Written by freevanx Friday, 05 November 2010 02:51 -

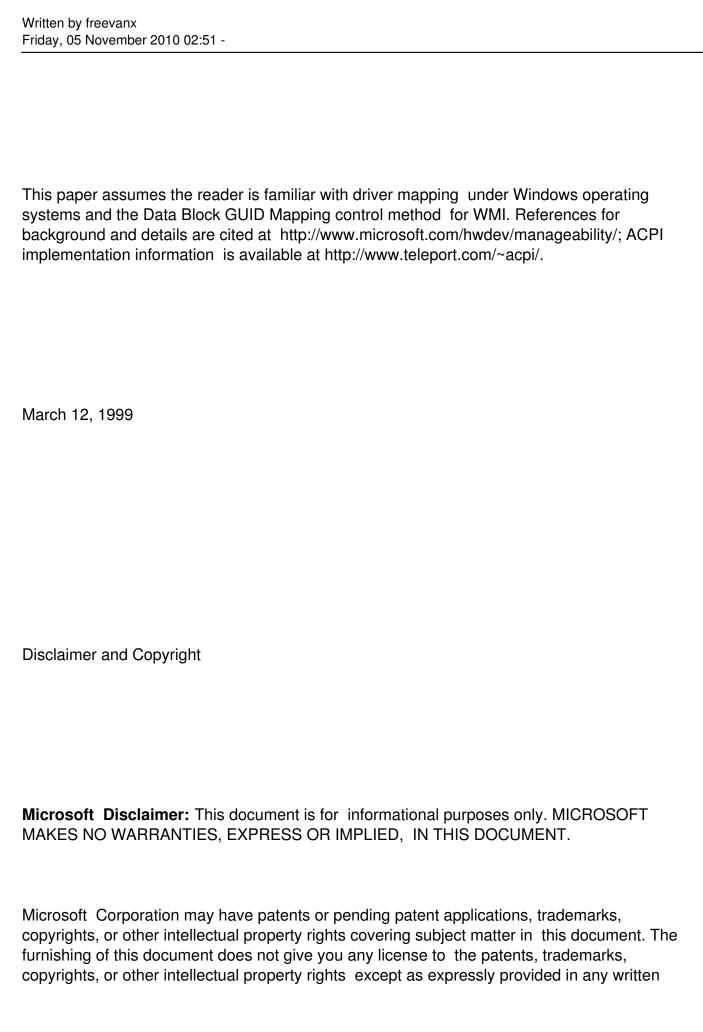
WinH	EC 99	White	Paper
------	-------	-------	-------

Abstract

Windows® Hardware Engineering Conference: Advancing the Platform
Windows 2000 Instrumentation: WMI and ACPI
Williams 2000 instrumentation. Wivii and AOFT

This paper describes the process that system manufacturers (OEMs) can use to provide instrumentation information by including ACPI objects in the systems they build that will be recognized by Microsoft® Windows® Management Instrumentation (WMI). This information applies for Windows 2000 and Windows 98 OSR 1. By including ACPI objects in the systems they build, OEMs can take advantage of a generic mapping driver that allows WMI to make the information available to the instrumentation consumers.

The ACPI subsystem contains a wealth of instrumentation information; OEMs are encouraged to use ACPI to add additional platform specific instrumentation information. However, ACPI objects are not readily accessible by instrumentation data consumers such as Web-Based Enterprise Management (WBEM).



Written by freevanx Friday, 05 November 2010 02:51 -

license agreement from Microsoft Corporation.

Microsoft does not make any representation or warranty regarding specifications in this document or any product or item developed based on these specifications. Microsoft disclaims all express and implied warranties, including but not limited to the implied warranties or merchantability, fitness for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Microsoft does not make any warranty of any kind that any item developed based on these specifications, or any portion of a specification, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is your responsibility to seek licenses for such intellectual property rights where appropriate. Microsoft shall not be liable for any damages arising out of or in connection with the use of these specifications, including liability for lost profit, business interruption, or any other damages whatsoever. Some states do not allow the exclusion or limitation of liability or consequential or incidental damages; the above limitation may not apply to you.

Microsoft, DirectX, MSI DOS, Win32, Windows, and Windows NT are registered trademarks of Microsoft Corporation. Other product and company names mentioned herein may be the trademarks of their respective owners.

© 1999 Microsoft Corporation. All rights reserved.

Contents

Written by freevanx

Friday, 05 November 2010 02:51 -WMI Overview for Windows 2000 ACPI-to-WMI Mapper Goals for Windows 2000 ACPI Control Method Naming Conventions and Functionality for Windows 2000 Instrumentation..... 5 Data Block Design for Windows 2000 FAQ about WMI and ACPI for Windows 2000 ASL Methods and Sample ASL Sample Code for an Event and Initiating Sample ASL Code Embedding MOD Data in Sample .Mof File..... 17

Written by freevanx Friday, 05 November 2010 02:51 -

Technology Overview for Windows 2000 Instrumentation

The ACPI-to-WMI mapping functionality is achieved by means of two device drivers provided with the Windows 2000 and Windows 98 OSR1 operating systems:

- Acpi.sys is the regular ACPI device driver with some modifications.
- Wmiacpi.sys registers with Plug and Play ID PNP0c14.

OEMs can differentiate their PC system capabilities by writing ACPI Source Language (ASL) code and a Managed Object Format (.mof or MOF) file. The .mof file can be in the BIOS or on disk. For more information about MOF, see the "MOF Data Types" section later in this paper.

ASL code is never executed directly by the Wmiacpi.sys driver. ASL code is always executed by the Acpi.sys driver (see the ASL information at http://www.teleport.com/~acpi/). Wmiacpi.sys will invoke Acpi.sys to call control methods that access the management data exposed by the mapping driver.

Microsoft does not ship a .mof file that is associated with the Wmiacpi.sys driver. The only

Written by freevanx Friday, 05 November 2010 02:51 -

information surfaced through ACPI is the temperature zone information, which is surfaced through and associated with the Acpi.sys device driver.

WMI Overview for Windows 2000 Instrumentation

WMI organizes individual data items (properties) into data blocks (structures) that contain related information. Data blocks may have one or more data items. Each data item has a unique index within the data block, and each data block is named by a globally unique 128-bit number called a globally unique identifier (GUID). WMI can provide notifications to the data producer as to when to start and stop collecting the data items that compose a data block. WMI has no knowledge of the data format for individual data blocks.

WMI functionality allows for querying all instances of a data block or a single instance of a data block. It also allows for setting all data items in an instance of a data block or a single data item within a single instance of a data block.

In addition to queries and sets, WMI allows WMI method calls, which are functionally equivalent to an I/O control (IOCTL) call to a device. Each WMI method call is identified by a GUID and a method index for that GUID. All WMI method calls use one buffer for input and output parameters. WMI allows notifications of significant events to be delivered to interested user-mode applications. Each type of event is uniquely named by a GUID. Events may also carry a data block with additional information about the event. WMI can provide notifications to the event generator about when to enable and disable an event type.

Written by freevanx Friday, 05 November 2010 02:51 -

WMI is an open architecture that allows OEMs to define their own data blocks, methods, and events. Along with the data that composes the custom data block, the OEM must also provide a description that generally represents how a data block or WMI method is mapped to a 2-character ID. This 2\(\text{D}\) character ID is part of the names of the control methods that act upon the data block.

For example, when a call is made to query about the data block represented by a WMI GUID, the mapper will evaluate the WQxx control method (where xx is the 2-character ID mapped to that GUID). These mappings are defined by the ACPI code and obtained by the mapper evaluating the _WDG control method. For more information, see "ACPI Control Method Naming Conventions and Functionality" later in this paper.

The mapping process is similar for events. The _WDG control method provides a mapping between the WMI event GUID that represents the event and the notification code specified in the ASL notify instruction. For example, when ACPI provides a callback to the mapper that a control method executed a **notify(mapper-device, 0x81)** function, the mapper will look up the WMI GUID mapped to 0x81 and use this WMI GUID in building the WMI event. Before launching the WMI event, the mapper will evaluate _WED to retrieve any additional data that belongs with the event.

Loading the Mapping Driver. The Plug and Play ID PNP0c14 is assigned as the WMI-mapping pseudo device; the operating system device INFs (Plug and Play ID-to-device driver lookup table) point this Plug and Play ID to the ACPI-to-WMI mapping driver. To cause the ACPI-to-WMI mapping driver to load, an ACPI system needs to define one or more devices with that Plug and Play ID in the ACPI device tree. Each device declared in the ACPI device tree would have its own operating system device object with its own set of mappings.

Written by freevanx Friday, 05 November 2010 02:51 -

In this way, different sets of data blocks can be organized in the appropriate place within the device tree. This organization allows the different devices and their corresponding data blocks to come and go from the ACPI device tree. Note that if there are multiple WMI-mapping pseudo devices in the ACPI device tree, each device must have a unique value for its UID.

Mapping Driver Functionality. Essentially the mapping driver will do the following:

- Manage all registration and unregistration with WMI and other interactions with the operating system. This registration of data and data blocks is done when the mapper gets the IRP MN REGINFO IRP.
- At WMI initialization time, the mapping driver will query an ACPI method for the list of data block, event, and method GUIDs it supports, as well as the mapping from the GUID to control method IDs. With this information, the mapping driver will register those GUIDs with WMI.
- Translate data block queries, sets, and method call I/O IRPs into the appropriate ACPI control method call.
- Receive notifications from the ACPI event handler control methods and relaunch them as WMI events.
- Translate strings between ASCIZ and UNICODE for data blocks marked as wholly composed of a string.

ACPI-to-WMI Mapper Goals for Windows 2000 Instrumentation

The following list describes the goals for the ACPI-to-WMI mapper:

Written by freevanx Friday, 05 November 2010 02:51 -

- Expose data maintained by the hardware or firmware and accessible through ACPI to user-mode instrumentation data consumers, such as WBEM or DMI.
- Allow specific ACPI control methods to be called from a user-mode instrumentation data consumer or hardware configuration application—for example, from a control panel applet.
- Deliver specific ACPI events to all user-mode callers that request notification of that particular event.
- Allow OEMs to include OEM-specific data blocks, user-mode—callable ACPI control methods, and events without requiring any changes to the ACPI-to-WMI mapper.
- Allow general-purpose data consumer applications—those written without any special knowledge of the data blocks exposed by a particular machine—to be able to access and understand the data blocks, user-mode—callable ACPI control methods, and events being mapped—including those that are OEM specific.
- Define standard data block formats, user-mode—callable ACPI control methods, events and their WMI GUID mappings for common data blocks, and callable control methods and events expected to be provided by all OEMs. This can include dynamic data and functions identified by the industry in the SMBIOS specification and other specifications.

These goals are achieved by having supporting code in the ACPI-to-WMI mapper (Wmiacpi.sys) as well as in the core ACPI code itself (Acpi.sys).

The following are *not* goals for the ACPI-to-WMI mapper:

- To have specific knowledge about any data block that passes through the mapper.
- To provide interfaces specifically for SMBIOS data and functions. The mapper is an open architecture that is not restricted to SMBIOS data and functionality.

Written by freevanx Friday, 05 November 2010 02:51 -

How SMBIOS-provided information is handled. Vendors who want to provide OEM and system-specific instrumentation data may choose to use SMBIOS as the mechanism. To use the capabilities of the WMI infrastructure to surface this SMBIOS data, they must conform to any SMBIOS version between 2.0 and 2.2. (Microsoft intends to support future revisions of SMBIOS as and when they appear. To date, we have only one SMBIOS 2.2 machine for testing). This allows the Microsoft Win32® provider—which is shipped with Windows 2000 and future versions of Windows and is available as an update to Windows 98—to populate almost all of the SMBIOS-provided information into the CIMv2 namespace. In particular, almost all of the information is put into Win32 classes. Some of these Win32 classes are derived from the CIMv2.1 physical MOF.

The one exception where SMBIOS information will not be automatically populated by the Win32 provider into the CIMv2 namespace is SMBIOS vendor-specific data. Such SMBIOS vendor-defined data will be placed in a "VendorBucket" class in a "RootVendorDefined" namespace, and will not be available in the CIMv2 namespace by default. Any system vendor who wants to provide such data must write a provider that will interpret this data.

The SMBIOS data is read only once, either at boot time in Windows 2000 or post boot on Windows 98. Dynamic updates that are made to the SMBIOS data after it has been read will not be reflected in the namespaces in this implementation. Microsoft is working with the industry to define standard ACPI methods for dynamic updates.

The SMBIOS raw data will be available as a WMI data block in Windows 2000 and as a flat file in Windows 98. This data will be interpreted and populated into the namespaces by the Win32 provider.

Written	by	freevanx	
Friday,	05	November 2010 02:51	

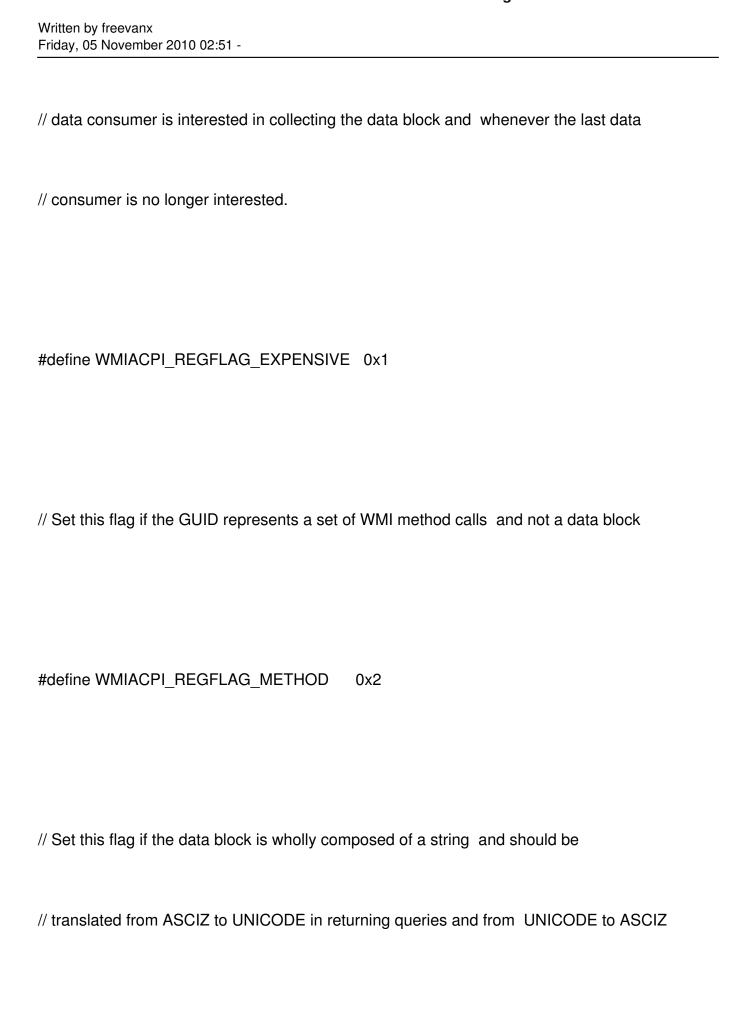
Note: The Win32 provider capabilities used to read and populate SMBIOS data as described above will be available in Windows 2000 Beta 3.

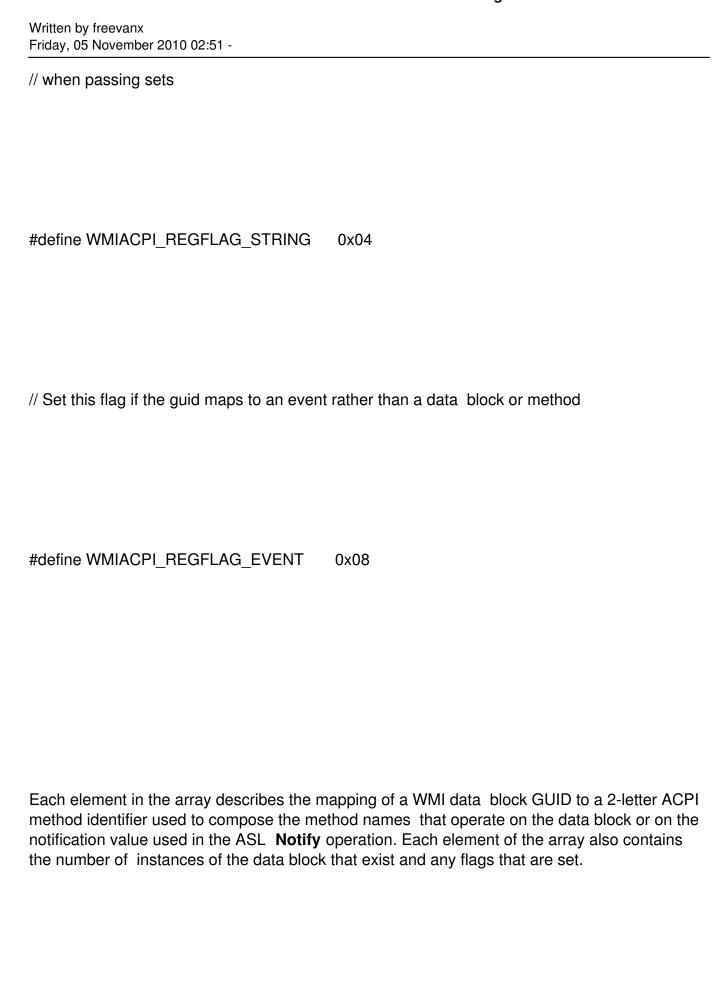
ACPI Control Method Naming Conventions and Functionality for Windows 2000 Instrumentation

The Data Block GUID Mapping control method named _WDG evaluates to a buffer that has the GUID mapping information for data blocks, events, and WMI methods. The result of the evaluation is a buffer containing an array of the following structure.

typedef struct			
{			
GUID guid;	// GUID	that names dat	a block
union			
{			

Written by freevanx Friday, 05 November 2010 02:51 -CHAR ObjectId[2]; // 2-character ACPI ID (Data Blocks and Methods) struct { UCHAR NotificationValue; // Byte value passed by event handler control method UCHAR Reserved[1]; } NotifyId; } USHORT InstanceCount; // Number of separate instances of data block **USHORT Flags**; // Flags **}**; // Set this flag if the WCxx control method should be run to whenever the first

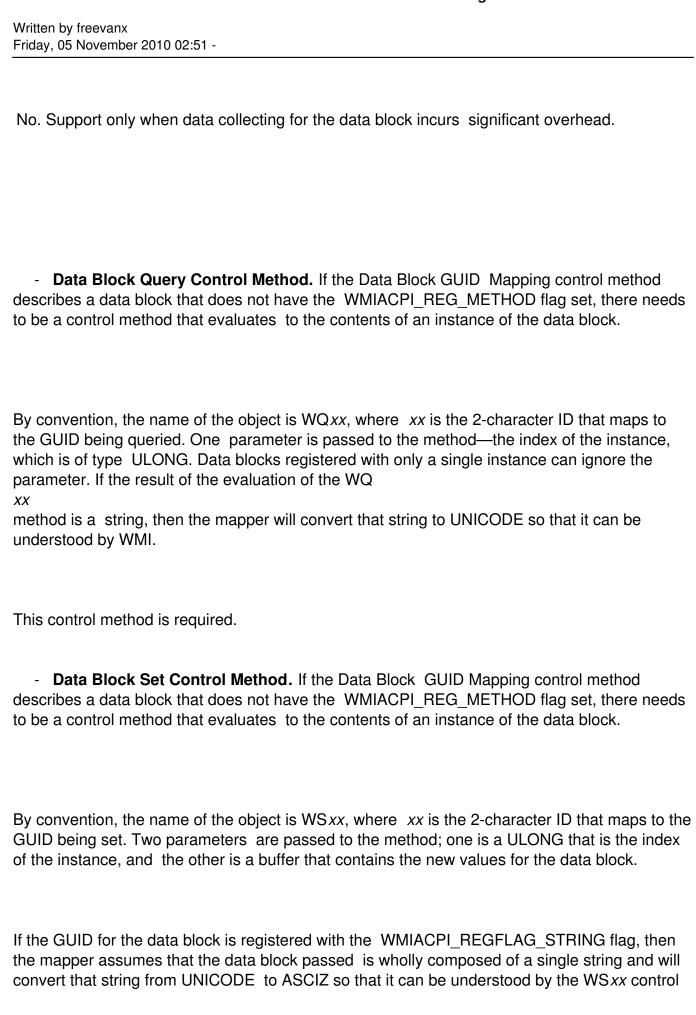




Friday, 05 November 2010 02:51 -
This control method is required.
The following table summarizes the information for each control method described later in this section.
Control Method Summary
Control method
Object name
Parameters
Control method required
Data Block Query
WQ xx

Written by freevanx Friday, 05 November 2010 02:51 -
ULONG
Yes.
Data Block Set
WS xx
ULONG, buffer
No. Not required for data blocks that are read only.
Data Item Set
Not supported
Method Execution

Written by freevanx Friday, 05 November 2010 02:51 -
WM xx
ULONG, method ID, buffer
Yes. Required for data blocks specified in the data block GUID mapping array and that have the WMIA
Event Enable and Disable
WE xx
UCHAR
No. Support only when keeping the event enabled incurs significant overhead.
Data Collection Enable and Disable
WC xx
UCHAR



Written by freevanx Friday, 05 November 2010 02:51 -

method.

This control method is not required for those data blocks that are read only.

Because the mapper is unaware of the format of the data block passed by the caller of the WMI method, it will pass the buffer as a single buffer parameter to the WMxx control method. The control method can use the ASL instructions

CreateBitField

CreateDWordField

CreateField

, and

CreateWordField

to break up the buffer into its parts

- **Data Item Set Control Method.** Setting of individual data items within a data block is not supported. Typically, data items that can be modified should be placed in their own data block or the entire data block should be modified.
- **Method Execution Control Method.** If the Data Block GUID Mapping control method describes a data block that does have the WMIACPI_REG_METHOD flag set, there needs to be a control method that performs the action required of the method.

By convention, the name of the control method is WMxx, where xx is the 2-character ID that maps to the GUID. This method call has three parameters; the first is a ULONG that has the instance index being executed; the second contains the method ID for the method being executed; and the third is a buffer that contains the input for the method call.

If the GUID for the WMI method is registered with the WMIACPI_REGFLAG_STRING flag, then the mapper assumes that buffer passed in is a string. The mapper will convert the incoming string from UNICODE to ASCIZ.

If the result of the WMxx control method is a string, the mapper will convert the result string from ASCIZ to UNICODE. The return value of the WM xx method

Written by freevanx Friday, 05 November 2010 02:51 -

should be a buffer with the result of the method call.

This control method is required for those data blocks specified in the Data Block GUID Mapping array that have the WMIACPI REG METHOD flag set.

Because the mapper is unaware of the format of the data block passed by the caller of the WMI method, it will pass the buffer as a single buffer parameter to the WMxx control method. The control method can use the ASL instructions

CreateBitField

CreateDWordField

CreateField

, and

CreateWordField

to break the buffer into its parts.

- **Event Enable and Disable Control Method.** For each event specified in the Event GUID Mapping control method that has the WMIACPI_REG_EXPENSIVE flag set, there needs to be a control method that is invoked whenever launching of the event should be enabled and disabled.

By convention, the control method is named WExx, where xx is the hex value of the notification code passed by the event handler control method. This method has one parameter, a UCHAR that has a value of 0 if the event is to be disabled or a nonzero value if it is to be enabled.

This method is optional and should only be supported if keeping the event enabled incurs significant overhead.

- **Data Collection Enable and Disable Control Method.** For each data block described in the Data Block GUID Mapping control method that has the WMIACPI_REG_EXPENSIVE flag set, there needs to be a control method that is invoked whenever collection of the data that composes the data block should be enabled and disabled

Written by freevanx Friday, 05 November 2010 02:51 -

By convention, the control method is named WCxx, where xx is the 2-character ID that maps to the GUID. This method has one parameter, a UCHAR that has a value of 0 if data block collection is to be disabled or a nonzero value if it is to be enabled.

This method is optional and should only be supported if collecting the data for the data block incurs significant overhead.

- **Additional Event Data.** The _WED control method is evaluated by the mapper in response to receiving a notification from a control method. The results of the evaluation are passed as part of the WMI event information. This mechanism allows additional data to be included with an event.

The control method takes one parameter, which is the notification code that caused the notification to occur. If the result of the _WED control method is a string, then the string is converted from ASCIZ to UNICODE before launching the WMI event.

Data Block Design for Windows 2000 Instrumentation

Design Considerations. Consider the following in designing data blocks for instrumentation under Windows 2000:

- Data items that are read-only and are commonly used together should be combined into a single data block.
- Data items that are strings must be segregated into their own data block and registered with the WMIACPI_REGFLAG_STRING flag set so that the mapper can convert between ASCIZ and UNICODE.
- Data items that can be set individually should be segregated into their own data blocks. For example, a set of data items that must be set all at the same time can be combined into a single data block.

Written by freevanx Friday, 05 November 2010 02:51 -

MOF Data Types. The Managed Object Format (MOF) for the data blocks implemented can be supplied as either a resource attached to a file or as the buffer that results from the evaluation of a control method. To establish the former, either bind the resource to the Wmiacpi.sys image or establish a REG_EXPAND_SZ registry value named

MofImagePath under the WMIACPI service key. The contents of the value is a path to the image file that contains the resource. In either case, the resource must be named

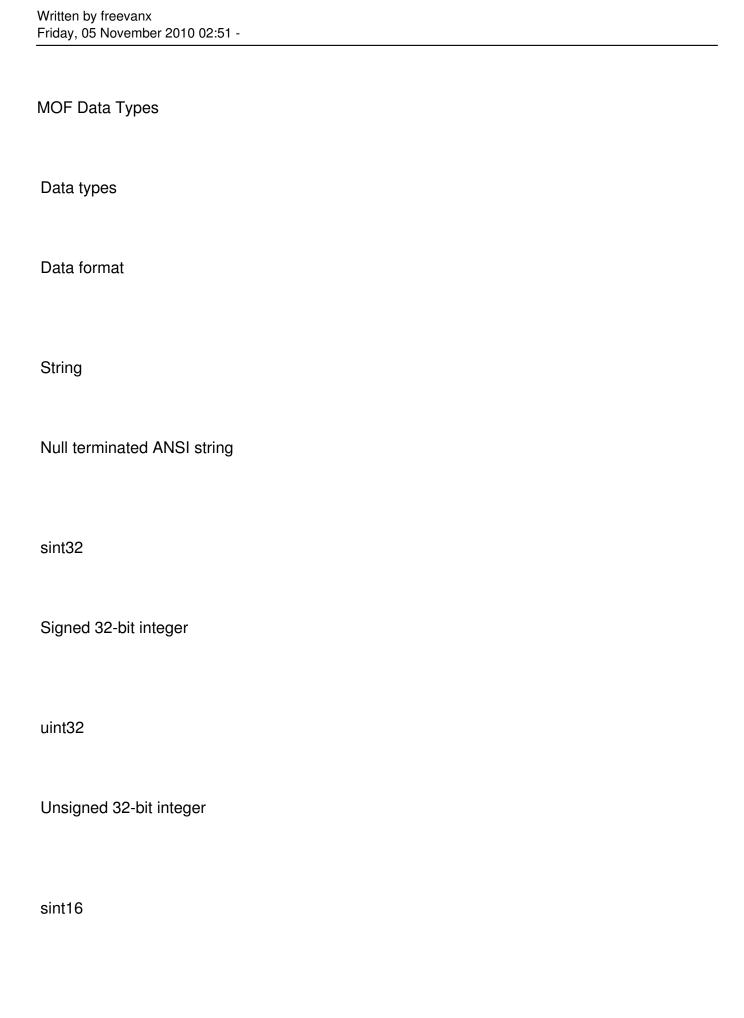
MofResourceName

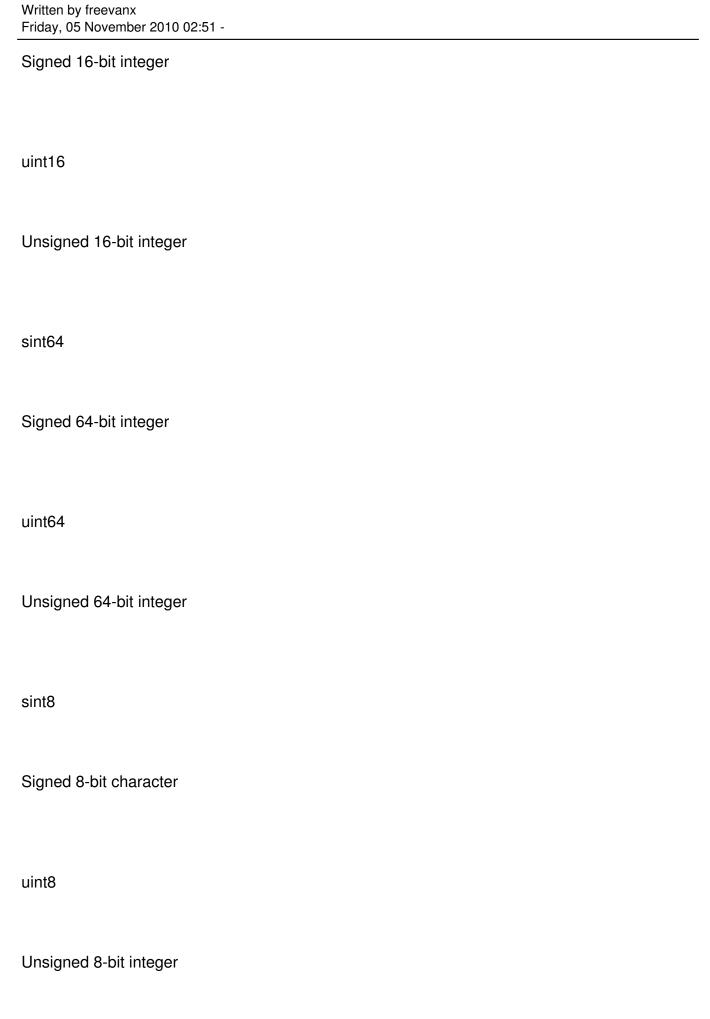
.

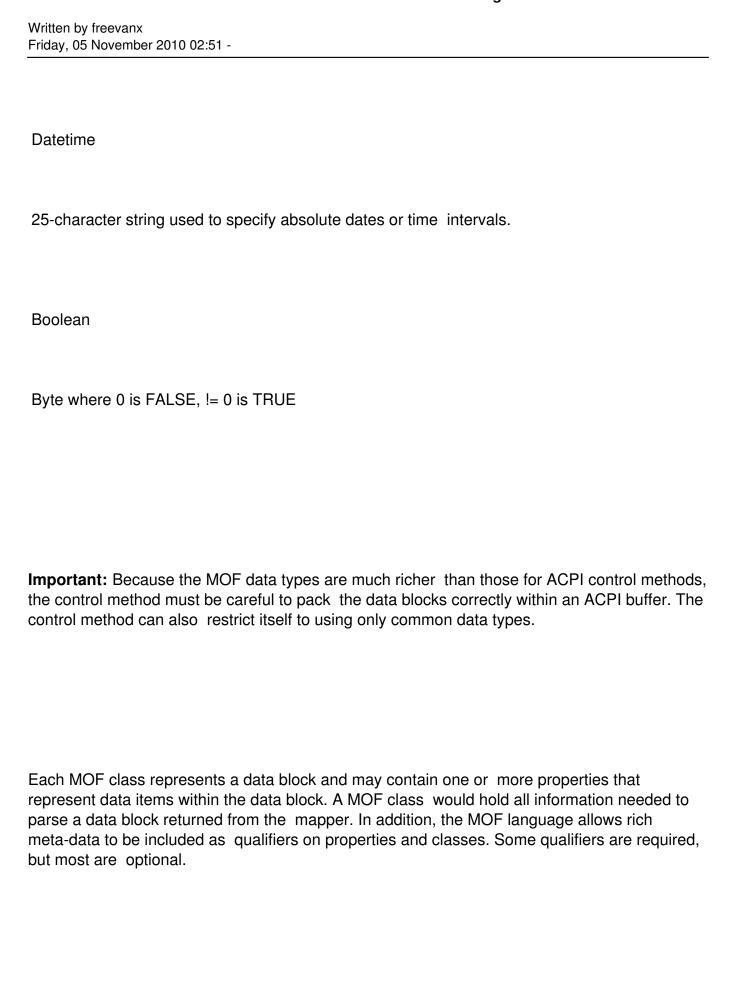
The buffer resulting from the evaluation of the WQxx control method assigned to the binary MOF GUID describes all data blocks, WMI methods, and events for the device in a compressed binary format. This binary data is created by building a text file using the MOF language and compiling it with the MOF compiler.

MOF data types are very rich. MOF supports the basic data types of 8-, 16-, 32-, and 64-bit signed and unsigned integers, Boolean terms, floating points, strings, and UTC datetimes. Embedded classes—that is, structures that can contain basic data types and other embedded classes—are also supported. In addition, fixed and variable length arrays of basic data types and embedded classes are supported.

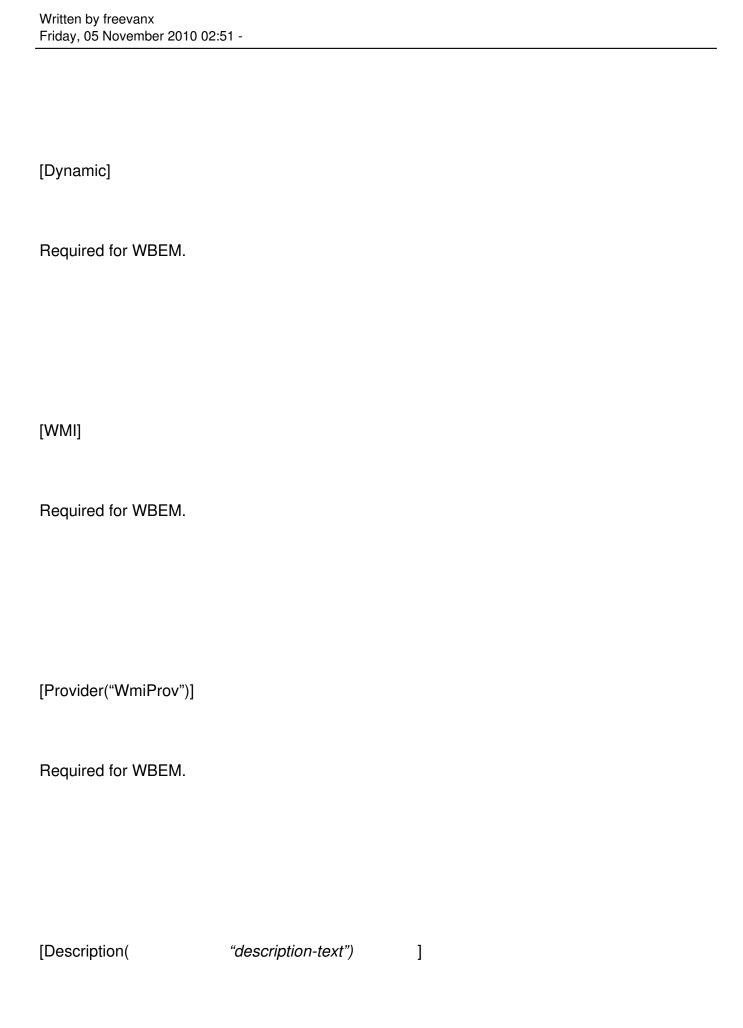
The MOF language defines the data types shown in the following table.





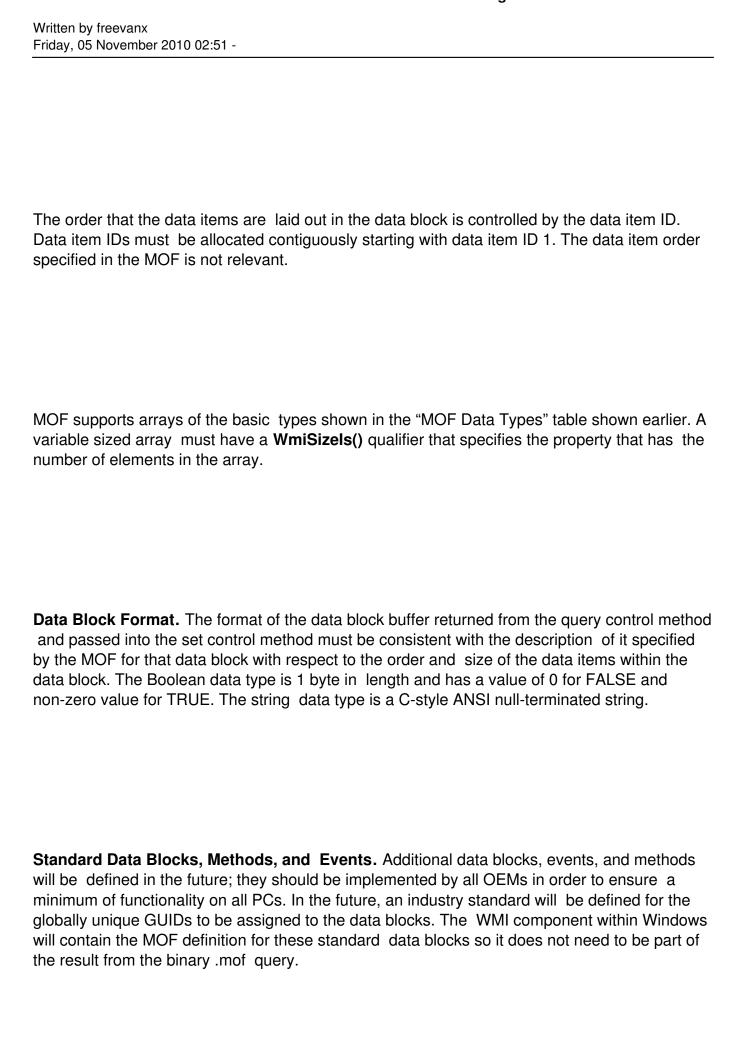


Friday, 05 November 2010 02:51 -				
Class and data item qualifi	ers are defined in th	e following table.		
Class and Data Item Quali	fiers			
Qualifier				
Description				
Class qualifiers:				
[guid("	guid-string	")]		
Declares the GUID that represents the class within WMI. This qualifier is required.				



Written by freevanx Friday, 05 November 2010 02:51 -				
Specifies description tex	t for the class or property	in the locale	specified for the loca	le qualifier
[WmiExpense(expense-value)]		
Specifies the quantity of	system resources require	edin <i>eogolems</i> te	- valle ct data in the da	ta block;
Data item qualifiers:				
[read]				
Specifies that the data it	em may be read.			
[write]				

Written by freevanx Friday, 05 November 2010 02:51 -				
Specifies that the data item may be written.				
[WmiDataID(data-item-ID)]		
Specifies the data item ID for the data item. This qualifier is required.				
[WmiScale(scale-factor)]		
Specifies the scaling fac	tor to use when displaying	thes data-factore displaying the data returned from a		
[WmiComplexity("	complexity-category")]		
Specifies the level of detail associated with the counterComplexity-category				
[WmiVolatility(validity-interval)]		
Specifies how often this data item value is updated intelvatibility-interval				
[WmiSizeIs("	property-name	")]		
Specifies the property within the current class that has the count of the number of array elements (not be				



Written by freevanx Friday, 05 November 2010 02:51 -

Custom Data Blocks, Methods and Events. Custom or OEM-specific data blocks, events, and methods can be added by including them in the result of the _WDG method. The GUIDs that are assigned must be globally unique so they can be generated by a tool such as Guidgen or Uuidgen, which are provided with the WMI information in the Microsoft Platform SDK.

The MOF definition for these custom data blocks must be included in the results of the WQxx method, where

XX

has been mapped to the MOF Data GUID, which is the GUID that is queried and returns MOF data—in order for applications to be able to access the data blocks. Or the MOF could be added as a resource to Wmiacpi.sys with a name of

MofResourceName

and a type MOFDATA. It can also be a resource in another image file with same name and type that is pointed to by the

MoflmagePath

value in the registry key HKLMCurrentControlSetServicesWmiAcpi

FAQ about WMI and ACPI for Windows 2000 Instrumentation

How does WMI find ACPI/ASL code?

In ASL, the developer creates a device with an _HID of PNP0c14. The operating system enumerates the device and loads the Wmiacpi.sys driver on top of it.

Written by freevanx Friday, 05 November 2010 02:51 -

How does the MOF associated with ACPI BIOS get registered?

It is either a resource attached to Wmiacpi.sys or another image file such as a resource-only DLL.

How does a management application discover the classes and properties provided by ASL instrumentation?

By looking in the WMI namespace of the schema.

Is the following true? Because very few ACPI standards exist for instrumentation, most of the ACPI instrumented features will appear differently on each vendor's product, and management applications will have to be "taught" to interpret the varying classes and methods.

Microsoft is looking at standardizing this. Any suggestions are appreciated.

Who provides the .mof files for standard ACPI features such as thermal monitoring?

Windows 2000 has a .mof file for thermal zone temperature as part of the operating system and instruments it within Acpi.sys, outside of the mapper.

Typically, .mof files are compiled into .bmf files and attached to a driver as a resource. The .bmf files can be in the ROM or on disk. WMI determines the location of the .mof information by looking at the registry for the **MoflmagePath** value under the WMIACPI service. If this does not exist, then WMI looks at the **ImagePath** value. If Wmiacpi.sys does not have a .mof resource, then WMI will query the binary .mof GUID for the .mof information.

Written by freevanx Friday, 05 November 2010 02:51 -

A driver may have a static list of pre-built .mof files; if so, it can "dynamically" report one of them. The mechanism is to report the file using a predefined GUID that returns a binary .mof.

To dynamically build a .mof file, a driver would have to build a .mof file and then launch the .mof compiler, which is difficult. Currently, to do this on the machine running Wmiacpi.sys, the **mofcomp**

command can be used to load the .mof file directly into the CIMOM database.

ASL Methods and Sample Code

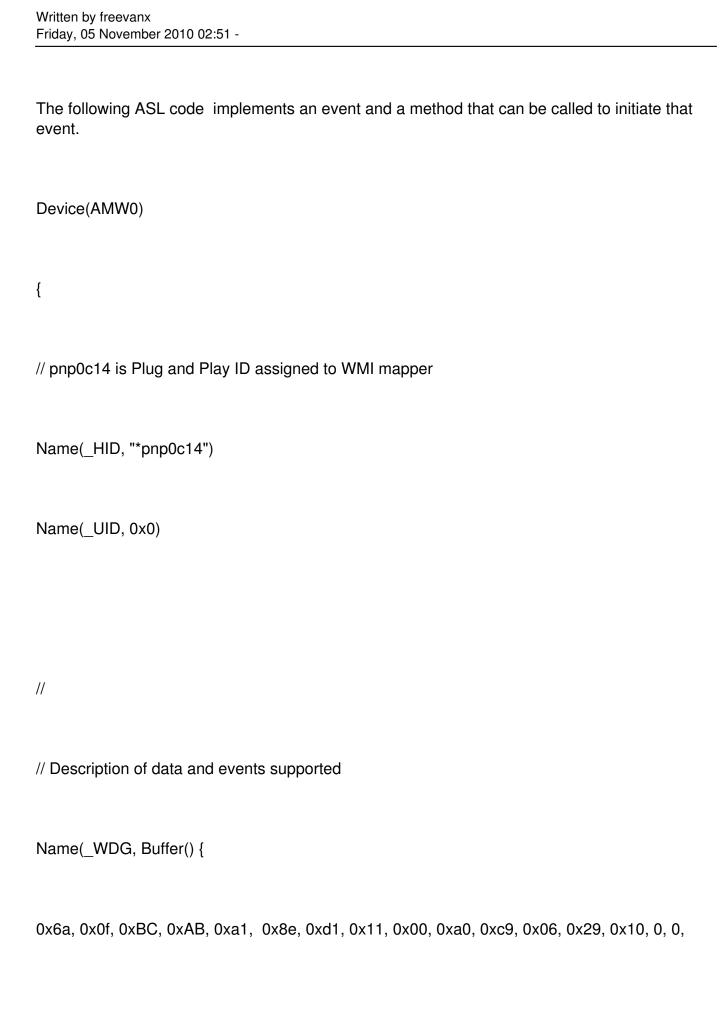
The following list represents some of the ASL methods defined by the ACPI specification. These methods are of particular interest for systems management. None of these methods have been implemented yet within the WMI/ACPI mapper to date. A BIOS developer, for example, could use these methods to expose data using the mapper. These methods represent good opportunities for OEMs to differentiate their products with minimal effort:

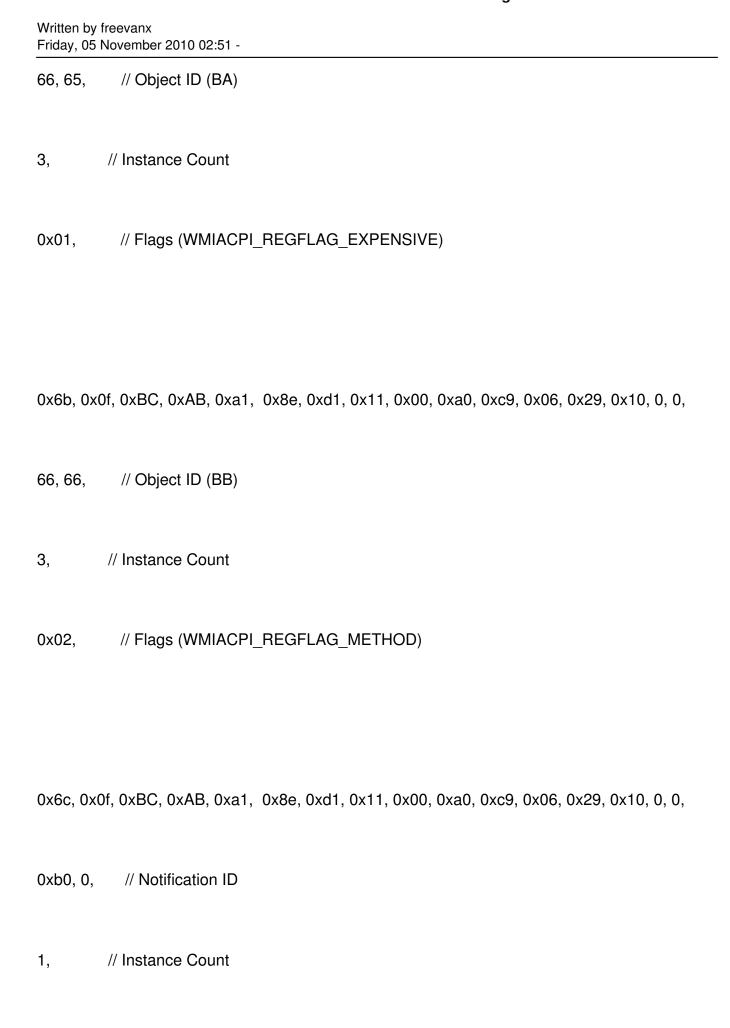
- ACX Temperature threshold at which various degrees of active cooling are engaged
- _CRT Critical temperature at which system will shut down
- _PSV Temperature at which system will throttle CPU in order to cool system
- LID Status of the lid (open or closed)
- PSR— Whether the machine running on AC

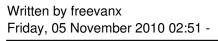
The same applies for these method for Control Method Battery devices:

- _BIF Battery information such as model, serial number, design capacity, last full charge capacity, technology, and battery capacity
- _BST Battery state, battery present rate, battery remaining capacity, and battery voltage present

ASL Sample Code for an Event and Initiating Method



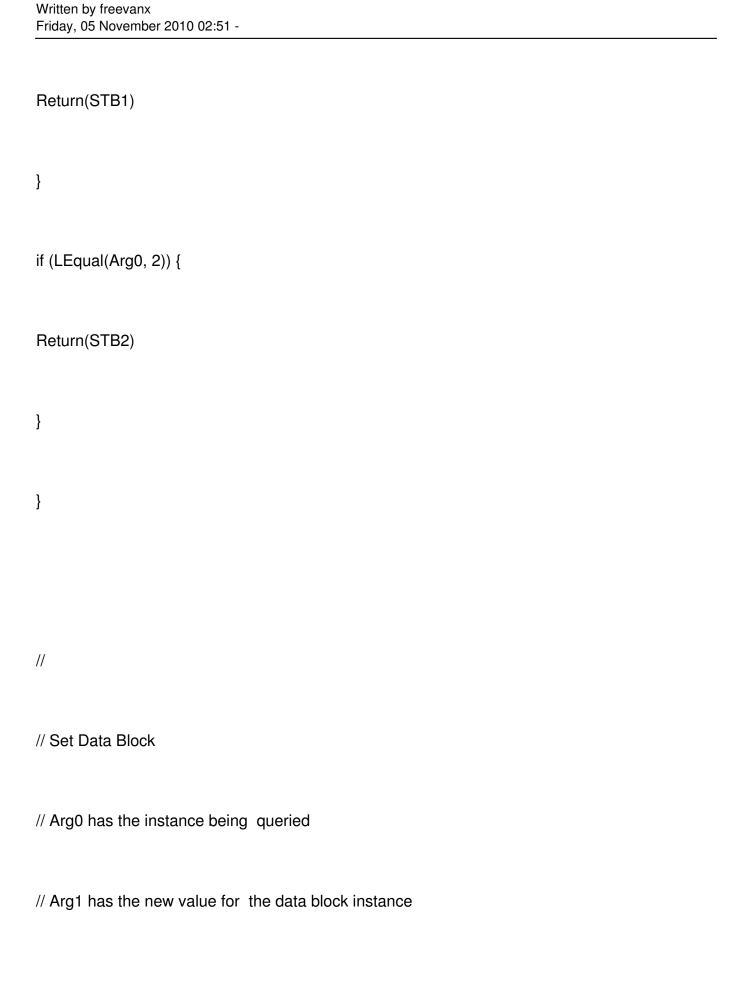




```
// Flags (WMIACPI_REGFLAG_EVENT)
80x0
})
//
// Storage for the 3 instances of BA
Name(STB0, Buffer(0x10) {
1,0,0,0, 2,0,0,0, 3,0,0,0, 4,0,0,0
})
Name(STB1, Buffer(0x10) {
0,1,0,0, 0,2,0,0, 0,3,0,0, 0,4,0,0
})
```

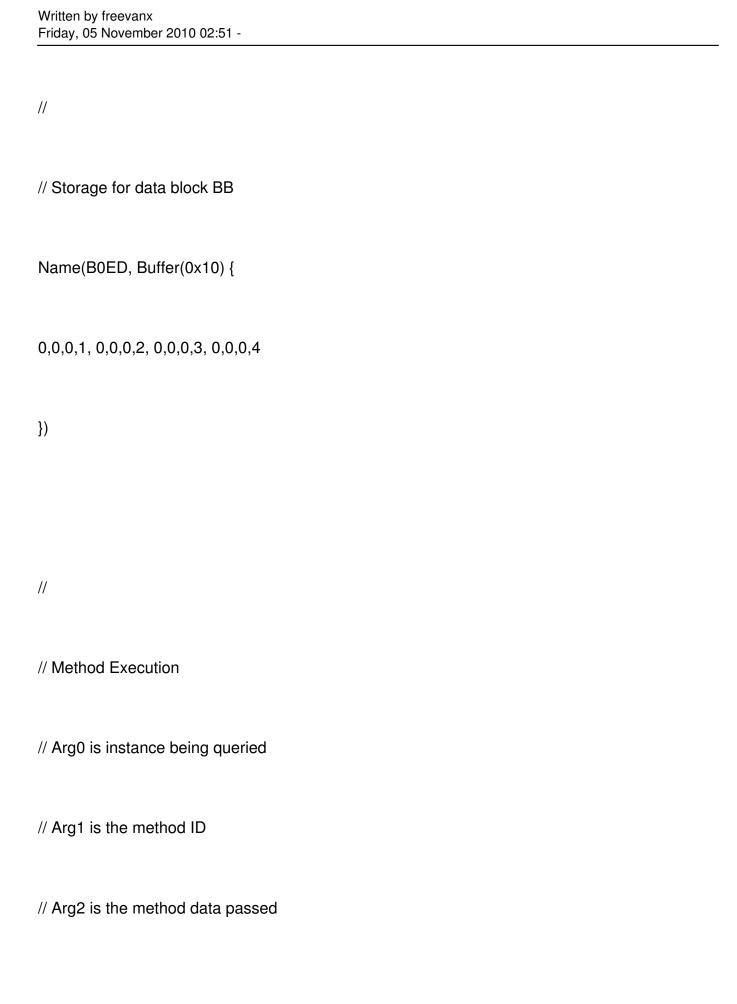
Written by freevanx

Friday, 05 November 2010 02:51 -Name(STB2, Buffer(0x10) { 0,0,1,0, 0,0,2,0, 0,0,3,0, 0,0,4,0 }) // // Query data block // Arg0 has the instance being queried Method(WQBA, 1) { if (LEqual(Arg0, 0)) { Return(STB0) } if (LEqual(Arg0, 1)) $\{$



Written by freevanx Friday, 05 November 2010 02:51 -

```
Method(WSBA, 2) {
if (LEqual(Arg0, 0)) {
Store(Arg1, STB0)
}
if (LEqual(Arg0, 1)) {
Store(Arg1, STB1)
}
if (LEqual(Arg0, 2)) {
Store(Arg1, STB2)
}
}
```



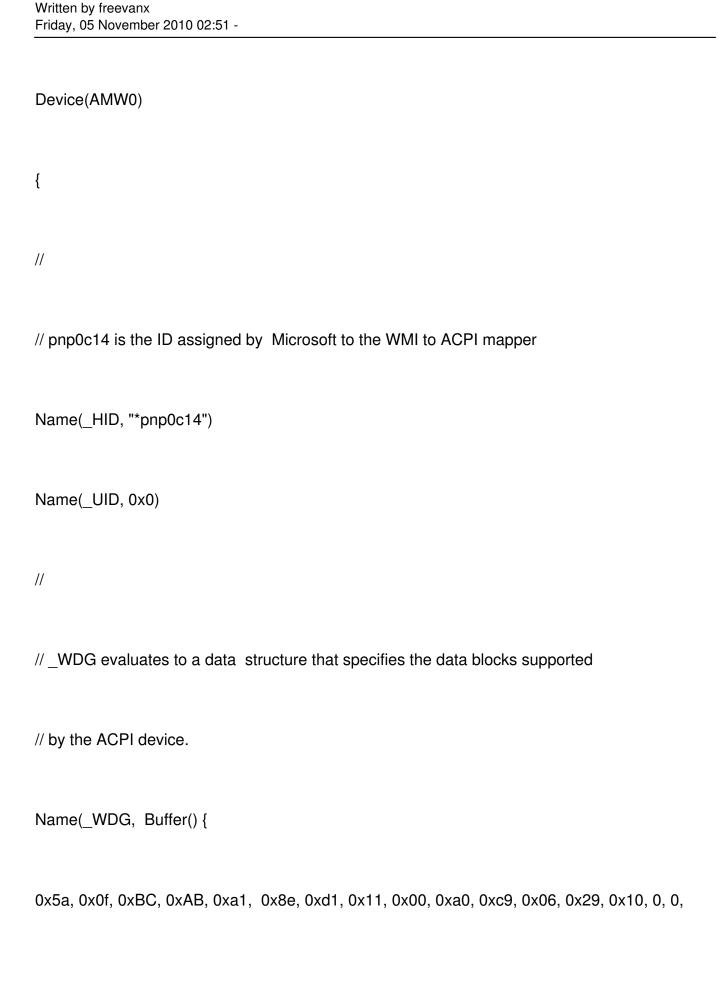
Written by freevanx Friday, 05 November 2010 02:51 -Method(WMBB, 3) { if (LEqual(Arg1, 1)) { Store(Arg3, B0ED) Notify(AMW0, 0xB0) Return(Arg3) } else { Return(Arg1) } } //

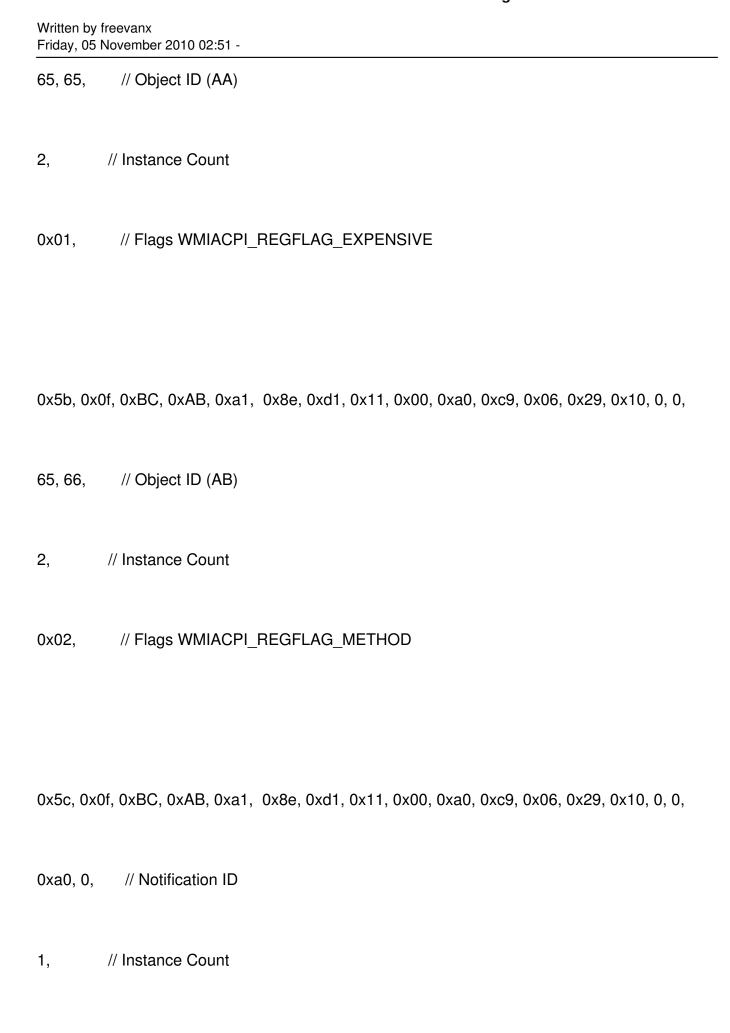


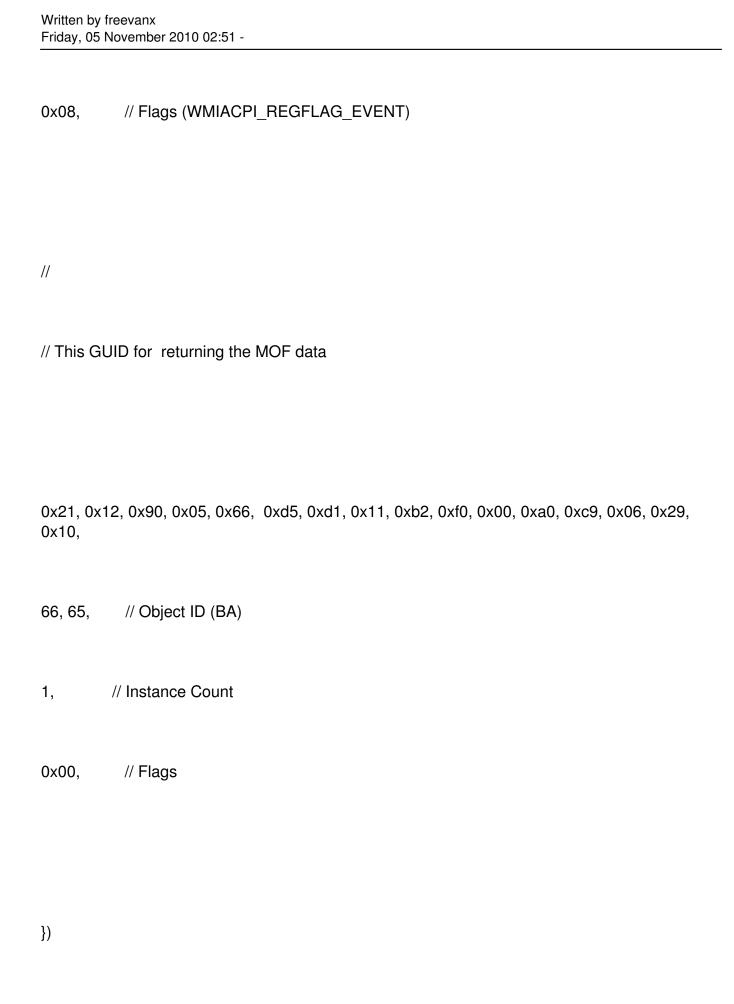
Sample ASL Code Embedding MOD Data in ASL

}

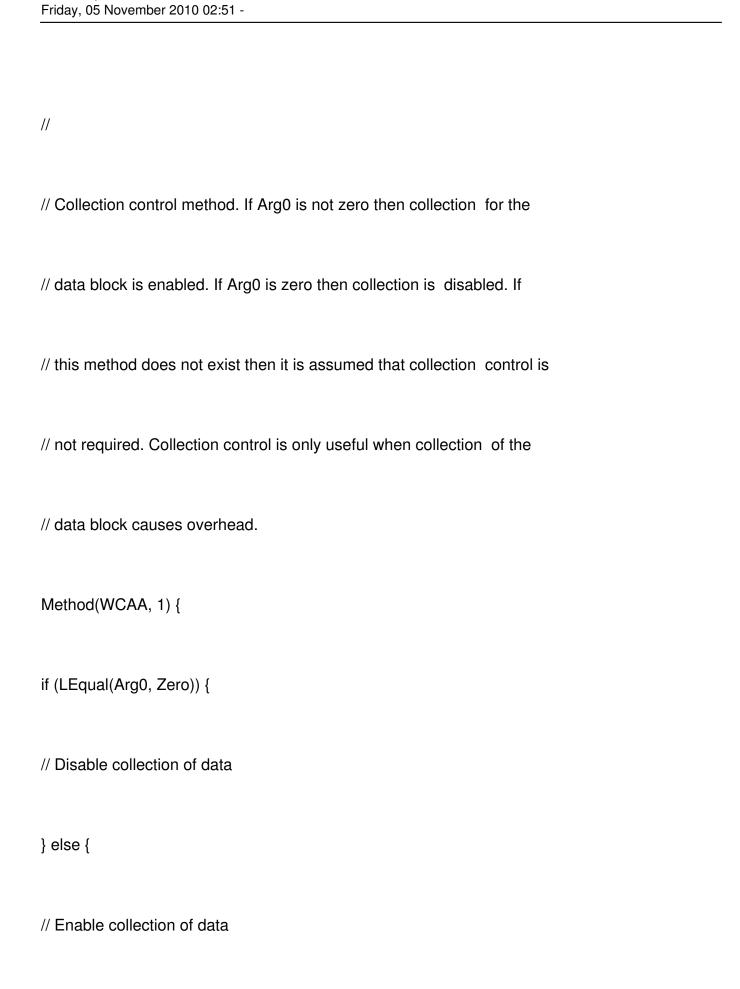
The following sample ASL code shows another example of implementing an event mechanism using ASL code. It also provides an example of embedding MOF data into ASL.

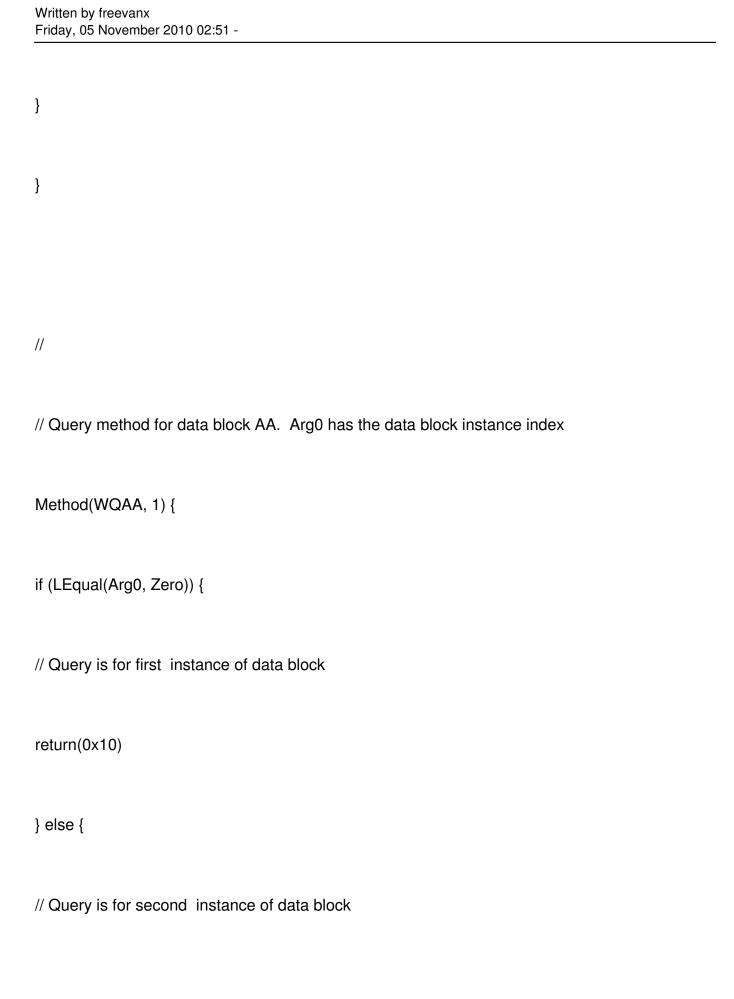


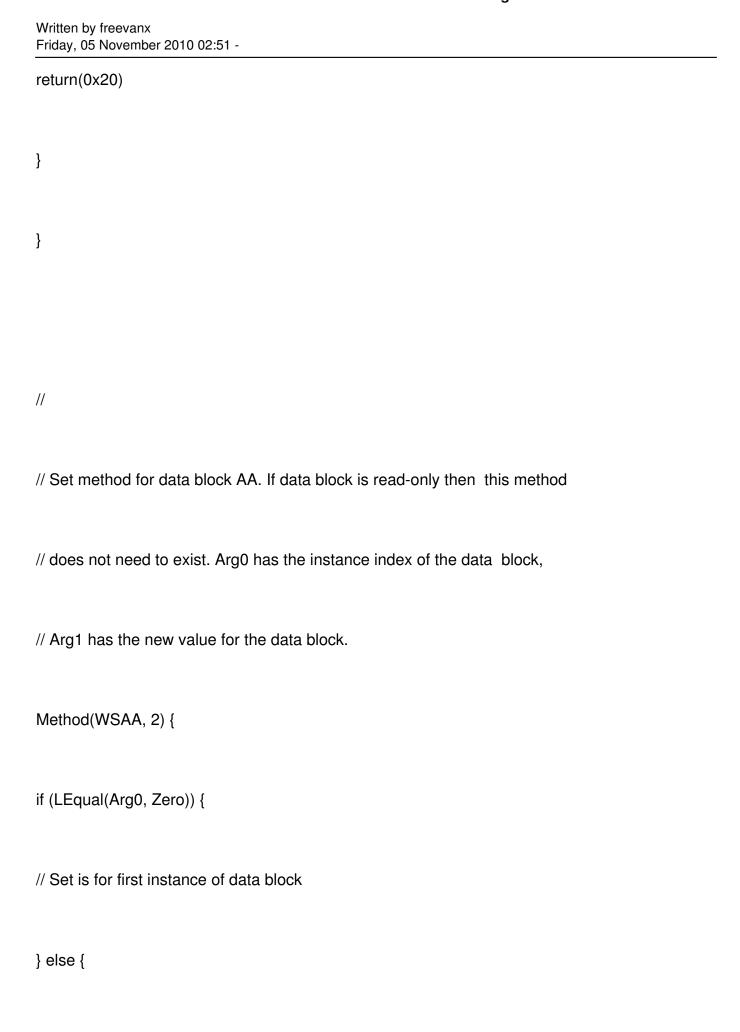


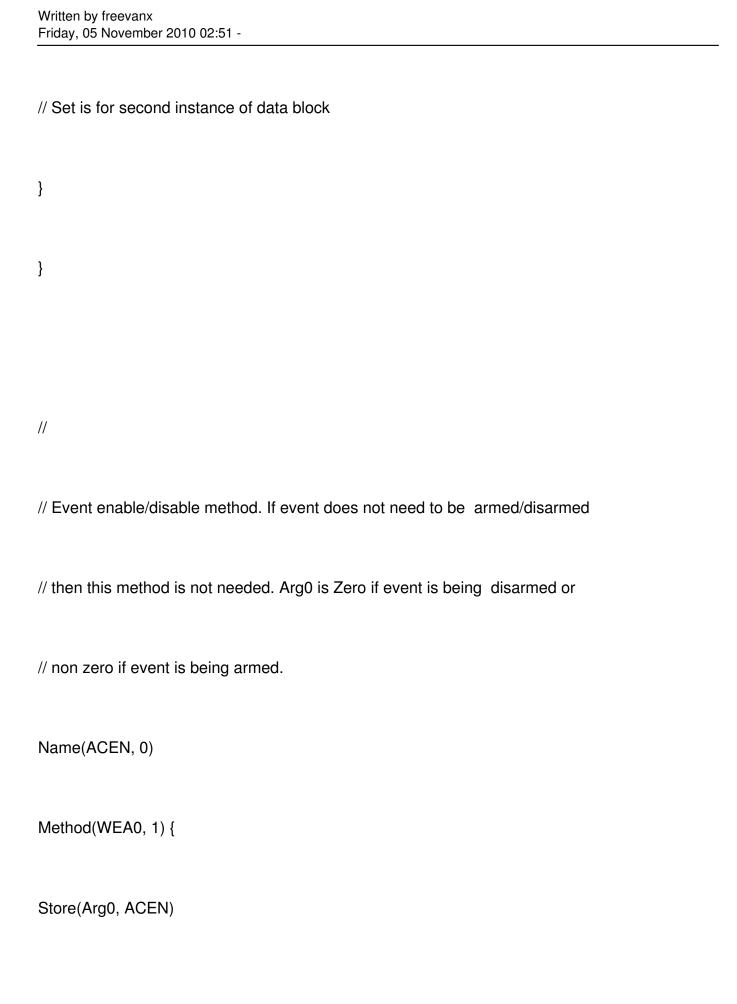


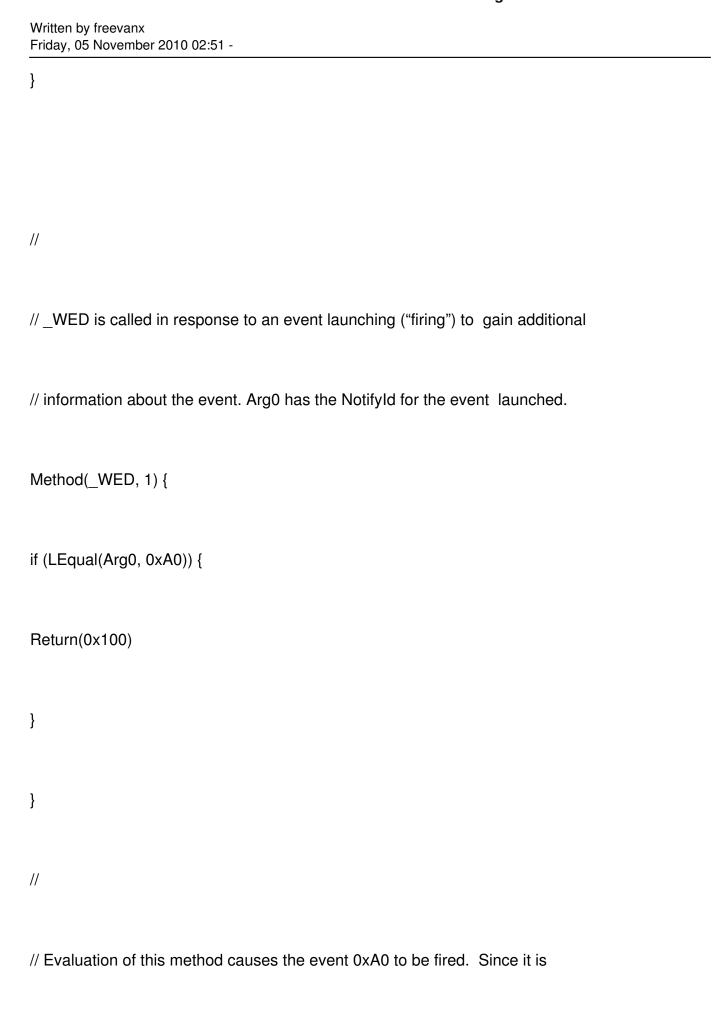
Written by freevanx

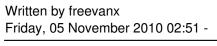












```
// defined by the _WDG method it is callable via WMI. Arg0 has the instance
// index and Arg1 has any input parameters.
Method(WMAB, 3) {
//
// If event was armed then launch it
if (LEqual(ACEN, 1)) {
Notify(AMW0, 0xa0)
}
Return(Arg1)
}
```

Name(WQBA, Buffer(926) {

0x46, 0x4f, 0x4d, 0x42, 0x01, 0x00, 0x00, 0x00, 0x8e, 0x03, 0x00, 0x00, 0x66, 0x0f, 0x00, 0x00,

0x44, 0x53, 0x00, 0x01, 0x1a, 0x7d, 0xda, 0x54, 0x98, 0xdd, 0x87, 0x00, 0x01, 0x06, 0x18, 0x42,

0x10, 0x0b, 0x10, 0x0a, 0x0b, 0x21, 0x02, 0xcb, 0x82, 0x50, 0x3c, 0x18, 0x14, 0xa0, 0x25, 0x41,

0xc8, 0x05, 0x14, 0x55, 0x02, 0x21, 0xc3, 0x02, 0x14, 0x0b, 0x70, 0x2e, 0x40, 0xba, 0x00, 0xe5,

0x28, 0x72, 0x0c, 0x22, 0x82, 0xfd, 0xfb, 0x07, 0xc1, 0x90, 0x02, 0x08, 0x29, 0x84, 0x90, 0x08,

0x58, 0x2a, 0x04, 0x8d, 0x10, 0xf4, 0x2b, 0x00, 0xa1, 0x43, 0x01, 0x32, 0x05, 0x18, 0x14, 0xe0,

0x14, 0x41, 0x04, 0x41, 0x62, 0x17, 0x2e, 0xc0, 0x34, 0x8c, 0x06, 0xd0, 0x36, 0x8a, 0x64, 0x0b,

0xb0, 0x0c, 0x2e, 0x98, 0xa3, 0x08, 0x92, 0xa0, 0xc6, 0x09, 0xa0, 0xc4, 0x4c, 0x00, 0xa5, 0x13,

0x5c, 0x36, 0x05, 0x58, 0xc4, 0x96, 0x50, 0x14, 0x0d, 0x22, 0x4a, 0x82, 0x13, 0xea, 0x1b, 0x41,

0x13, 0x2a, 0x57, 0x80, 0x64, 0x78, 0x69, 0x1e, 0x81, 0xac, 0xcf, 0x41, 0x93, 0xf2, 0x04, 0xb8,

0x9a, 0x05, 0x7a, 0x8c, 0x34, 0xff, 0x30, 0x41, 0x99, 0x14, 0x43, 0x0e, 0x20, 0x24, 0x71, 0x98,

0xa0, 0x9d, 0x59, 0xed, 0x18, 0xd2, 0x3d, 0x07, 0x32, 0x4d, 0x60, 0x21, 0x70, 0x9e, 0xb8, 0x19,

0xa0, 0xf0, 0x5b, 0x1d, 0x80, 0xe0, 0x2b, 0x1d, 0x15, 0xd2, 0xeb, 0x34, 0x64, 0x72, 0x46, 0x48,

0xf8, 0xff, 0x7f, 0x02, 0x26, 0xe3, 0xb7, 0x60, 0x02, 0xa5, 0xd9, 0xb2, 0x82, 0x4b, 0x80, 0xc1,

0x68, 0x00, 0x91, 0xa2, 0x69, 0xa3, 0xe6, 0xea, 0xf9, 0x36, 0x8f, 0xaf, 0x59, 0x7a, 0x9e, 0x47,

0x7a, 0x34, 0x56, 0x36, 0x05, 0xd4, 0xf8, 0x3d, 0x9d, 0x93, 0xf3, 0x4c, 0x02, 0x1e, 0x9c, 0x61,

0x4e, 0x87, 0x83, 0xf1, 0xb1, 0xb1, 0x51, 0x70, 0x74, 0x03, 0xb2, 0x31, 0x38, 0xc6, 0xb0, 0xd1,

0x73, 0x39, 0x81, 0x47, 0x82, 0x43, 0x89, 0x7e, 0x0e, 0x6f, 0x00, 0x47, 0x17, 0xe3, 0x04, 0xce,

Written by freevanx Friday, 05 November 2010 02:51 -

0x27, 0xc1, 0x61, 0x06, 0x39, 0xe3, 0x33, 0xf4, 0x44, 0x2c, 0x68, 0xd6, 0x02, 0x0a, 0x62, 0xa4,

0x58, 0xa7, 0xf5, 0x7c, 0x10, 0x8b, 0x41, 0x05, 0x8b, 0x11, 0xdb, 0x50, 0x87, 0x60, 0x18, 0x8b,

0x46, 0x11, 0xc8, 0x49, 0x3c, 0x49, 0x30, 0x94, 0x40, 0x51, 0x0c, 0x12, 0xda, 0xc3, 0x36, 0x92,

0x81, 0xcf, 0xdb, 0x20, 0xc7, 0x84, 0x51, 0x01, 0x21, 0xcf, 0xe3, 0xd0, 0x28, 0x4d, 0xd0, 0xfd,

0x29, 0x40, 0x37, 0x8b, 0x08, 0x67, 0x54, 0xd8, 0x44, 0x64, 0x6d, 0x02, 0xb2, 0x25, 0x40, 0x1c,

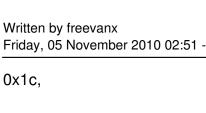
0xbe, 0x40, 0x1a, 0x43, 0x11, 0x44, 0x84, 0x98, 0x51, 0x8c, 0x19, 0x30, 0x82, 0x51, 0x0e, 0xa6,

0x39, 0x10, 0x69, 0x13, 0x30, 0xf6, 0x20, 0xd1, 0x62, 0x31, 0x04, 0xdb, 0x9f, 0x83, 0x30, 0x0e,

0x05, 0xa3, 0x03, 0x42, 0xe7, 0x84, 0xc3, 0x3b, 0x30, 0x9f, 0x1e, 0x4c, 0x70, 0xda, 0xcf, 0x07,

0xaf, 0x0b, 0x21, 0x8b, 0x17, 0x20, 0x0d, 0x43, 0xf8, 0x09, 0x6a, 0x7d, 0x51, 0xe8, 0x5a, 0xe0,

0x34, 0xe0, 0xa8, 0xeb, 0x82, 0x6f, 0x01, 0xbe, 0x01, 0x9c, 0xe0, 0xe3, 0x85, 0xf1, 0x83,



0xc1, 0x01, 0x3c, 0x44, 0xbc, 0x1a, 0x78, 0x08, 0x9e, 0xc3, 0xfb, 0x05, 0x3b, 0x0f, 0x60, 0xff,

0xff, 0x04, 0x5d, 0xe3, 0xe9, 0x92, 0x70, 0x02, 0x96, 0x83, 0x86, 0x1a, 0xac, 0x2f, 0x00, 0x27,

0xe9, 0xc1, 0x1a, 0xae, 0xae, 0xd3, 0x06, 0x7a, 0xba, 0xa7, 0x72, 0x5a, 0xa5, 0x0a, 0x30, 0x7b,

0x94, 0x20, 0x04, 0xcf, 0x1e, 0x6c, 0xde, 0x67, 0x73, 0xe6, 0x09, 0x9e, 0x14, 0x3c, 0x05, 0x3e,

0x2d, 0xcf, 0xd2, 0x97, 0x0e, 0x5f, 0x09, 0x7c, 0x9f, 0x30, 0x41, 0xf4, 0x27, 0x17, 0x36, 0x1a,

0xb8, 0xc3, 0xc6, 0x8d, 0x06, 0xce, 0xe5, 0xe0, 0xb1, 0xc3, 0x33, 0xf7, 0x5c, 0x4d, 0x50, 0xf3,

0xe5, 0x42, 0x4e, 0x66, 0x83, 0xd2, 0x03, 0xa2, 0x01, 0x3f, 0x34, 0x60, 0xd0, 0x1f, 0x19, 0xb8,

0xc8, 0x8b, 0x02, 0x95, 0x86, 0xac, 0xbf, 0x86, 0x45, 0x8d, 0x9b, 0x12, 0x58, 0xca, 0xa1, 0x82,

0xdc, 0x33, 0x7c, 0x9e, 0x38, 0x8c, 0x57, 0x00, 0xcf, 0xe6, 0xa0, 0x7c, 0x73, 0x71, 0xba, 0x7b,

0x05, 0x68, 0x66, 0x83, 0xbb, 0x51, 0x80, 0x05, 0xc3, 0xd7, 0x03, 0xdf, 0x30, 0xd8, 0xf1, 0xc3,

0xd7, 0x0c, 0x36, 0x24, 0x83, 0x45, 0x89, 0x14, 0x9b, 0x4d, 0xca, 0x03, 0xc0, 0xe0, 0xbd, 0xd7,

0xf8, 0x70, 0x61, 0x48, 0x9f, 0x31, 0xe0, 0x1e, 0x05, 0xe0, 0xfd, 0xff, 0xcf, 0x09, 0xe0, 0xb8,

0x6d, 0xf8, 0x2a, 0x62, 0x67, 0xf7, 0x0b, 0x5d, 0x6f, 0xb0, 0xf7, 0x1d, 0x78, 0xf8, 0x87, 0x85,

0xbb, 0x0b, 0x30, 0xb0, 0x13, 0xc5, 0x1c, 0x78, 0x80, 0xc7, 0x64, 0x1e, 0x78, 0xc0, 0x75, 0x96,

0x82, 0x3d, 0x04, 0xae, 0xfa, 0xc0, 0x83, 0xca, 0xf1, 0x6a, 0xa0, 0x67, 0x1e, 0xc0, 0xec, 0xff,

0xff, 0xcc, 0x03, 0x8c, 0xe0, 0x9f, 0x79, 0x80, 0x6b, 0xf4, 0x6b, 0x81, 0xde, 0x57, 0x3e, 0xf3,

0x00, 0x7c, 0x50, 0x79, 0x33, 0x01, 0xcd, 0xff, 0xff, 0x66, 0x02, 0xe3, 0xe0, 0xe0, 0x83, 0x88,

0xaf, 0x32, 0x3e, 0x11, 0x02, 0x93, 0xab, 0x09, 0x70, 0x09, 0x79, 0x27, 0xa2, 0x01, 0x07, 0x41,

0xaf, 0x01, 0x5c, 0x0b, 0x88, 0x66, 0xc8, 0xa6, 0x89, 0x25, 0x98, 0xe5, 0x22, 0x40, 0xef, 0x8a.

0x3e, 0x2a, 0xf1, 0x31, 0xfa, 0xa8, 0xc4, 0x70, 0xdf, 0x85, 0x8c, 0x7b, 0x7a, 0x67, 0xf7, 0xac,

0x84, 0xb9, 0x04, 0xbc, 0x8f, 0x80, 0x65, 0xf2, 0xf8, 0xd3, 0x07, 0x47, 0xf4, 0x85, 0xc1, 0x77,

0x23, 0x78, 0x04, 0xd5, 0x5f, 0x65, 0xa8, 0xfe, 0xbd, 0x48, 0x2f, 0x0c, 0xea, 0x2a, 0x03, 0x5c,

0xff, 0xff, 0x57, 0x19, 0x36, 0xc8, 0x63, 0x05, 0xcb, 0xf9, 0x11, 0x33, 0xc7, 0xd3, 0x8c, 0xe2,

0xa9, 0x78, 0xb8, 0xec, 0x62, 0x65, 0xef, 0x53, 0x25, 0xc7, 0x17, 0x5f, 0xab, 0xf0, 0x20, 0x8f,

0x31, 0xbe, 0xc3, 0x80, 0x71, 0x04, 0xef, 0x30, 0xc0, 0x35, 0xf0, 0xcb, 0x41, 0xd7, 0x40, 0xc0,

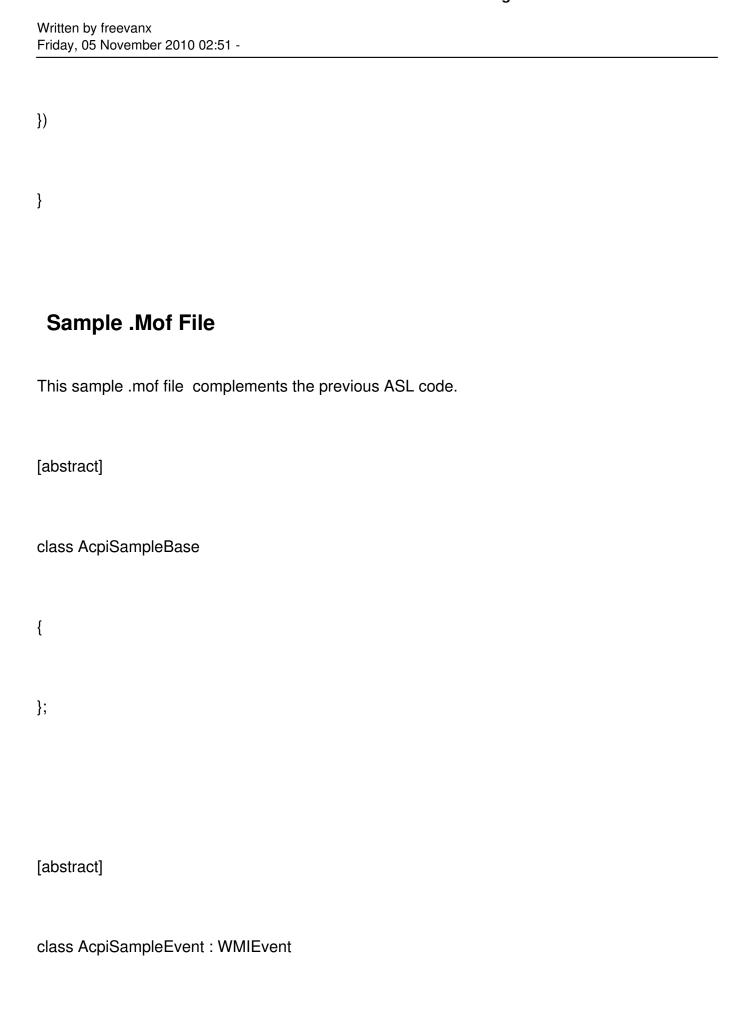
0xf6, 0xff, 0xff, 0x0e, 0x03, 0x96, 0xe0, 0x10, 0xba, 0x06, 0xe2, 0x64, 0x1c, 0x5b, 0xc8, 0x4d,

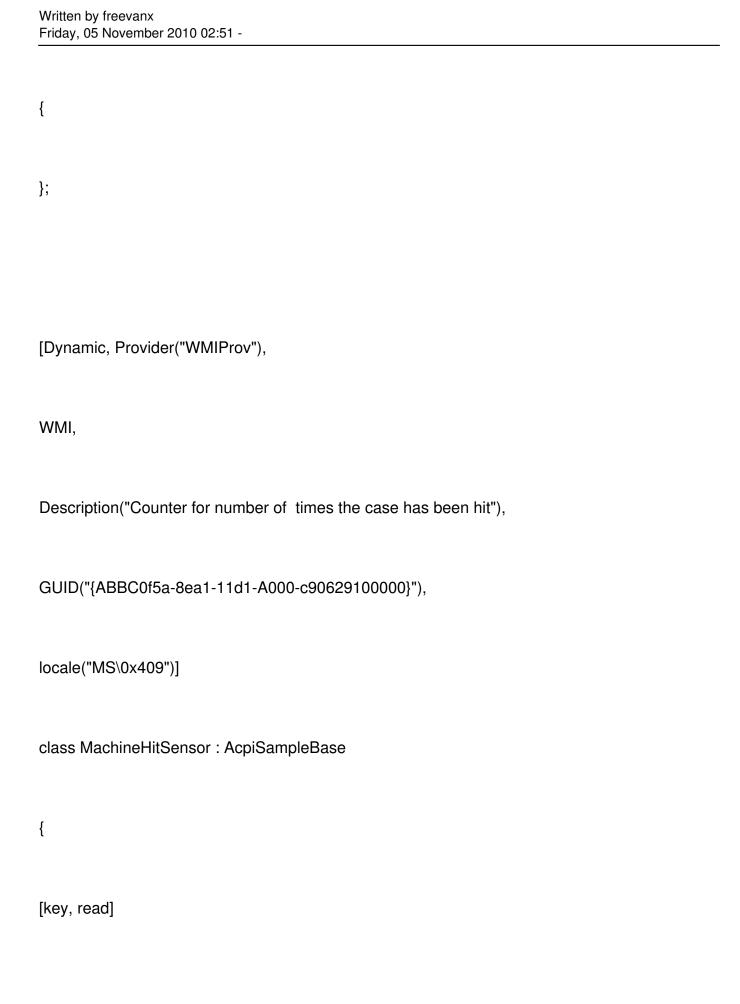
0xca, 0x53, 0x36, 0xc1, 0xa0, 0x13, 0xa6, 0x47, 0x40, 0xf0, 0xdc, 0x2b, 0x7c, 0x98, 0x00, 0xc7,

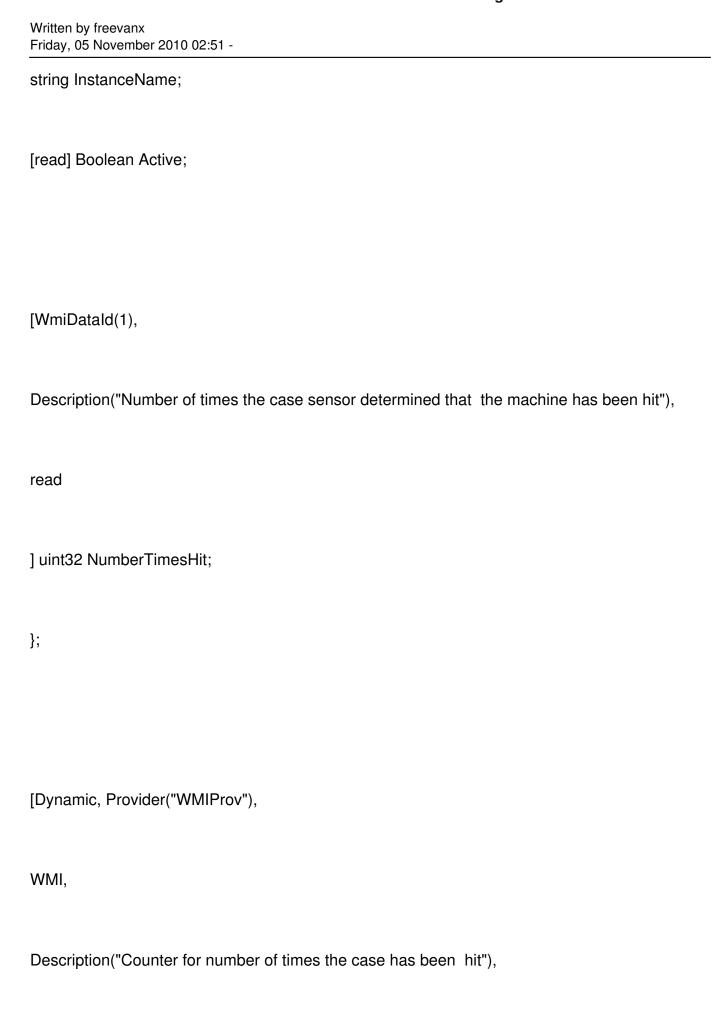
0x48, 0x30, 0xe7, 0x08, 0x9f, 0x1f, 0x7c, 0x7d, 0x78, 0x93, 0x60, 0x37, 0x0e, 0xc3, 0xf8, 0xca,

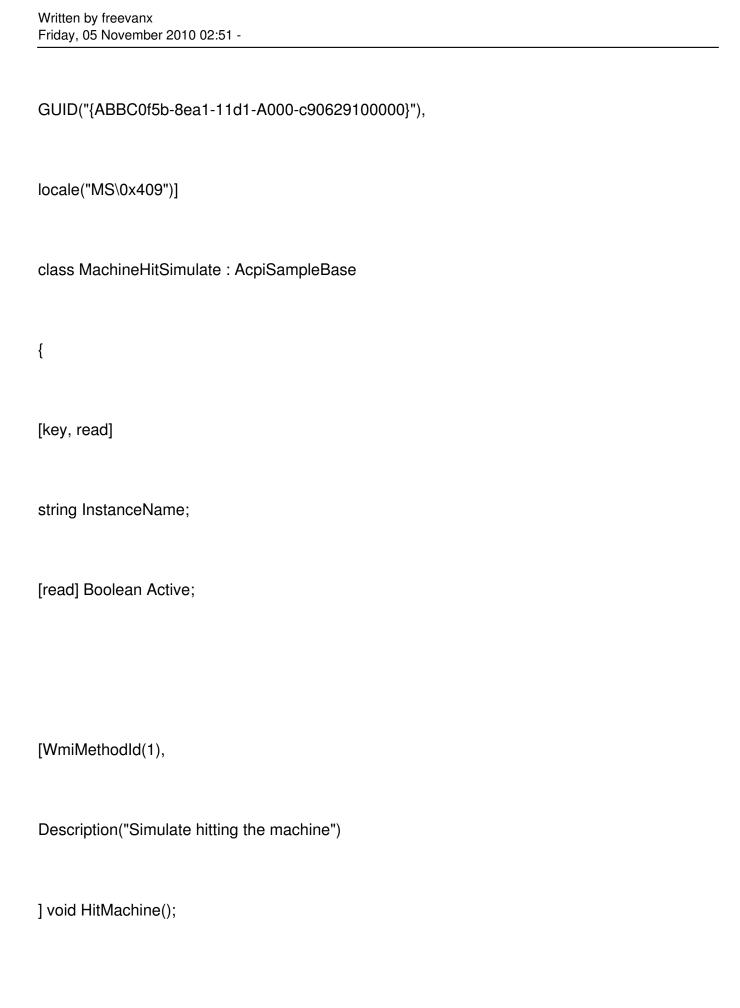
0x07, 0x0f, 0xf2, 0x15, 0x8b, 0x5d, 0x26, 0xf8, 0x49, 0x0f, 0x6c, 0x17, 0x65, 0x70, 0xdc, 0x7f,

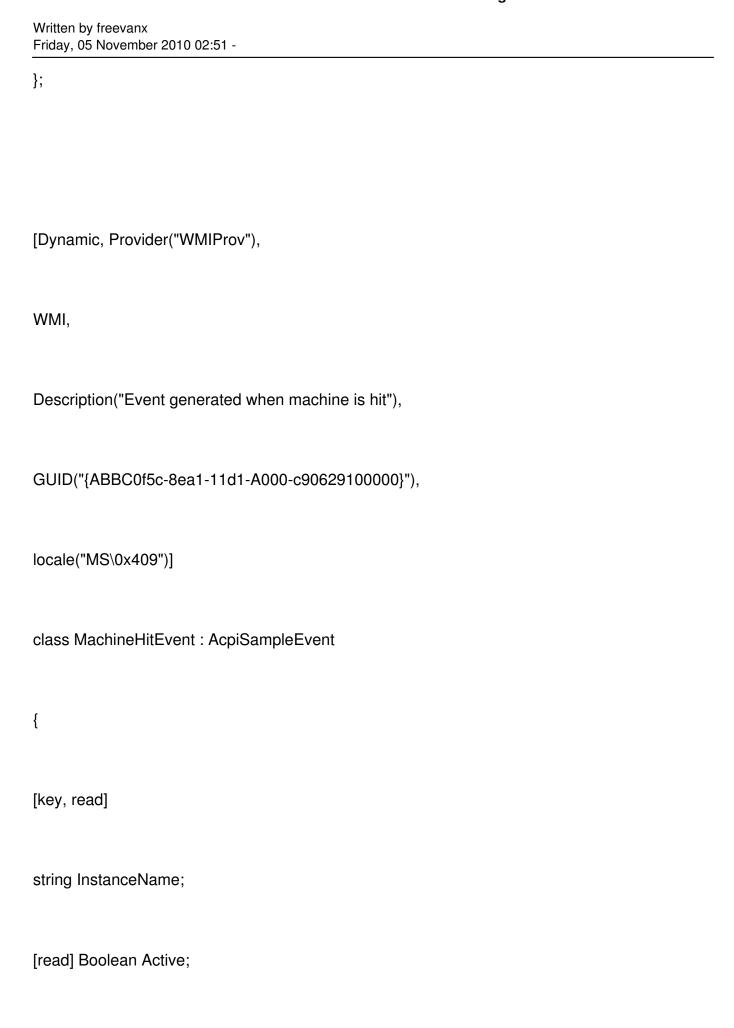
0xe0, 0x5c, 0x94, 0x81, 0x11, 0xee, 0xe3, 0x0f, 0xf8, 0x0f, 0xcb, 0x70, 0xfe, 0xff

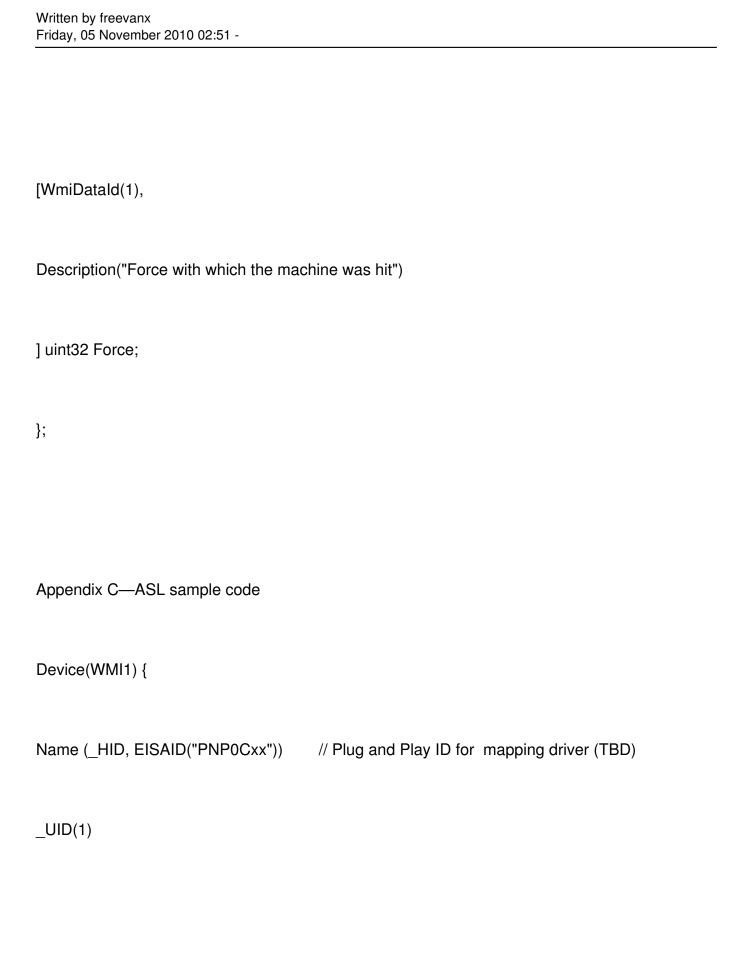








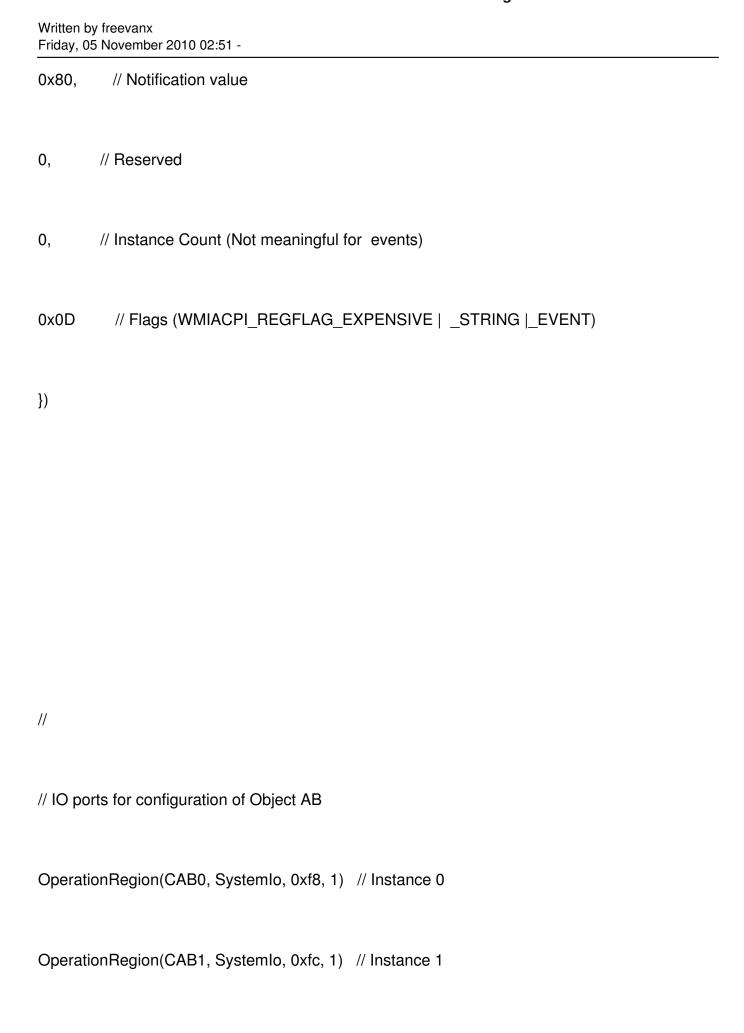


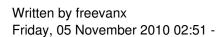


Written by freevanx

```
Friday, 05 November 2010 02:51 -
//
// Data block and Wmi method to Object ID mappings
Name( WDG, Buffer() {
// Object AA - {ABBC0F5A-8EA1-11d1-A53F-00A0C9062910}
0xABBC0F5A, 0x8ea1, 0x11d1, 0x00, 0xa0, 0xc9, 0x06, 0x29, 0x10,
`A','A',
         // Object ID
1,
        // Instance Count
0x04,
          // Flags (WMIACPI_REGFLAG_STRING)
// Object AB - {ABBC0F5B-8EA1-11d1-A53F-00A0C9062910}
0xABBC0F5B, 0x8ea1, 0x11d1, 0x00, 0xa0, 0xc9, 0x06, 0x29, 0x10,
`A','B', // Object ID
```

```
2,
       // Instance Count
0x01,
        // Flag (WMIACPI REGFLAG EXPENSIVE)
// Object AC - {ABBC0F5C-8EA1-11d1-A53F-00A0C9062910}
0xABBC0F5C, 0x8ea1, 0x11d1, 0x00, 0xa0, 0xc9, 0x06, 0x29, 0x10,
`A','C',
       // Object ID
1,
       // Instance Count
0x06,
        // Flag (WMIACPI REGFLAG METHOD | STRING)
// Event 0x80 - {ABBC0F5D-8EA1-11d1-A53F-00A0C9062910}
0xABBC0F5D, 0x8ea1, 0x11d1, 0x00, 0xa0, 0xc9, 0x06, 0x29, 0x10,
```



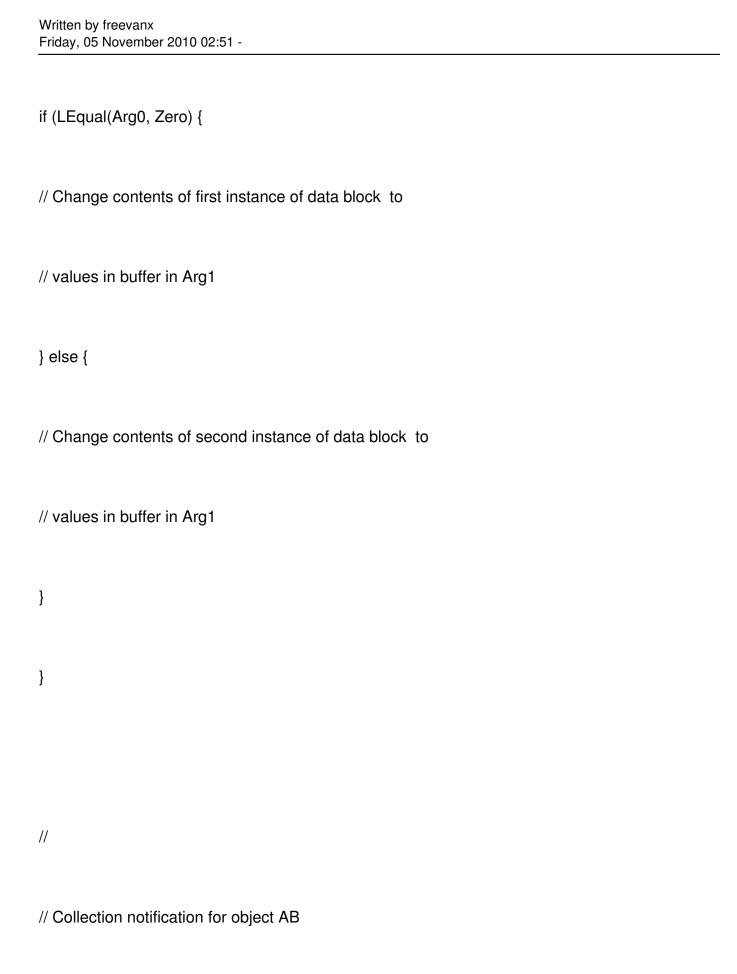


OperationRegion(CABC, Systemlo, 0xf4, 1) // Enable/Disable Collection

```
Method(WQAB, 1) {
//
// Read value from IO space for instance
if (LEqual(Arg0, Zero) {
Store(CAB0, Local0)
} else {
Store(CAB1, Local0)
}
```

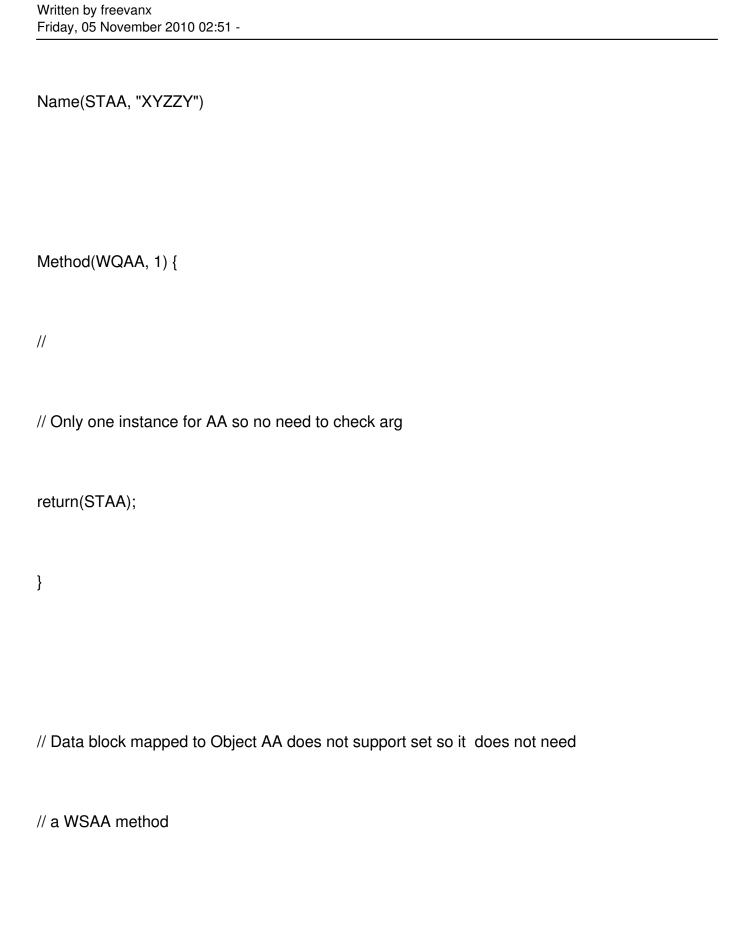
Written by freevanx

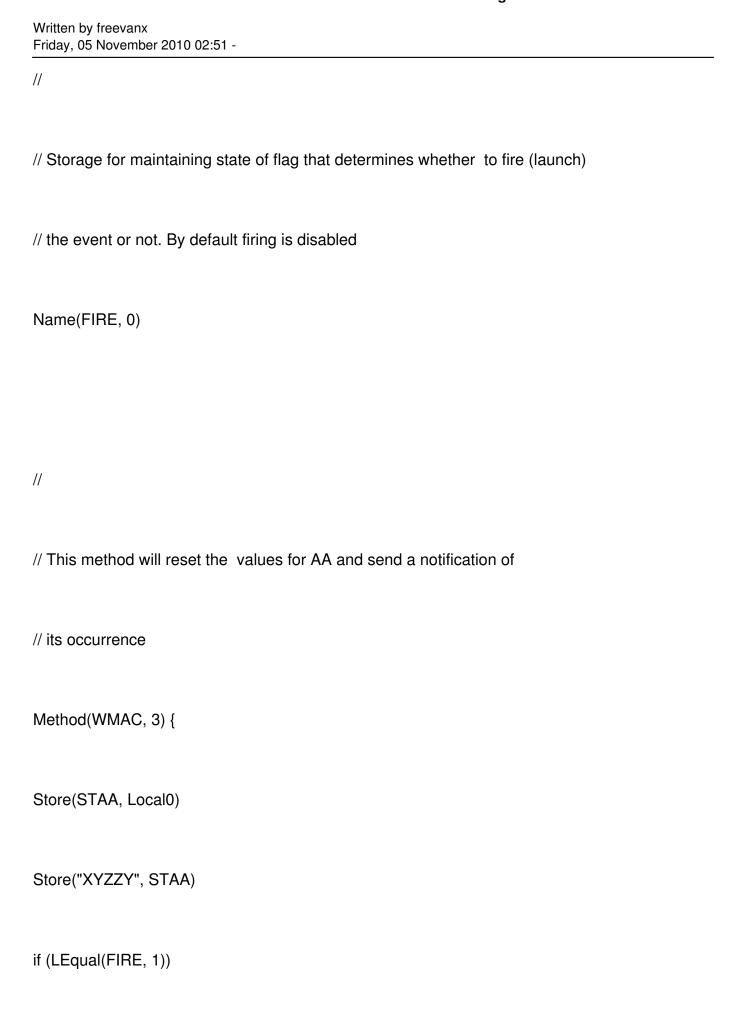
Friday, 05 November 2010 02:51 -// // If any of the lower 3 bits are set then return TRUE, else FALSE if (And(Local0, 7)) Return(0x00000001) } else { Return(0x00000000) } } // // Set the values for object AB Method(WSAB, 2) {

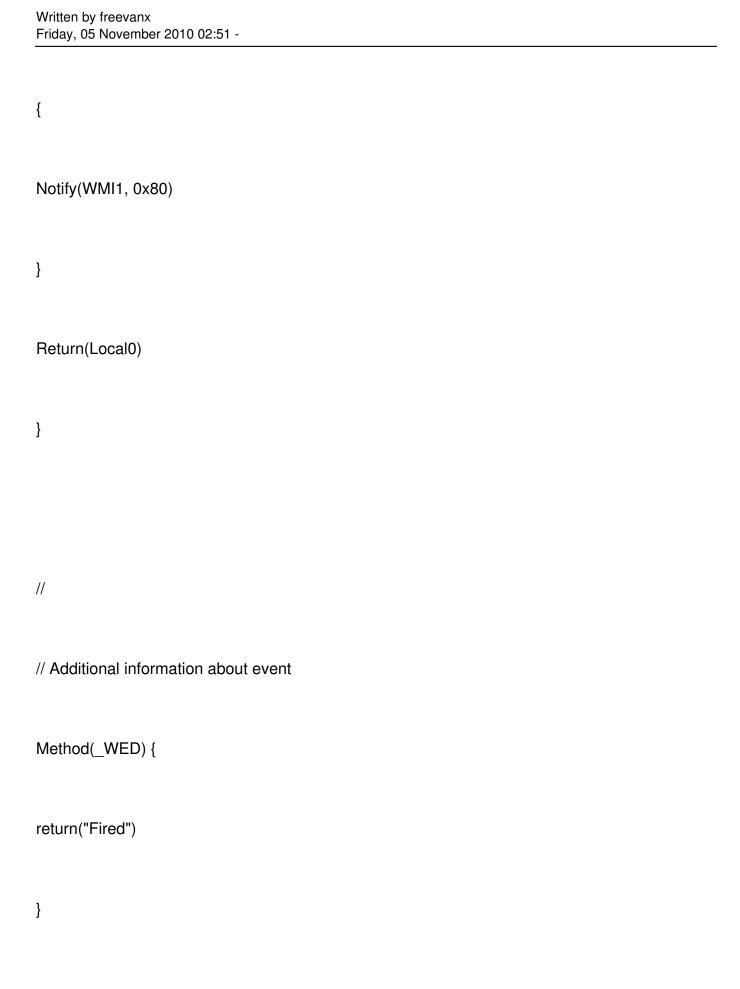


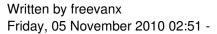
Written by freevanx Friday, 05 November 2010 02:51 -

```
Method(WCAB, 1) {
if (LEqual(Arg0, 1)
{
Store(One, CABC)
                      // If enable, write all 1's to port
} else {
Store(Zero, CABC) // If disable, write all 0's to port
}
}
//
// Storage for maintaining values for the AA method.
```









```
//
// Event 0x80 Enable/Disable control method

Method(WE80, 1) {

Store(FIRE, Arg0)

}
```

Joomla SEO powered by JoomSEF