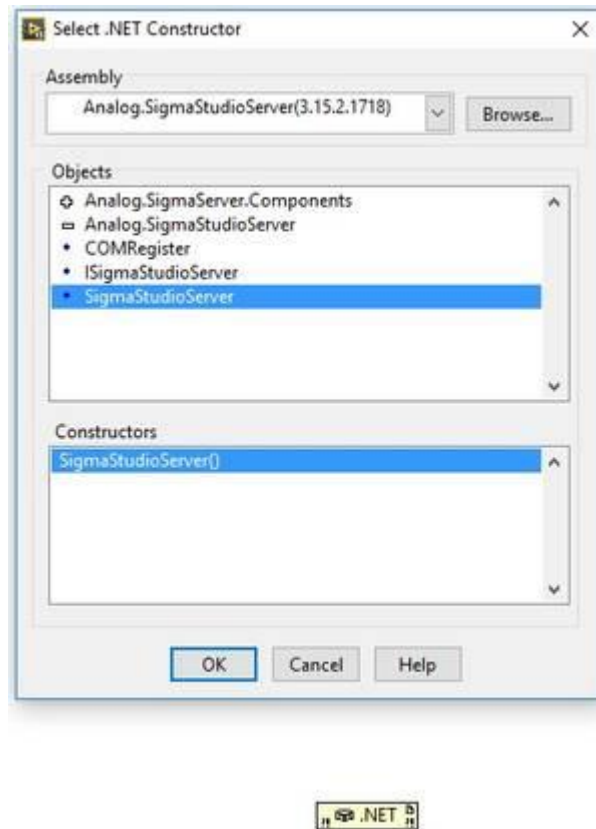


Figure 4: Tool palette in LabVIEW

Step 4. Click the Browse... button.

Step 5. Navigate to the SigmaStudio installation directory, select Analog.SigmaStudioServer.dll

and press OK. [Note :- You can copy the dll to local path if multiple versions of SigmaStudio exists in the machine to override]



**Figure 5: Server selection from .Net constructor**

Step 6. Next choose SigmaStudioServer from the object list, SigmaStudioServer() should be displayed in the constructors list.

Step 7. Click OK to select the SigmaStudioServer constructor.

Step 8. Insert an Invoke Node and connect the constructor Node to it.

9. Click on Method to display and select the accessible SigmaStudioServer commands.



Figure 6: Methods exposed in SigmaServer

Step 10. Example for a single method “Open Project”.

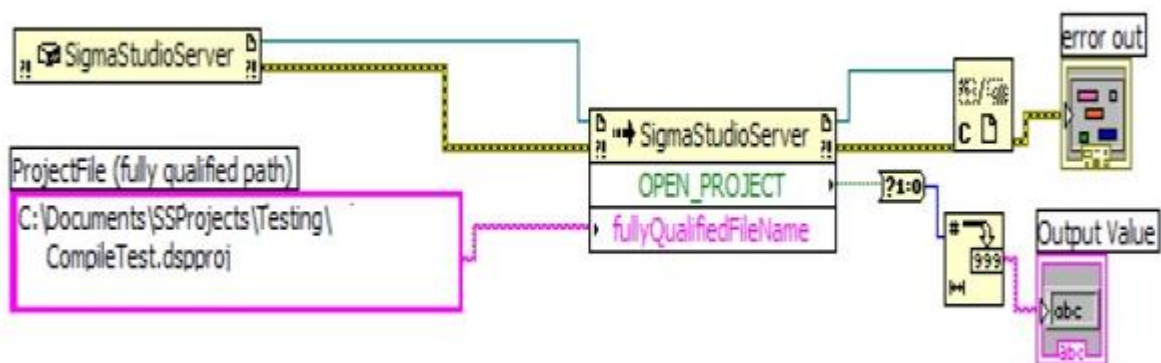


Figure 7: Example for Open Project VI

11. Example VI for sequence of operations “open project” → “Compile” → “Download” then verify the network discovery status and verify the status of each slave.

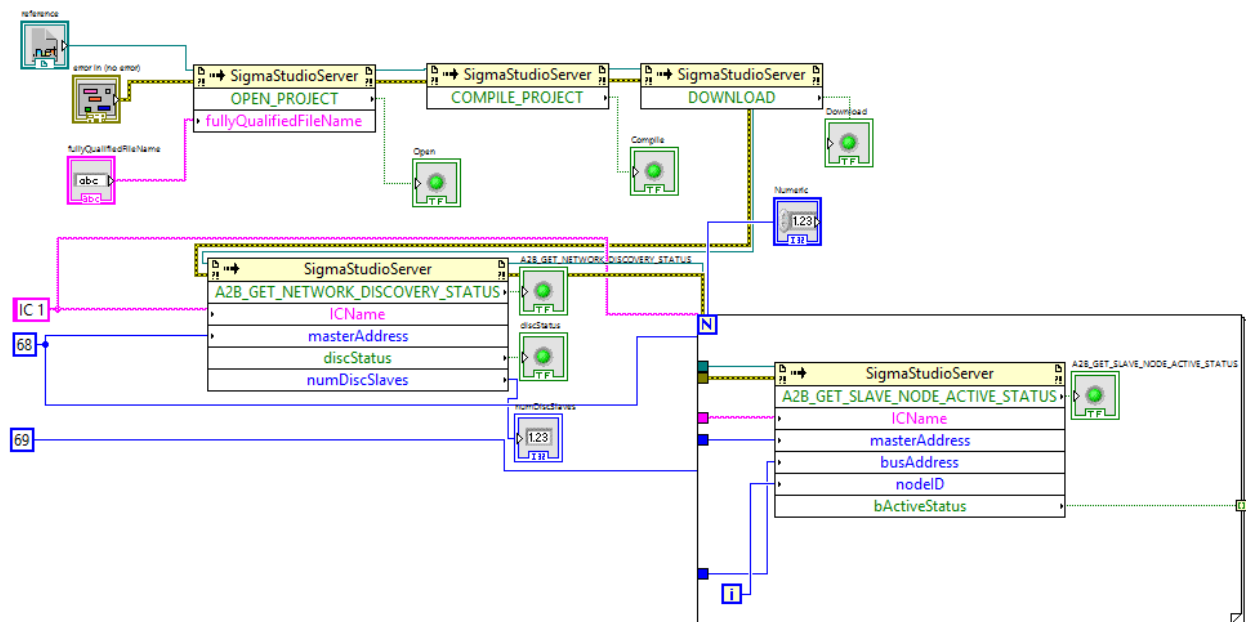


Figure 8: Example of A2B node discovery status check

Step 12. Running SigmaStudio Iscripts from LabView.

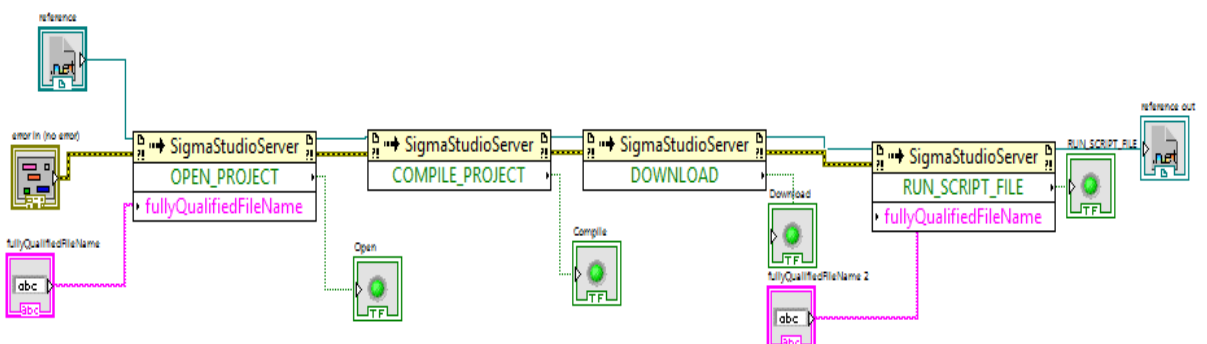


Figure 9: Running Iscript from LabView

## Terminology

**Table 2: Terminology**

Term	Description
API	Application Program Interface
A2B	Automotive Audio Bus
COM	Component Object Model
IC	Integrated Circuit

## References

**Table 3: References**

Reference No.	Description
[1]	<a href="http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/ss_sigst_02.html#dsp-relatedsoftware">http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/ss_sigst_02.html#dsp-relatedsoftware</a>
[2]	<a href="https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting">https://wiki.analog.com/resources/tools-software/sigmastudio/usingsigmastudio/scripting</a>