Title:	Device Interoperability Level 1 Test Descriptions	File Number:	
Security Level:	Public	Release Status:	v 1.7
Owner Group:	RIOLAB	Revision Date:	25 Feb 2007

DEVICE INTEROPERABILITY LEVEL 1 TEST DESCRIPTION

Title:	Device Interoperability Level 1 Test Descriptions	File Number:	
Security Level:	Public	Release Status:	v 1.7
Owner Group:	RIOLAB	Revision Date:	25 Feb 2007

1 REVISION HISTORY

Release Status	Rev. #	Date	Author/ Reviser	Group	Description of Changes
Draft	0.85	4 May 2006	FET	Engineering	Initial Draft.
Draft	0.90	9 May 2006	FET	Engineering	Updated descriptions
Draft	0.96	12 May 2006	FET	Engineering	Updated User inputs/actions and descriptions
First Release	1.0	16 May 2006	FET	Engineering	First Release
Amendment	1.1	17 May 2006	FET	Engineering	Incorporated feedback
Amendment	1.2	18 May 2006	FET	Engineering	Final release
Amendment	1.21	19 May 2006	GR	AE	Update page numbering
Amendment	1.4	21 May 2006	FET	Engineering	Fixed typo in 5.2.2
Update	1.5	31 Jan 2007	FET	Engineering	Update
Update	1.6	22 Feb 2007	FET	Engineering	Update
Update	1.7	25 Feb 2007	FET	Engineering	Update to TID 357 and 360 to address reset requirements.

Title:	Device Interoperability Level 1 Test Descriptions	File Number:	
Security Level:	Public	Release Status:	v 1.7
Owner Group:	RIOLAB	Revision Date:	25 Feb 2007

TABLE OF CONTENTS

2 INTRODUCTION

2.1 Purpose

This document describes Device Interoperability Level 1 (DIL1) tests. This is a living document that will be revised each time the test description is changed. The revision number of this document will form a basis of a formal interoperability report.

DIL1 tests represent the lowest level of interoperability testing. Proof of meeting this level is part of the entry criteria to the lab as testing to higher levels requires this level of fundamental capability.

DIL1 testing will be formally repeated as the first activity in the lab. This level of testing essentially verifies support for initialization, enumeration and basic read and write packet transactions.

As an Interoperability Test Lab, the repeatability of test set-ups, method and sequence is necessary. This document ensures that the test method and sequence is repeatable.

2.2 Related Documents

The following documents were used in the creation of, or are referenced in this document:

- RapidIO™ Interconnect Specification Rev. 1.2, 6/2002
- RapidIO™ Interconnect Specification Part VII: System and Device Inter-operability Specification Rev. 1.2, 06/2002
- RapidIO™ Interconnect Specification Annex 1: Software/System Bring Up Specification Rev. 1.3, 02/2005
- RapidIO™ Interconnect Specification Device Inter-operability and Certification Checklists Rev. 1.0.a, 09/2004

Other related documents:

- DIL1_Test_Setup.doc
- Lab_Hardware_Library.doc
- Command_Interpreter_Description.doc
- DIL1_Test_Scripts.doc

2.3 Terms and Definitions

DUT	Device Under Test. The DUT may be the source of transactions, the target of transactions, and/or a conduit for transactions.
DIL1	Device Interoperability Level 1 testing where the DUT is tested against all other vendor devices in the Hardware Library for both request and response level testing for all combinations of A:B, B:A, A:C, C:A, A:B:C, C:B:A, etc. Focus is on the interaction between devices.

DIL2/DIL3	Device Interoperability Level 2 and 3 testing where the DUT is monitored for proper request and response protocol adherence with a limited set of Qualified endpoints and/or switches. Level 3 quantitatively tests deeper into the protocol than that of 2. Focus is on the behaviour of the DUT.
Hardware Library	RIOLAB library of qualified devices
LA	Logic Analyzer
Qualified Device	A device that has successfully passed DIL1 where it has been directly tested against all other vendor devices in the hardware library.
Qualified Endpoint	A qualified device capable of running scripts or receiving transactions for a device running a script.
Scripts	RapidIO transactions defined using a specific syntax that identifies the transaction type, and the parameters required to fully define it.
Target	A destination for RapidIO transactions issued from the DUT, through the DUT or that represents the DUT.
User Input	Parameters required to be specified by the person conducting the test such as Destination ID, Hop Count, Offset address, number of bytes etc.
User Actions	Interaction required by the person conducting the test such as resetting hardware, Connecting LA probes, Analyzing packet content, etc

3 TEST LEVEL DEFINITION

3.1 Overview

Testing will fall into two basic categories:

- Initiate transactions:
  - For Endpoints this will entail basic I/O reads and writes as well as basic maintenance transactions to a target endpoint.
    - Successful data write transactions will be validated by data read transactions to a known good endpoint.
  - For switches – n/a.
- Respond to transactions:
  - For Endpoints this will entail basic I/O read and write responses as applicable – this does not include specific tests to validate acknowledge packets as these will be implicitly tested.
    - Read responses will be validated by comparing data read from a test target to the data written to the test target.
  - For switches, this will entail propagating packets to and from endpoints appropriately.
  - Maintenance transactions: Each register that is required for enumeration should be verified to ensure that a tested Processing element can support enumeration – registers such as Dest ID, Component Tag, etc.

Note: PE's within the RIOLAB hardware library will be referred to as "qualified" devices.

3.2 Test Setup

Refer to document: DIL1_Test_Setup_1_0.doc.

3.3 Other Processing Elements Tested Against

All of the tests run can be done between multiple devices as shown below in the Devices table.

Refer to document: Lab_Hardware_Library1_0.doc.

3.4 Test Case Descriptions and Checklists

The following RTA Interoperability checklist items are covered within this test:

Title:	Device Interoperability Level 1 Test Descriptions	File Number:	
Security Level:	Public	Release Status:	v 1.7
Owner Group:	RIOLAB	Revision Date:	25 Feb 2007

Test ID +	Item (1.3)	Item (1.2)	Description	Spec Level	Level - Direct	Level - Indirect	Test Method
253.00		3.9.15	A RapidIO device will accept packets of length up to 276 bytes.	Interop	DIL1	-	
357.00		4.2.2.C	Component Tag CSR; reset value is 0x00000000.	Interop	DIL1	-	SCRIPT
360.00		4.2.3.A	Host Base Device ID CSR Host_base_deviceID field reset value is 0xFFFF.	Interop	DIL1	-	SCRIPT
361.00		4.2.3.B	When Host Base Device ID CSR Host_base_deviceID field value is 0xFFFF, the field value can be changed to any value.	Interop	DIL1	-	SCRIPT
362.00		4.2.3.C	When Host Base Device ID CSR Host_base_deviceID field value is not 0xFFFF, the field value will change to 0xFFFF when a value equal to the current field value is written.	Interop	DIL1	-	SCRIPT
363.00		4.2.3.D	When Host Base Device ID CSR Host_base_deviceID field value is not 0xFFFF, the field value does not change when a value not equal to the current field value is written.	Interop	DIL1	-	SCRIPT
364.00		4.2.3.E	When Host Base Device ID CSR Host_base_deviceID field value is 0xFFFF, and 0xFFFF is written to the field, subsequet writes of values not equal to 0xFFFF behave as per 4.2.2.B, 4.2.2.C, and 4.2.2.D.	Interop	DIL1	-	SCRIPT
365.00		4.2.3.F	The Component Tag CSR component_tag field can be written to any value.	Interop	DIL1	-	SCRIPT
412.00		5.3.1.B	MAINTENANCE read request generates a MAINTENANCE read response.		DIL1	-	
417.00		5.3.2.B	MAINTENANCE write request generates a MAINTENANCE write responses.		DIL1	-	
577.00		6.3.1	MAINTENANCE read transaction.	Interop	DIL1	-	SCRIPT

578.00	6.3.1.A	MAINTENANCE read request size of 4 bytes must be supported.		DIL1	-	SCRIPT
579.00	6.3.1.B	MAINTENANCE read request generates a MAINTENANCE read response.		DIL1	-	SCRIPT
582.00	6.3.2	MAINTENANCE write transaction.	Interop	DIL1	-	SCRIPT
583.00	6.3.2.A	MAINTENANCE write request may be for 4 bytes.		DIL1	-	SCRIPT
584.00	6.3.2.B	MAINTENANCE write request generates a MAINTENANCE write response.		DIL1	-	SCRIPT

The following general transaction checklist is used to assess general RapidIO transaction capability:

Speeds / Cycles	**Payload (Bytes)**
Nwrite_R	8
	32
	256
Nwrite	8
	32
	256
Nread	8
	32
	256
Swrite	32
	256

Note: These tests will be performed in 1x and 4x modes if both are supported and at the highest link speed supported.

3.5 User controlled inputs

User controlled inputs are required to control how RapidIO transactions are issued from a given source device to a target device where the device under test may be the source, the target, or a switch or bridge in the data path.

RIOLAB provides test scripts that define the detailed RapidIO transactions issued for any given test. User controlled inputs such as Destination ID, payload size, hop count, etc. are also defined for each of these scripts. The syntax for these scripts are defined in the document titled “Command_Interpreter_Description_0_03.doc”. A simple Script Command Interpreter may be written in support of any processor endpoint to leverage the published scripts. Alternatively, an application called RapidFET™ from Fabric Embedded Tools Corporation may be used to perform various interoperability related tests before or after coming to RIOLAB.

User controlled inputs are defined for each checklist item in section 5.

4 TEST CASES

For DIL1 there are two stages to testing. The first stage involves RTA checklist testing and the second stage involves testing that standard RapidIO transactions are supported.

RTA checklist tests for DIL1 involve discovery related register testing. It requires tests be issued from any qualified endpoint to the DUT.

Standard transaction checklist testing for DIL1 requires tests be issued from a number of qualified endpoints to or through the DUT and visa versa. Please refer to the Device Interoperability Level 1 Test Procedures document for details.

Note: At the DIL1 level, none of the tests will utilize any external equipment such as logic analyzers to validate packet type, content or format. Also, as the tests conducted are RapidIO interoperability tests and not device verification tests, RapidIO in-band verification will be utilized wherever possible.

4.1 RTA Checklist Testing

DIL1 testing for an Endpoint shall involve:

- **Phase A**: Test Scripts shall be run from all of the qualified endpoints, using the DUT as the target for transactions responses – no switch in-circuit. This will prove that the DUT can respond to transactions initiated by all qualified endpoints.
- **Phase B**: The same transactions as performed in Phase A shall be repeated using **all** qualified switches from **one** of the endpoints that successfully passed in Phase A. This will prove that the DUT can successfully respond to transactions through a number of switches. Note: Any 2 ports of the switch may be selected for use in the test and that switches are deemed to have ports identical in function and operation.
- **Phase C:** The same transactions as performed in Phase A shall be run from the Device Under Test (DUT) to **all** of the qualified endpoints or switches. This will prove that the DUT can initiate the appropriate transactions to a number of endpoints.

DIL testing for a switch shall involve:

- **Phase A**: Test scripts shall be run from **all** of the qualified endpoints for maintenance transactions checklist items only, using the DUT as the target. This will prove that the DUT can successfully respond to transactions from a number of endpoints.
- **Phase B**: All Test Scripts shall be run from **one** of the qualified endpoints to **one** of the other qualified endpoints through the DUT. This will prove that DUT can properly pass transactions and responses between devices. Note: any 2 or more ports of a given switch may be used. Reports must specify ports used.

4.1.1 Maximum packet size (ID=253)

Test ID 253: A RapidIO device will accept packets of length up to 276 bytes

Using a qualified endpoint, issue an nWrite packet with a 256 byte payload to a target (DUT). The packet contains known data. The known good endpoint then issues an nRead requesting 256 bytes. The read data is then compared to the written data confirming that the first 276-byte packet was correctly accepted.

User input: Device ID, Hop Count, Target Address

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID253.txt.

4.1.2 Component Tag CSR (ID=357)

Test ID 357: Component Tag CSR; reset value is 0x00000000

Following a reset of the DUT, and using a qualified endpoint, issue a maintenance read to the Component Tag CSR and verify it is equal to the expected value of 0x00000000.

User input: Device ID, Hop Count

User actions: Reset DUT if it has not just been reset. Prompt User.

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID357.txt.

4.1.3 Host Base Device ID CSR (ID=360)

Test ID 360: Host Base Device ID CSR Host_base_deviceID field reset value is 0xFFFF

Following a reset of the DUT, and using a qualified endpoint, issue a maintenance read to the Host Base Device ID CSR on the DUT. The Host Base Device ID CSR Host_base_deviceID field is compared to an expected value of 0xFFFF.

User input: Device ID, Hop Count

User actions: Reset DUT if it has not just been reset. Prompt User.

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID360.txt.

4.1.4 Host Base Device ID CSR (ID=361)

Test ID 361: When Host Base Device ID CSR Host_base_deviceID field value is 0xFFFF, the field value can be changed to any value.

Using a qualified endpoint, issue a maintenance read of the Host Base Device ID CSR on the DUT. Verify that the Device ID CSR Host_base_deviceID field is equal to 0xFFFF. If not, reset the DUT and confirm Host_base_deviceID field is now equal to 0xFFFF. From the qualified endpoint, issue a maintenance write using any value to the Host_base_deviceID field in the Host Base Device ID CSR. Then issue a maintenance read of the same register field and verify that the value written was accepted by comparing the value read to the value written.

User input: Device ID, Hop Count

User actions: Reset DUT - conditional

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID361.txt.

4.1.5 Host Base Device ID CSR (ID=362)

Test ID 362: When Host Base Device ID CSR Host_base_deviceID field value is not 0xFFFF, the field value will change to 0xFFFF when a value equal to the current field value is written.

Using a qualified endpoint, issue a maintenance read to verify that the Host Base Device ID CSR Host_base_deviceID field is not 0xFFFF. If it is 0xFFFF, write any other value to this register using a maintenance write. Verify that by writing the same value back to this register field that it currently holds that the Host_base_deviceID field will return to 0xFFFF.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID362.txt.

4.1.6 Host Base Device ID CSR (ID=363)

Test ID 363: When Host Base Device ID CSR Host_base_deviceID field value is not 0xFFFF, the field value does not change when a value not equal to the current field value is written.

Using a qualified endpoint, issue a maintenance read to verify that the Host Base Device ID CSR Host_base_deviceID field is not 0xFFFF. If it is 0xFFFF, write any other value to this register using a maintenance write. Verify that by writing any other value than the one stored in the Host_base_deviceID field that the register field remains unchanged.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID363.txt.

4.1.7 Host Base Device ID CSR (ID=364)

Test ID 364: When Host Base Device ID CSR Host_base_deviceID field value is 0xFFFF, and 0xFFFF is written to the field, subsequet writes of values not equal to 0xFFFF behave as per 4.2.2.B, 4.2.2.C, and 4.2.2.D.

Using a qualified endpoint, issue a maintenance Read and determine what the content is in the Host Base Device ID CSR Host_base_deviceID field. If it is not 0xFFFF write the contents of the register field back to the register field and verify that the value of the register field returns to 0xFFFF. Write a value of 0xFFFF to the register field followed by a write of any other value. Verify that the value of the register field changes. Verify that by writing the same value back to the register field that the value returns to 0xFFFF. Then write a non-0xFFFF value to the register field and verify that writing any other value than the one stored in the register field back to the register does not change it's contents.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID364.txt.

4.1.8 Component Tag CSR (ID=365)

Test ID 365: The Component Tag CSR component_tag field can be written to any value (ID=365)

Using a qualified endpoint, issue a maintenance write to the Component Tag CSR component_tag field with any value. Confirm that this value has been accepted using a maintenance read. Repeat this process with at least 8 other values including a minimum value of 0x0 and a maximum value of 0xFFFFFFFF.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID365.txt.

4.1.9 MAINTENANCE read (ID=412)

Test ID 412: MAINTENANCE read request generates a MAINTENANCE read response.

Using a qualified endpoint, issue a maintenance write to the Component Tag CSR. Read the data back and verify that the data read is the same value as the data written in the previous step

User input: Device ID, Hop Count

User actions: none

Note: This is an indirect test that an appropriate maintenance read response has been issued.

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID412.txt.

4.1.10 MAINTENANCE write (ID=417)

Test ID 417: MAINTENANCE write request generates a MAINTENANCE write responses.

Using a qualified endpoint, issue a maintenance write to the Component TAG register. Issue a maintenance read of the same register and verify that the contents holds the expected value.

User input: Device ID, Hop Count

User actions: none

Note: This is an indirect test that an appropriate maintenance write response has been issued.

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID417.txt.

4.1.11 MAINTENANCE read (ID=577)

Test ID 577: MAINTENANCE read transaction

Note: This is a checklist heading. It is included here for completeness and does not represent a checklist test in itself.

4.1.12 MAINTENANCE read (ID=578)

Test ID 578: MAINTENANCE read request size of 4 bytes must be supported.

Using a qualified endpoint, issue a maintenance write of 0x12345678 to the Component TAG register. Issue a maintenance read to the Component TAG register and confirm that the 4 byte value of 0x12345678 is received.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID578.txt.

4.1.13 MAINTENANCE read (ID=579)

Test ID 579: MAINTENANCE read request generates a MAINTENANCE read response.

Using a qualified endpoint, issue a maintenance write to the Component TAG register. Issue a maintenance read of the same register and verify that the contents holds the expected value.

User input: Device ID, Hop Count

User actions: none

Note: This is an indirect test that an appropriate maintenance write response has been issued. It is the same test as 5.1.10

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID579.txt.

4.1.14 MAINTENANCE write (ID=582)

Test ID 582: MAINTENANCE write transaction (ID=582)

Note: This is a checklist heading. It is included here for completeness and does not represent a checklist test in itself.

4.1.15 MAINTENANCE write (ID=583)

Test ID 583: MAINTENANCE write request may be for 4 bytes.

Using a qualified endpoint, issue a maintenance write to the Component TAG register. Issue a maintenance read to the same register and confirm that the value read is the same 4 bytes that was written.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID583.txt.

4.1.16 MAINTENANCE write (ID=584)

Test ID 584: MAINTENANCE write request generates a MAINTENANCE write response.

Using a qualified endpoint, issue a maintenance write to the Component TAG register. Issue a maintenance read of the same register and verify that the contents holds the expected value.

User input: Device ID, Hop Count

User actions: none

Note: This is an indirect test that an appropriate maintenance write response has been issued. It is the same test as 5.1.10 and 5.1.13

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID584.txt.

4.2 General Transaction Checklist Testing

These tests will be conducted for different payload sizes with the DUT issuing the transaction as well as being the target of the transaction. Qualified devices will be used within this test as specified in the Device Interoperability Level 1 Test Procedures document.

In general, validation of write transactions will be done by reading back data and comparing it with known values. Similarly, validation of read transactions will be done by first writing data and then comparing the read data with expected values. Successful completion of such a test requires both read and write transactions to be successful. Failure of either type of transaction will result in failing both read and write related tests.

Given Interoperability testing is not intended to replace design verification nor is it required to provide conclusive route cause analysis, this approach is taken over other independent or out of band validation techniques as it simplifies an already complicated test procedure and allows for a greater level of automation.

As successful DIL1 testing is an entry criterion for RIOLAB, it is assumed that devices will pass all of these fundamental transactions. Higher levels of DIL and SCL testing will rely on successful basic read, write and maintenance transactions as valid in band mechanisms for validating tests through scripts.

DIL testing for an Endpoint shall involve:

- **Phase A**: a set of Test Scripts shall be run from the Device Under Test (DUT) to **all** qualified endpoints – no switch. This will prove that the DUT can initiate the appropriate transactions to a number of endpoints.
- **Phase B**: This test shall be repeated using **all** qualified switches to **one** of the endpoints that successfully passed in Phase A. This will prove that the DUT can successfully initiate transactions through a number of switches. Note: Any 2 ports of the switch may be selected for use in the test and that switches are deemed to have ports identical in function and operation
- **Phase C**: All Test Scripts shall then be run from **one or more** qualified endpoints, using the DUT as the target for transactions responses – no switch. This will prove that the DUT can respond to transaction initiated by all qualified endpoints.

DIL testing for a switch shall involve:

- **Not applicable**

4.2.1 nWrite with Response

The following tests will be conducted using 8, 32, and 256 byte payloads. The DUT will be tested to ensure it can issue and respond to the transaction specified.

Using a payload of 0xFFFFFFFF word values write 512 bytes of data to a memory location. Using a data payload comprised of 0x55555555 word values, issue an nWrite with Response transaction to an appropriate memory location in the DUT (if the DUT is an endpoint) or through the DUT (if the DUT is a switch). Then issue an nRead transaction with a larger payload payload size to the same memory location. Verify that the data received was the same data written and that correct number of bytes of data was written.

Repeat the above transaction with a data payload comprised of 0xAAAAAAAA.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID nWrite_R.txt.

4.2.2 nWrite

The following tests will be conducted using 8, 32, and 256 byte payloads. The DUT will be tested to ensure it can issue and respond to the transaction specified.

Using a payload of 0xFFFFFFFF word values write 512 bytes of data to a memory location. Using a data payload comprised of 0x55555555 word values, issue an nWrite transaction to an appropriate memory location in the DUT (if the DUT is an endpoint) or through the DUT (if the DUT is a switch). Then issue an nRead transaction with a larger payload payload size to the same memory location. Verify that the data received was the same data written and that correct number of bytes of data was written.

Repeat the above transaction with a data payload comprised of 0xAAAAAAAA.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID nWrite.txt.

4.2.3 nRead

The following tests will be conducted using 8, 32, and 256 byte payloads. The DUT will be tested to ensure it can issue and respond to the transaction specified.

Using a data payload comprised of 0x55555555 word values, issue an nWrite with Response transaction to an appropriate memory location in the DUT (if the DUT is an endpoint) or through the DUT (if the DUT is a switch). Then using an nRead transaction with equivalent payload size to the same memory location, read the data back and compare it to what was written.

Repeat the above transaction with a data payload comprised of 0xAAAAAAAA.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID nRead.txt.

4.2.4 sWrite

The following tests will be conducted using 32, and 256 byte payloads. The DUT will be tested to ensure it can issue and respond to the transaction specified.

Using a payload of 0xFFFFFFFF word values write 512 bytes of data to a memory location. Using a data payload comprised of 0x55555555 word values, issue an sWrite transaction to an appropriate memory location in the DUT (if the DUT is an endpoint) or through the DUT (if the DUT is a switch). Then issue an nRead transaction with a larger payload payload size to the same memory location. Verify that the data received was the same data written and that correct number of bytes of data was written.

Repeat the above transaction with a data payload comprised of 0xAAAAAAAA.

User input: Device ID, Hop Count

User actions: none

The scripts that are representative of the RapidIO transactions that reproduce this test can be found in document DIL1 Test Scripts ID sWrite.txt.