SCSI TOOLBOX, LLC Using SAT to access SATA drives

Contents

What is SAT?	3
How do I know that I need to use SAT?	4
Does my controller card support SAT?	5
A Simple test command	6
Issuing the ATA SMART command	8
Details of defining an IDENTIFY command	10
Details of defining a SMART command	13
Conclusion	13

What is SAT?

SAT (SCSI->ATA Translation) is a mechanism whereby ATA task register commands may be sent to a device which is seen by the operating system as a SCSI device. This is most often the case when SATA drives are connected to an add-in PCI bus type of SATA controller card. Even though the card is a SATA controller in most cases Windows will see the controller as if it were a SCSI HBA, and so will not allow you to issue ATA task register level commands to the connected devices.

Documentation on SAT can be found at the T10.org site http://www.t10.org/drafts.htm

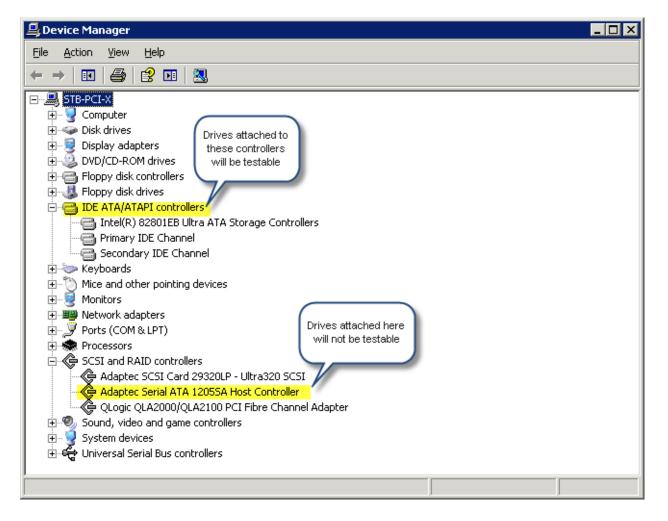
In short, SAT uses a 12 or 16 byte SCSI cdb which contains an embedded ATA task register command.

How do I know that I need to use SAT?

If your controller is seen by Windows as an ATA/IDE type of controller than you do not need to use SAT, you can simply issue normal ATA task register commands. How do you know what type of controller you have?

You can confirm how your operating system views your controller scheme by using Device Manager as shown below – note that the only drives that will be able to process actual ATA task register commands *must* be attached to a controller that Windows sees as an *IDE ATA/ATAPI controller*

In the example below, the second controller (Adaptec Serial ATA 1205A Host Controller) *might* be able to use SAT to send and ATA command to an attached drive.



Does my controller card support SAT?

The easiest way to determine of your controller supports SAT is to use the STB Suite SCSI User Defined CDB to try issuing a SAT command.

Byte\Bit	7	6	5	4	3	2	1	0				
0		OPERATION CODE (A1h)										
1	MU	LTIPLE_COU	NT		PRO	TOCOL		Reserved				
2	OFF_	LINE	CK_COND	Reserved	T_DIR	BYTE_BLOCK	T_LE	NGTH				
3				FEATUR	RES (7:0)							
4		SECTOR_COUNT (7:0)										
5	lba_low (7:0)											
6	LBA_MID (7:0)											
7		LBA_HIGH (7:0)										
8		DEVICE										
9		COMMAND										
10		Reserved										
11												

The 12-byte ATA Passthrough CDB we will use is defined as:

There are some obscure aspects of using this command – rather than going into detail about them right now we will instead simply describe how to issue a command which will tell us immediately if the controller supports SAT or not. We will discuss the details of each parameter of this command later on in this article.

A Simple test command

Here is a picture of the STB Suite User Defined CDB with an ATA Pass-through command defined which will issue an ATA IDENTIFY command to the attached drive

SCSI Commands ATA-SAT-IDENTIFY 12.1, 200 A1 0C 0D 01 00 00 00 00 EC 00 00 01 1 00 00 4.3] ATA-SAT-SMART,12,1, 200 A1,0C,0E,D0,01,00,4F,C2,00,B0,00,00,01,1,00,00,4.3] Compare,10,0,00,39,00,00,00,00,00,00,00,00,00,00,00,00 Compare,10,0,00,39,00,00,00,00,00,00,00,00,00,00,00,00,00	User Defined CDB	x
ATA-SAT-SMART,12,1, 200,A1,0C,0E,D0,01,00,4F,C2,00,B0,00,00,01,1,00,00,4,3] Change Definition,10,0,00,40,00,00,00,00,00,00,00,00,00,00,		
Change Definition, 10,00,040,00,00,00,00,00,00,00,00,00,00,0		
Compare 10,0,00,39,00,00,00,00,00,00,00,00,00,00,00,00,00		
Extended Read,10,1,200,28,00,00,00,00,00,00,00,00 Evtended Seek 10.0.0.2b 00.00.00.00.00.00.00.00 Add CDB Delete CDB Load CDB File Save CDB File Add CDB Delete CDB 0 1 2 3 4 5 6 7 8 9 10 11 A1 0C 0D 01 00 <	Compare,10,0,00,39,00,00,00,00,00,00,00,00	
Evended Seek 10 0 0 2b 00 00 00 00 00 00 00 00 00 00 Load CDB File Save CDB File Add CDB Delete CDB 0 1 2 3 4 5 6 7 8 9 10 11 A1 0C 0D 01 00 00 00 00 00 00 CDB Name = ATA-SAT-IDENTIFY Data In • Out • Increment LBA Repeat 1 CDB Length : • 0 at in • Out • • Inc . by 00 00 00 00 • 10 Bytes • 512 • IBA MSB 3 • • • • • 12 Bytes • <	Copy and Verify,10,0,ff,3a,00,00,00,00,00,00,00,00,00	
0 1 2 3 4 5 6 7 8 9 10 11 A1 0C 0D 01 00 00 00 00 00 00 00 CDB Name = ATA-SAT-IDENTIFY	Extended Read, 10, 1,200,28,00,00,00,00,00,00,00,00,00,00	
0 1 2 3 4 5 6 7 8 9 10 11 A1 0C 0D 01 00 00 00 00 00 00 00 CDB Name = ATA-SAT-IDENTIFY		
A1 0C 0D 01 00 00 00 00 00 00 CDB Name = ATA-SAT-IDENTIFY CDB Length :	Load CDB File Save CDB File Add CDB Delete CDB	
CDB Name = ATA-SAT-IDENTIFY CDB Length : O 6 Bytes O 10 Bytes O 12 Bytes Data In O Ut O Inc. by 00 LBA MSB 3 O Stop on error	0 1 2 3 4 5 6 7 8 9 10 11	
CDB Length : Data In • Out • • 6 Bytes Transfer Length = • 10 Bytes 512 • 12 Bytes 512	A1 0C 0D 01 00 00 00 00 EC 00 00	
CDB Length : Data In • Out • • 6 Bytes Transfer Length = • 10 Bytes 512 • 12 Bytes 512		
O 6 Bytes Transfer Length = Inc . by 00 Timeout 0 O 10 Bytes 512 LBA MSB 3 O Stop on error	CDB Name = ATA-SAT-IDENTIFY	
© 6 Bytes Transfer Length = Inc. by 00 Timeout © 10 Bytes 512 LBA MSB 3	CDB Length : Data In Out Data In CDB Length : Data In Out Data In Out Data In Out Out Out Out Out Out Out Out	
O 10 Bytes Fransfer Length = LBA MSB 3 O Stop on error	C 6 Butes	
12 Bytes 512 LBA MSB IS O Stop on error	C 10 Bytes Transfer Length = T	
	12 Bytes 512 LBA MSB 5	
C 16 Bytes Dunier 19 2 5	O 16 Bytes Buffer 1 ⊙ 2 O LBA LSB 4	
CDB Result =	CDB Result =	
Return Send CDB View Results Buffers Stop CDB	Return Send CDB View Results Buffers Stop CDB	

Select your target drive on the SATA controller that you hope will support SAT. Then right-click on the drive and choose *User Defined CDBs*, then enter the command exactly as show above.

Click the **Send CDB** button to issue the command – the CDB Result field will tell if the command was successfully issued (congratulations, your controller implements SAT!) or if it failed (sorry, you won't be able to use SAT with this controller)

dit Buffer																	×
																	Size 1048576 (1024Kb)
	00	01	02	03	04	05	06	07	08	09	ØA	ØB	ØC	ØD	ØE	ØF	<u> </u>
000000	78	42	FF	3F	37	C8	10	00	00	00	00	00	3F	00	00	00	zB.?7?
000010	00	00	00	00	20	20	20	20	57	20	2D	44	4D	57	4D	41	W -DMWMA
000020	39	39	30	33	34	35	35	32	00	00	00	40	41	00	30	31	99034552@A.01
000030	30	2E	45	31	31	30	44	57	20	43	44	57	30	38	4A	30	0.E110DW CDW08J0
000040	2D	44	35	37	53	4D	31	41	20	20	20	20	20	20	20	20	-D57SM1A
000050	20	20	20	20	20	20	20	20	20	20	20	20	20	20	10	80	
000060	00	00	00	2F	01	40	00	00	00	00	07	00	FF	3F	10	00	/.@?
000070	3F	00	10	FC	FB	00	00	01	90	2F	50	09	00	00	07	00	?/P
			-											des - co			▼
File Operat	inn	5							Fi	11	Buł	Ffe	rl	A1	1	Zer	05 (0)
		_												1			
Buffer 1		Bu	Ffe	er :	2												
Return																	Cancel

If the command completed successfully you can click the *Buffer* button so view the ATA IDENTIFY data returned from the drive.

Note: ATA commands data byte-swapped.

Issuing the ATA SMART command

To retrieve SMART data from this drive define your CDB like this:

Jser Defined CDB			
CSI Commands			
		00,00,00,EC,00,00,01,1,00,00,4	
		,C2,00,B0,00,00,01,1,00,00,4,3	
),0,00,40,00,00,00,00,00,00,00,00, ,00,00,00,00,	,00,00	
	,ff,3a,00,00,00,00,00,00,00,00,00,00,	00	
	,200,28,00,00,00,00,00,00,00,00,0		
Extended Seek 10.0	0.26.00.00.00.00.00.00.00.00	00	
<u>. </u>	2		
Load CDB File	Save CDB File	Add CDB	Delete CDB
0 1 2	3 4 5 6 7	8 9 10 11	
A1 OC OE	D0 01 00 4F C2	00 80 00 00	
CDB Name = ATA	-SAT-SMART		
CDB Length :	Data In 💿 Out 🔿	Increment LBA	Repeat 1
C 6 Bytes		Inc. by 00	
C 10 Bytes	Transfer Length =		Timeout 0
12 Bytes	512	LBA MSB 3	C 01
C 16 Bytes	Buffer 1 • 2 O	LBA LSB 4	C Stop on error
io bytes			
CDB Result = Statu	is Good - Command completed	without error	
Return	Send CDB View F	Results Buffers	Stop CDB
	Lances and in the second se		

Edit Buffer																		x
																3	Size 1048576 (1024Kb)	
	00	01	02	03	04	05	06	07	08	09	ØA	ØB	ØC	ØD	ØE	ØF	·	
000000	10	00	01	ØF	00	C8	C8	00	00	00	00	00	00	00	03	03		
000010	00	A6	AЗ	8C	ØA	00	00	00	00	00	64	32	00	64	64	46	2.ddF	
000020	00	00	00	00	00	00	05	33	00	C8	C 8	00	00	00	00	00		
000030	00	00	07	ØF	00	C 8	C8	00	00	00	00	00	00	00	09	32	2	
000040	00	63	63	EØ	03	00	00	00	00	00	ØA	13	00	64	FD	00	.ccd	
000050	00	00	00	00	00	00	ØB	12	00	64	FD	00	00	00	00	00	d	
000060	00	00	ØC	32	00	64	64	3E	00	00	00	00	00	00	BE	22	2.dd>"	
000070	1.11			1.1		1.1		199		1					10.0		.I;".tf.	
000010	00	**			00	00	00	00	00	00	02		00	• •	00		<u> </u>	
	•	1							-									
File Operat	100	S						-	F1.		BUI	Ffe	r	нт	. 1	zer	'os (0) 🔽	
Buffer 1		Bui	Ffe	r :	2													
Return																	Canc	el

And as before, the data is available to view, edit, or save to a file

Details of defining an IDENTIFY command

Byte\Bit	7	6	5	4	3	2	1	0				
0		OPERATION CODE (A1h)										
1	MU	ILTIPLE_COU	NT		PRO	TOCOL		Reserved				
2	OFF_	LINE	CK_COND	Reserved	T_DIR	BYTE_BLOCK	T_LE	NGTH				
3				FEATUR	RES (7:0)							
4		SECTOR_COUNT (7:0)										
5	LBA_LOW (7:0)											
6	LBA_MID (7:0)											
7				LBA_HI	GH (7:0)							
8				DE	VICE							
9		COMMAND										
10		Reserved										
11												

The bytes of the command we constructed to issue an IDENTIFY command were:

0xA1, 0x0C, 0x0D, 00,00,00,00,00,0xEC,00,00

Byte 0:

Looking at the command description we see that the first byte is the SCSI op code.

Byte 1:

the next byte is used to specify the protocol, as defined in this table:

Code	Description
0	ATA hardware reset
1	SRST
2	Reserved
3	Non-data
4	PIO Data-In
5	PIO Data-Out
6	DMA
7	DMA Queued
8	Device Diagnostic
9	DEVICE RESET
10	UDMA Data In
11	UDMA Data Out
12	FPDMA ^a
13, 14	Reserved
15	Return Response Information
^a See SA	TA-2.6.

In our command the 0x0C says we are requesting a DMA transfer. Note that you need to take care when defining this byte because of the offset caused by bit 0 being reserved.

Byte 2:

This byte is used to define how much data we are expecting, how the amount is specified, and which further bytes of the command will be used to specify. In the case of our IDENTIFY command we use 0x0D, which specifies that T_DIR = 1, BYTE_BLOCK=1, and T_LENGTH = 1. Referring to the SAT specification we see:

If the τ_{DIR} bit is set to zero, then the SATL shall transfer data from the application client to the ATA device. If the τ_{DIR} bit is set to one, then the SATL shall transfer data from the ATA device to the application client. The SATL shall ignore the τ_{DIR} bit if the τ_{LENGTH} field is set to zero.

T_DIR = 1 says that the data direction will be receiving data from the drive.

The BYTE_BLOCK (Byte/Block) bit specifies whether the transfer length in the location specified by the T_LENGTH field specifies the number of bytes to transfer or the number of blocks to transfer. If the value in the BYTE_BLOCK bit is set to zero, then the SATL shall transfer the number of bytes specified in the location specified by the T_LENGTH field. If the value in the BYTE_BLOCK bit is set to one the SATL shall transfer the number of blocks specified in the location specified by the T_LENGTH field. If the value in the BYTE_BLOCK bit is set to one the SATL shall transfer the number of blocks specified in the location specified by the T_LENGTH field. The SATL shall ignore the BYTE_BLOCK bit when the T_LENGTH field is set to zero.

BYTE_BLOCK=1 says that we are expecting to transfer one block (512 bytes) of data.

The Transfer Length (T_LENGTH) field specifies where in the CDB the SATL shall locate the transfer length for the command (see table 98).

Code	Description
00b	No data is transferred
01b	The transfer length is an unsigned integer specified in the FEATURES (7:0) field.
10b	The transfer length is an unsigned integer specified in the SECTOR_COUNT (7:0) field.
11b	The transfer length is an unsigned integer specified in the TPSIU (see 3.1.98).

Table 98 — T_LENGTH field

Finall

y, T_LENGTH = 01 tells us that our transfer length is going to be specified by the integer placed in the ATA FEATURES byte field – which in this case is byte 3 of the CDB.

Byte 3:

-as we just said – because of our T_LENGTH setting we have specified that this byte will contain the number of blocks (because of the BYTE_BLOCK setting) of data that will be transferred, in the direction specified by T_DIR.

Byte 4:

- contains the ATA task register SECTOR COUNT data, in this case we are transferring 1 block so this must be set to 1.

Byte 5:

- contains the ATA task register LBA LOW byte – the IDENTIFY command needs this to be 0.

Byte 6:

- contains the ATA task register LBA MID byte – the IDENTIFY command needs this to be 0.

Byte 7:

- contains the ATA task register LBA HIGH byte – the IDENTIFY command needs this to be 0.

Byte 8:

- contains the ATA task register DEVICE byte – the IDENTIFY command needs this to be 0.

Byte 9:

- contains the ATA task register COMMAND byte – the IDENTIFY command is 0xEC.

Bytes 10 and 11 :

-are reserved and so set to 0.

Details of defining a SMART command

Refer to the T10 SAT specification documentation.

Note that we specify T_LENGTH = 10, which uses the SECTOR COUNT field (Byte 4) to specify our data length. Why did we need to do this, rather than use the FEATURES field (Byte 3) like we did for the IDENTIFY command?

We had to do this because the ATA SMART command needs to use the FEATURES field to define which type of SMART command we are sending – 0xD0 in this case.

Conclusion

SAT is a versatile if complicated method of issuing ATA commands to drives which are connected to controllers which Windows thinks are SCSI type. It is preferable rather than being forced to implementing controller vendor-unique pass through methods.

SAT is not universally supported or implemented. The controller that was used to illustrate this article is an LSI 8888 card, which happily does implement SAT and so allows access to "raw" ATA commands which would otherwise not be usable.