



Broadcom Management API for Linux

Revision 17.0.0 • Date Aug. 21, 2014

Prepared by: Stephen Hurd

Copyright © 1999-2012 Broadcom Corporation
All Rights Reserved

No part of this document may be reproduced, in any form or by any means, without permission in writing from Broadcom Corporation.

Broadcom Corporation reserves the right to make changes to the products or information contained in this document without notice. No liability is assumed as a result of their use or application. No rights under any patent accompany the sale of any such products or information.

NetExtreme and NetX are trademarks of Broadcom Corporation.

Broadcom Corporation
5300 California Avenue
Irvine CA 92617
www.broadcom.com

CONFIDENTIAL

1. REVISION HISTORY	6
2. INTRODUCTION	6
3. BLOCK DIAGRAM	7
<i>BRCM</i>	7
<i>Linux & 3rd party</i>	7
4. IMPLEMENTATION	8
5. API REFERENCE	9
5.1 BmapiGetVersion	9
5.2 BmapiInitialize	10
5.3 BmapiUninitialize	10
5.4 BmapiGetNumPhyNic	11
5.5 BmapiGetAllPhyNic	12
5.6 BmapiDoNicIOCTL	13
5.7 BmapiGetServiceName	14
5.8 BmapiGetBRCMNicStatistics	15
5.9 BmapiEnableDevice	16
5.10 BmapiInitializeEx	17
5.11 BmapiIsInitialized	18
5.12 BmapiGetBRCMNicParam	18
5.13 BmapiSetBRCMNicParam	18
5.14 BmapiGetMultiBRCMNicParams	18
5.15 BmapiSetMultiBRCMNicParams	18
5.16 BmapiRefreshData	19
5.17 BmapiGetHandleByServiceName	19
5.18 BmapiGetNicStatistics64Ex	20
5.19 BmapiGetNicPciInfo	21
5.20 BmapiInitDiag	22
5.21 BmapiUnInitDiag	23
5.22 BmapiGetNumPhyNicEx	23
5.23 BmapiGetAllPhyNicHandles	24
5.24 BmapiGetPhyNic	25
5.25 BmapiGetASFTTable	26
5.26 BmapiSetASFTTable	27
5.27 BmapiGetBIOS	28
5.28 BmapiTestControlRegistersEx	29
5.29 BmapiTestMIIRegistersEx	30
5.30 BmapiTestEEPROMEx	31
5.31 BmapiTestInternalMemoryEx	32
5.32 BmapiTestInterruptEx	33
5.33 BmapiTestLoopBackEx	34
5.34 BmapiTestCPUEx	35
5.35 BmapiTestLEDsEx	36
5.36 BmapiSuspendDriverEx	37
5.37 BmapiResumeDriverEx	38
5.38 BmapiGetFirmwareInfo	39

5.39	BmapiWriteFirmware.....	40
5.40	BmapiReadFirmware	41
5.41	BmapiGetSystemASFTables.....	42
5.42	BmapiReadNicMem.....	43
5.43	BmapiWriteNicMem.....	44
5.44	BmapiGetIpAddrInfo	45
5.45	BmapiGetBRCMNicInfoEx	46
5.46	BmapiWriteFirmwareInfo.....	47
5.47	BmapiRetrieveLinkStatusEx.....	48
5.48	BmapiGetBRCMNicStatisticsEx	49
5.49	BmapiGetBrcmNicParamList	50
5.50	BmapiGetBrcmNicParamInfo	51
5.51	BmapiSetBrcmNicParam2	53
5.52	BmapiGetLicenseKey	53
5.53	BmapiSetLicenseKey	53
5.54	BmapiGet5706FwInfo.....	53
5.55	BmapiGet57710FwInfo.....	53
5.56	BmapiGetMgmtProcessors.....	54
5.57	BmapiGetMgmtEnableState	55
5.58	BmapiSetMgmtEnableState	56
5.59	BmapiAssertMgmtEvent.....	57
5.60	BmapiGetMgmtOTPKeys.....	58
5.61	BmapiGetMgmtDataLength.....	59
5.62	BmapiGetMgmtData	60
5.63	BmapiSetMgmtData.....	61
5.64	BmapiGetMgmtConfigLength.....	62
5.65	BmapiGetMgmtConfig.....	63
5.66	BmapiSetMgmtConfig	64
5.67	BmapiGetMgmtSharedMem	65
5.68	BmapiWritePhyFirmware	65
5.69	BmapiCreateMgmtData	66
5.70	BmapiGetMgmtWebDataLength	67
5.71	BmapiGetMgmtWebData.....	68
5.72	BmapiSetMgmtWebData	69
5.73	BmapiCreateMgmtWebData	70
5.74	BmapiRetrieveMultiLinkStatus	71
5.75	BmapiGetISCSIRuntimeIPCount.....	71
5.76	BmapiGetISCSIRuntimeIP	71
5.77	BmapiGetISCSIRuntimeStatistics	71
5.78	BmapiGetISCSISessionStatistics	71
5.79	BmapiWriteFirmware2.....	72
5.80	BmapiReadFirmware2	73
5.81	BmapiGetNicStatisticsV3	74
5.82	BmapiGetLldpParams	75
5.83	BmapiGetDcbxParams	76
5.84	BmapiGetIscsiCfg	77

5.85	BmapiSetIscsiCfg.....	78
5.86	BmapiGetFcoeCfg.....	79
5.87	BmapiSetFcoeCfg	80
5.88	BmapiGetMBAParams.....	81
5.89	BmapiSetMBAParams	81
5.90	BmapiGetNicPartCfg	81
5.91	BmapiSetNicPartCfg.....	82
5.92	BmapiGetExtVPD.....	82
5.93	BmapiSetExtVPD.....	83
5.94	BmapiGetDcbNvramCfg.....	83
5.95	BmapiSetDcbNvramCfg	83
5.96	BmapiGetNivCfg	83
5.97	BmapiSetNivCfg.....	83
5.98	BmapiGetNivPortProfile.....	83
5.99	BmapiGetFLRCfg.....	84
5.100	BmapiSetFLRCfg.....	84
5.101	BmapiGetSRIOVforSF	84
5.102	BmapiSetSRIOVforSF	84
5.103	BmapiGetSRIOVSwitchInfo.....	84
5.104	BmapiGetSRIOVSwitchStats.....	84
5.105	BmapiGetSRIOVVFStats	84
5.106	BmapiGetSRIOVVFInfo.....	84
6.	QUICK PROGRAMMING GUIDE.....	85
6.1	How to get physical network adapter information?	85
6.2	How to get handle to a physical network adapter?.....	85
6.3	How to do diagnostics?	85
6.4	How to do ASF?.....	85
6.5	How to tell which type of NIC it is?	85

1. Revision History

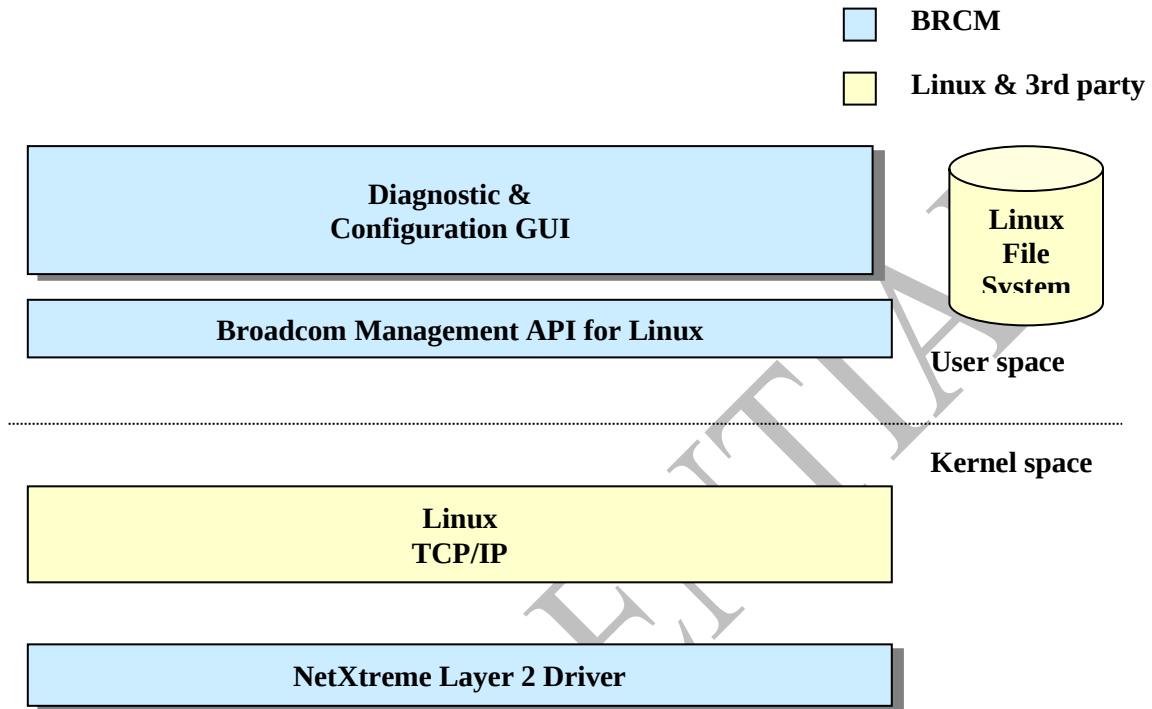
Version	Date	Author	Comments
6.13.1	03/20/12	Bing Liu	Initial draft for Linux & VMWare, based on Windows BMAPI specs
6.17.50	10/04/12	Jason Hsu	Updated BMAPILNX document based on bmapilnx, v6.17.50 release. Updated information in Section 2, 5.10, and 6.3.
6.18.0	10/12/12	Jason Hsu	Updated version and date.
6.23.1	10/03/13	Sujatha Kattam Reddy	Added new APIs, BmapiGetSRIOVforSF, BmapiSetSRIOVforSF, BmapiGetSRIOVSwitchInfo, BmapiGetSRIOVSwitchStats, BmapiGetSRIOVVfStats, BmapiGetSRIOVVfInfo, with version and date update.
17.0.0	08/21/14	Stephen Hurd	QLogic-supported controllers (ie: NX2) are no longer supported. Formatting updates.

2. Introduction

The purposes to have Broadcom Management API (BMAPI) for Linux, BMAPILNX are

- Help applications to view status, statistics and configuration on network Interface Cards (NIC).
- Help applications to configure and diagnose Broadcom made NIC and configuration related to Broadcom products.
- Help applications not to include driver specific or implementation related knowledge.
- Help applications to immune from the change of driver's implementation.
- Help applications to work with all Broadcom made NIC products with same API set.

3. Block Diagram



4. Implementation

Broadcom Management API will be implemented in ANSI C as much as possible to make the module more portable. Current implementation supports x86 and x86_64 architectures on the following platforms:

RedHat Enterprise Linux 5

RedHat Enterprise Linux 6

SuSE Linux Enterprise Server 11

VMWare ESX/ESXi

When applications start, applications should:

1. Call BmapiGetVersion() to make sure it is the correct version of BMAPI.
2. Call BmapiInitializeEx() to initialize BMAPI internal data.

When applications end, applications should call BmapiUninitialize() to release internal resources in BMAPI if applications had successfully called BmapiInitialize() or BmapiInitializeEx().

CONFIDENTIAL

5. API reference

5.1 BmapiGetVersion

Get version number of BMAPI.

```
void BmapiGetVersion(  
    OUT U32 *pMajorVersion,  
    OUT U32 *pMinorVersion,  
    OUT U32 *pBuildVersion );
```

Parameters:

pMajorVersion

Pointer to major version number.

pMinorVersion

Pointer to minor version number.

pBuildVersion

Pointer to build version number.

Return Value:

No return value.

Supported Network Adapters:

Not applicable

Remarks:

Applications can call this function to verify that it is using the correct version of BMAPI. Applications can call this function at any time i.e. it can be called before BMAPI is initialized.

5.2 BmapiInitialize

Initialize internal data and read NIC configuration from system.

```
U32 BmapiInitialize( void );
```

Parameters:

This function does not have parameters.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

Applications **MUST** call this function before calling any other functions except BmapiGetVersion(). Applications only need to call once. However, application can call more than once without getting error. When applications are finished, applications **MUST** call BmapiUninitialize().

The calling thread of BmapiUninitialize() **MUST** be the thread called BmapiInitialize().

Multithread applications should call BmapiInitializeEx().

5.3 BmapiUninitialize

Release allocated resources.

```
U32 BmapiUninitialize( void );
```

Parameters:

This function does not have parameters.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

Applications **MUST** call BmapiUninitialize() before exiting, otherwise, application may have memory leak. An application need to call BmapiUninitialize() as many times as it calls BmapiInitialize() or BmapiInitializeEx().

The calling thread of BmapiUninitialize() **MUST** be the thread called BmapiInitialize().

5.4 BmapiGetNumPhyNic

The BmapiGetNumPhyNic will return the number of physical network adapters that are installed with drivers in the system.

```
U32 BmapiGetNumPhyNic(  
    OUT U32 *pNumOfPhy );
```

Parameters:

pNumOfPhy

Pointer to the number of physical network adapters.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including 4401, 57xx and third party NICs

Remarks:

Applications call this function to understand how many physical network adapters are installed with driver in the system, allocate enough buffers to fill all adapters' data and call BmapiGetAllPhyNic() to retrieve all data.

5.5 BmapiGetAllPhyNic

The BmapiGetAllPhyNic will return information for all physical network adapters.

```
U32 BmapiGetAllPhyNic(  
    OUT BM_ADAPTER_INFO *pPhyList,  
    IN U32 uNumOfPhy );
```

Parameters:

pPhyList

Pointer to the array of BM_ADAPTER_INFO for all physical network adapters.

uNumOfPhy

Number of BM_ADAPTER_INFO structure allocated for *pPhyList* array. This number must be more than the number of physical network adapters in the system. If not, the function will return error code BMAPI_BUFSHORT.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including 4401, 57xx and third party NICs

Remarks:

Applications must call BmapiGetNumPhyNic() to understand how many NICs are in system first. Applications, then, allocate enough buffers to fill all adapters' data and call BmapiGetAllPhyNic() to retrieve data. If uNumOfPhy is not enough, the function will return error code.

5.6 BmapiDoNicIOCTL

The BmapiDoNicIOCTL will issue IOCTL command to the driver that the ‘handle’ belongs to.

```
U32 BmapiDoNicIOCTL(  
    IN U32 handle,  
    IN U32 ioctlCode,  
    IN void *pInBuf,  
    IN U32 inBufSize,  
    IN/OUT void *pOutBuf,  
    IN/OUT U32 *pOutBufSize,  
    OUT U32 *pResult );
```

Parameters:

handle

Handle to an adapter. (can be found in BM_ADAPTER_INFO structure)

ioctlCode

IOCTL control code.

pInBuf

Pointer to an input buffer that will pass to driver.

inBufSize

Size of the input buffer pointed *pInBuf*.

pOutBuf

Pointer to an output buffer that will pass to driver and filled by driver.

pOutBufSize

On input, it is the size of the output buffer pointed *pOutBuf*. On output, it is the amount of data stored in *pOutBuf*.

pResult

Applications must pass a pointer to a U32 variable. The variable will store the result of GetLastError().

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including 4401, 57xx and third party NICs

Remarks:

Applications could call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the ‘handle’ in BM_ADAPTER_INFO. Applications should use the ‘handle’ to call BmapiDoNicIOCTL(). If the driver is not running at the time application calls BmapiDoNicIOCTL(), BmapiDoNicIOCTL() will return error code.

5.7 BmapiGetServiceName

The BmapiGetServiceName will return the service name corresponding to the adapter's handle.

```
U32 BmapiGetServiceName (
    IN U32 handle,
    OUT U8 *pServiceName,
    IN U32 uBufLen );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pServiceName

Pointer to the buffer that will be filled with the adapter's service name on function's return.

uBufLen

Length of buffer pointed by 'pServiceName'.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including 4401, 57xx and third party NICs. All BASP created virtual adapters.

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.8 BmapiGetBRCMNicStatistics

The BmapiGetBRCMNicStatistics will return Broadcom network adapters proprietary statistics data.

```
U32 BmapiGetBRCMNicStatistics(  
    IN U32 handle,  
    IN/OUT BM_BRCM_STATISTICS *pBrcmStatistics );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pBrcmStatistics

Pointer to a BM_BRCM_STATISTICS structure for Broadcom network adapter proprietary statistics information. The data will be filled out on return.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications should allocate buffers for the statistics structures and pass pointers to those structures to the function. In case of NIC is disabled, the return code will be BMAPI_DRIVER_NOT_LOADED.

5.9 BmapiEnableDevice

BmapiEnableDevice will enable or disable a network adapter.

```
U32 BmapiEnableDevice(  
    IN U32 handle,  
    IN U32 uEnable );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uEnable

Non-zero value to enable the device, zero value to disable the device.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink, NetXtreme I and third party NICs. All BASP created virtual adapters.

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.10 BmapiInitializeEx

Initialization function of BMAPILNX.

```
U32 BmapiInitializeEx(  
    IN U32 bKeepCom );
```

Parameters:

bKeepCom

Keep COM initialized or not.

This parameter is not applicable to BMAPILNX and will be ignored.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

Applications **MUST** call this function before calling any other functions except BmapiGetVersion(). Applications only need to call once. However, application can call more than once without getting error. After calling BmapiInitializeEx(), applications do not need to call BmapiInitialize().

Applications **MUST** call BmapiUninitialize() before exiting, otherwise, application may have memory leak.

Applications that call BmapiInitializeEx() MUST use the same thread to call BmapiUninitialize().

5.11 BmapiIsInitialized

Applications use the function to determine BMAPI is initialized or not.

```
U32 BmapiIsInitialized()
```

Parameters:

none

Return Value:

Return BMAPI_OK if BMAPI is initialized; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

Applications can call this function without call BmapiInitialize() or BmapiInitializeEx(). This function will work well with single thread applications or applications that can control the calls to BmapiInitialize()/BmapiInitializeEx(). However, if the application is multiple threaded and the API calls to BmapiInitialize()/BmapiInitializeEx() are scattered and not controllable, the function will not guarantee correct result.

5.12 BmapiGetBRCMNicParam

The API is obsolete.

5.13 BmapiSetBRCMNicParam

The API is obsolete.

5.14 BmapiGetMultiBRCMNicParams

The API is obsolete.

5.15 BmapiSetMultiBRCMNicParams

The API is obsolete.

5.16 BmapiRefreshData

BmapiRefreshData() will force BMAPI to refresh its internal data.

```
U32 BmapiRefreshData();
```

Parameters:

none

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

For multithreaded applications, once a thread calls the function, all other threads will see the change of data.

5.17 BmapiGetHandleByServiceName

The BmapiGetHandleByServiceName will return the adapter's handle corresponding to the adapter's service name.

```
U32 BmapiGetHandleByServiceName(  
    IN U8 *pServiceName,  
    OUT U32 *handle );
```

Parameters:

pServiceName

Pointer to the ASCII NULL terminated service name.

handle

Pointer to a handle for an adapter.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink, NetXtreme I and third party NICs. All BASP created virtual adapters.

Remarks:

Make sure BMAPI is initialized before calling the function. If the NIC is not found, return code is BMAPI_NIC_NOT_FOUND.

5.18 BmapiGetNicStatistics64Ex

The BmapiGetNicStatistics64 will return all 64-bit statistics information for a network adapter.

```
U32 BmapiGetNicStatistics64Ex(  
    IN U32 handle,  
    OUT BM_GENERAL_STATISTICS64 *pGenStatistics,  
    OUT BM_ETHERNET_STATISTICS64 *pEthStatistics );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pGenStatistics

Pointer to a BM_GENERAL_STATISTICS64 structure for general statistics information of a network adapter. The data will be filled out on return.

pEthStatistics

Pointer to a BM_ETHERNET_STATISTICS64 structure for ethernet statistics information of a network adapter. The data will be filled out on return.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink and NetXtreme I

Remarks:

Applications should allocate buffers for the statistics structures and pass pointers to those structures to the function. Application should be aware that some information might not be available. In case of NIC is disabled, the return code will be BMAPI_DRIVER_NOT_LOADED.

5.19 BmapiGetNicPciInfo

The BmapiGetNicPciInfo will return PCI information of the NIC.

```
U32 BmapiGetNicPciInfo(  
    IN U32 handle,  
    OUT BM_NIC_PCI_INFO *pNicPciInfo );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pNicPciInfo

Pointer to the buffer that will be filled with the adapter's PCI information.
'version' field MUST be filled on input.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink, NetXtremeI and third party NICs.

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information.
Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function. If PCI IDs can not be determined for the NIC (for example, NIC is removed from the system under NT 4.0), PCI IDs will be 0xffff.
Applications **MUST** set 'version' field in BM_NIC_PCI_INFO structure prior to call the function.

5.20 BmapiInitDiag

The BmapiInitDiag will lock the access to a NIC during diagnostic so that no multiple diagnostics can perform on the same NIC concurrently.

```
U32 BmapiInitDiag(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information.

Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Applications **MUST** call this API prior to do any NIC diagnostic test including cable analysis. Otherwise, the test will fail. Applications also **MUST** call BmapiUnInitDiag() when all diagnostic tasks are done.

5.21 BmapiUnInitDiag

The BmapiUnInitDiag unlock the access to a NIC locked by BmapiInitDiag() API.

```
U32 BmapiUnInitDiag(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.22 BmapiGetNumPhyNicEx

The function will return the number of physical network adapters in the system. In Windows 2000 or above, the number includes network adapters sitting in PCI slots but not installed with drivers.

```
U32 BmapiGetNumPhyNicEx(  
    IN U32 *pNumOfPhy );
```

Parameters:

pNumOfPhy

Pointer to the number of physical network adapters.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink, NetXtremeI and third party NICs.

Remarks:

Applications call BmapiGetNumPhyNicEx() to understand how many physical network adapters are in the system, allocate enough buffers to fill all adapters' handles, call BmapiGetAllPhyNicHandles() to retrieve all handles and call BmapiGetPhyNic() to retrieve a specific network adapter's info.

5.23 BmapiGetAllPhyNicHandles

The function will return handles for all physical network adapters.

```
U32 BmapiGetAllPhyNicHandles (  
    OUT U32 *pHandleList,  
    IN U32 uNumOfPhy );
```

Parameters:

pHandleList

Pointer to the array of physical network adapter handles.

uNumOfPhy

Number of handles allocated in *pHandleList* array.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink, NetXtremeI and third party NICs.

Remarks:

Applications call BmapiGetNumPhyNicEx() to understand how many physical network adapters are in the system, allocate enough buffers to fill all adapters' handles, call BmapiGetAllPhyNicHandles() to retrieve all handles and call BmapiGetPhyNic() to retrieve a specific network adapter's info.

5.24 BmapiGetPhyNic

The function will return the physical network adapter information according the handle.

```
U32 BmapiGetPhyNic(  
    IN U32 handle,  
    OUT BM_ADAPTER_INFO_EX *pNicInfoEx );
```

Parameters:

handle

Handle to the NIC information that should be retrieved.

pNicInfoEx

Pointer to the buffer for physical network adapter information.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink and NetXtremeI and third party NICs.

Remarks:

Applications call BmapiGetNumPhyNicEx() to understand how many physical network adapters are in the system, allocate enough buffers to fill all adapters' handles, call BmapiGetAllPhyNicHandles() to retrieve all handles and call BmapiGetPhyNic() to retrieve a specific network adapter's info. Make sure set the 'version' in BM_ADAPTER_INFO_EX to BMAPI_ADAPTER_INFO_EX_VER before calling the function.

5.25 BmapiGetASFTable

BmapiGetASFTable will read ASF table from a Broadcom made ASF capable network adapter.

```
U32 BmapiGetASFTable(  
    IN U32 handle,  
    OUT BM_ASF_TABLE *pNicASF );
```

Parameters:

handle

Handle to an adapter.

pNicASF

Application allocated buffer for ASF table.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetXtreme I (exclude 5700 and 5701) with ASF firmware

Remarks:

Applications must call BmapiGetAllPhyNic(), BmapiGetAllPhyNicHandles etc. to get adapter handle.

Make sure set the 'version' in BM_ASF_TABLE to BMAPI_ASF_T_VERSION before calling the function.

If the application had called BmapiInitDiag() prior to call this API, BmapiInitDiag() must be called from the same thread that calls this API.

The function can be used to read configuration for both ASF and IPMI. If the management firmware is IPMI, the 'flags' defined in BM_ASF_TABLE will have BM_IPMI_SUPPORT set. The same 'AsfGui.cfg.Config.EnableASF' flag can be used to determine whether the firmware is enabled or not.

The function supports UMP firmware. If the firmware is UMP, the 'flags' defined in BM_ASF_TABLE will have BM_UMP_SUPPORT set. All ASF and IPMI configuration information will not be populated for UMP. Please refer to BM_ASF_TABLE in BMAPI.h for detail information.

5.26 BmapiSetASFTable

BmapiSetASFTable will write ASF table to a Broadcom made ASF capable network adapter.

```
U32 BmapiGetASFTable(  
    IN U32 handle,  
    OUT BM_ASF_TABLE *pNicASF );
```

Parameters:

handle

Handle to an adapter.

pNicASF

ASF table to write to the network adapter.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetXtreme I (exclude 5700 and 5701) with ASF firmware

Remarks:

Applications must call BmapiGetAllPhyNic(), BmapiGetAllPhyNicHandles etc. to get adapter handle.

This API will suspend and resume driver at the end of NVRAM access.

Make sure set the 'version' in BM_ASF_TABLE to BMAPI_ASF_T_VERSION before calling the function.

If the application had called BmapiInitDiag() prior to call this API, BmapiInitDiag() must be called from the same thread that calls this API.

5.27 BmapiGetBIOS

BmapiGetBIOS will read BIOS information from the 'stat_addr' for 'Length' bytes long.

```
U32 BmapiGetBIOS(  
    IN U64 start_addr,  
    OUT void *buffer,  
    IN U32 length );
```

Parameters:

start_addr

Starting address of physical memory to read.

buffer

Application allocated buffer to store data read from the physical memory.

length

bytes to read and length of *buffer*

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

Make sure BMAPI is initialized before calling this function..

5.28 BmapiTestControlRegistersEx

The BmapiTestControlRegistersEx will return test result of control registers for Broadcom made network adapters.

```
U32 BmapiTestControlRegistersEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() and BmapiSuspendDriverEx() to initialize diagnostic setup and suspend the NIC.

After the function returned, call BmapiResumeDriverEx() and BmapiUnInitDiag() to resume the NIC traffic and terminate diagnostic setup

5.29 BmapiTestMIRegistersEx

The BmapiTestMIRegistersEx will return test result of MII control registers for Broadcom made network adapters.

```
U32 BmapiTestMIRegistersEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() and BmapiSuspendDriverEx() to initialize diagnostic setup and suspend the NIC. After the function returned, call BmapiResumeDriverEx() and BmapiUnInitDiag() to resume the NIC traffic and terminate diagnostic setup.

5.30 BmapiTestEEPROMEx

The BmapiTestEEPROMEx will return test result of EEPROM for Broadcom made network adapters.

```
U32 BmapiTestEEPROMEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() to initialize diagnostic setup. After the function returned, call BmapiUnInitDiag() to terminate diagnostic setup.

5.31 BmapiTestInternalMemoryEx

The BmapiTestInternalMemoryEx will test internal memory and return result for Broadcom made network adapters.

```
U32 BmapiTestInternalMemoryEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() and BmapiSuspendDriverEx() to initialize diagnostic setup and suspend the NIC. After the function returned, call BmapiResumeDriverEx() and BmapiUnInitDiag() to resume the NIC traffic and terminate diagnostic setup.

5.32 BmapiTestInterruptEx

The BmapiTestInterruptEx will test internal memory and return result for Broadcom made network adapters.

```
U32 BmapiTestInterruptEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() and BmapiSuspendDriverEx() to initialize diagnostic setup and suspend the NIC. After the function returned, call BmapiResumeDriverEx() and BmapiUnInitDiag() to resume the NIC traffic and terminate diagnostic setup.

5.33 BmapiTestLoopBackEx

The BmapiTestLoopBackEx will perform loopback test and return test result for Broadcom made network adapters.

```
U32 BmapiTestLoopBackEx(  
    IN U32 handle,  
    IN ULONG Lbtype );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Lbtype

Loopback type to perform. Can be BMAPI_LOOPBACK_TYPE_MAC or BMAPI_LOOPBACK_TYPE_PHY.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() to initialize diagnostic setup. After the function returned, call BmapiUnInitDiag() to terminate diagnostic setup.

For 4401 and 57xx NICs, before calling this function, applications MUST make sure miniport driver is NOT in suspend mode. If it does, applications need to call BmapiResumeDriver() to get miniport driver running.

For 5706 family, before calling the function, make sure application had called BmapiSuspendDriverEx() to initialize diagnostic setup and suspend the NIC.

BMAPI_LOOPBACK_TYPE_EXTERNAL is supported for 570x based NICs only.

5.34 BmapiTestCPUEx

The BmapiTestCPUEx will enable test mode and test the on chip processor for Broadcom made network adapters. This test is not supported in BMAPILNX.

```
U32 BmapiTestCPUEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Always return BMAPI_NOT_SUPPORTED_NIC.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() and BmapiSuspendDriverEx() to initialize diagnostic setup and suspend the NIC.

After the function returned, call BmapiResumeDriverEx() and BmapiUnInitDiag() to resume the NIC traffic and terminate diagnostic setup.

4401 does not have CPU. CPU test simply return BMAPI_OK.

5.35 BmapiTestLEDsEx

The BmapiTestLEDsEx will flip the LEDs to indicate which logical NIC card is bound to physical NIC card for Broadcom made network adapters.

```
U32 BmapiTestLEDsEx(  
    IN U32 handle,  
    IN ULONG BlinkDurationSec );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

BlinkDurationSec

Specify how long to flip the LEDs (in seconds, up to 120 seconds).

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink, NetXtreme I, NetXtreme II 5706 family

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information.

Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() to initialize diagnostic setup. After the function returned, call BmapiUnInitDiag() to terminate diagnostic setup.

For 4401, applications must call BmapiSuspendDriverEx() prior to call the LED test.

5.36 BmapiSuspendDriverEx

BmapiSuspendDriver will shutdown APE if applicable to the NIC.

```
U32 BmapiSuspendDriverEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() to initialize diagnostic setup. After the function returned, call BmapiUnInitDiag() to terminate diagnostic setup.

5.37 BmapiResumeDriverEx

BmapiResumeDriver will restart APE if applicable to the NIC.

```
U32 BmapiResumeDriverEx(  
    IN U32 handle );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Before calling the function, make sure application had called BmapiInitDiag() to initialize diagnostic setup. After the function returned, call BmapiUnInitDiag() to terminate diagnostic setup.

5.38 BmapiGetFirmwareInfo

BmapiGetFirmwareInfo will get firmware information.

```
U32 BmapiGetFirmwareInfo(  
    IN U32 handle,  
    OUT BM_FW_INFO *pFWInfo );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pFWInfo

Pointer to the firmware information buffer.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink (exclude 4401)

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information.

Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

Make sure set the 'version' in BM_FW_INFO to BMAPI_FW_INFO_VER before calling the function.

This API will go in diagnostics mode during execution if the application had not called BmapiInitDiag() prior to call this API.

5.39 BmapiWriteFirmware

BmapiWriteFirmware will write firmware information starting at specified offset with provided data buffer and lenngth to write to.

```
U32 BmapiWriteFirmware(  
    IN U32 handle,  
    IN U32 uOffset,  
    IN U32 *pDataBuf,  
    IN U32 uBufLen,  
    IN U8 *pChk );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uOffset

Starting offset to write data to.

pDataBuf

Data to write to firmware. Data are in 32-bit array.

uBufLen

Number of elements in data buffer array. For example, 2 means 2 32-bit data in the data buffer.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink (exclude 4401) and NetXtreme I

Remarks:

1. Applications need to call BmapiInitDiag() prior to call this and call BmapiUnInitDiag() after everything is done.
2. The newly programmed firmware will not be effective till:
 - The machine reboot.
 - The driver is disabled and re-enabled. Can be done manually or through BmapiEnableDevice().
 - Call BmapiSuspendDriverEx() followed by BmapiResumeDriverEx().
This is the most preferable method.

5.40 BmapiReadFirmware

BmapiReadFirmware will read firmware information starting at specified offset with provided data buffer and length to read.

```
U32 BmapiReadFirmware(  
    IN U32 handle,  
    IN U32 uOffset,  
    IN U32 *pDataBuf,  
    IN U32 uBufLen,  
    IN U8 *pChk );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uOffset

Starting offset to read data from.

pDataBuf

Data buffer from firmware data. Data are in 32-bit array.

uBufLen

Number of elements in data buffer array. For example, 2 means 2 32-bit data in the data buffer.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink (exclude 4401), NetXtreme I

Remarks:

Applications need to call BmapiInitDiag() prior to call this and call BmapiUnInitDiag() after everything is done.

5.41 BmapiGetSystemASFTables

BmapiGetSystemASFTables will read ASF tables from system BIOS.

```
U32 BmapiGetSystemASFTables(  
    IN BM_ASF_TABLE *pAsfTbls );
```

Parameters:

pAsfTbls

Pointer to ASF table structure that will be filled.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

Not applicable

Remarks:

Make sure set the 'version' in BM_ASF_TABLE to BMAPI_ASF_T_VERSION before calling the function.

CONFIDENTIAL

5.42 BmapiReadNicMem

BmapiReadNicMem will read network adapter's registers and memory.

```
U32 BmapiReadNicMem(  
    IN U32 handle,  
    IN U32 uType,  
    IN U32 uOffset,  
    IN U32 *pData,  
    IN U8 *pChk );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uType

Type of register or memory access.

uOffset

Offset to read from.

pData

Pointer to the return data.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink (exclude 4401), NetXtreme I

Remarks:

1. The function always reads 4 bytes data at a time.
2. 'uType' could be BMAPI_INDIRECT_REG_READ, BMAPI_INDIRECT_MEM_READ or BMAPI_PHY_REG_READ for NetXtreme I NICs.
3. 'uType' could be BMAPI_REG_READ, BMAPI_MEM_READ or BMAPI_PHY_REG_READ for NetXtreme II NICs.
4. Value pointed by 'pData' is valid only when the function returns BMAPI_OK.

5.43 BmapiWriteNicMem

BmapiWriteNicMem will write network adapter's registers and memory.

```
U32 BmapiWriteNicMem(  
    IN U32 handle,  
    IN U32 uType,  
    IN U32 uOffset,  
    IN U32 uData,  
    IN U8 *pChk );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uType

Type of register or memory access.

uOffset

Offset to write to.

uData

Data to write to NIC.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink (exclude 4401), NetXtreme I

Remarks:

1. The function always reads 4 bytes data at a time.
2. 'uType' could be BMAPI_INDIRECT_REG_READ, BMAPI_INDIRECT_MEM_READ or BMAPI_PHY_REG_READ for NetXtreme I NIC.
3. 'uType' could be BMAPI_REG_READ, BMAPI_MEM_READ or BMAPI_PHY_REG_READ for NetXtreme II NIC.

5.44 BmapiGetIpAddrInfo

BmapiGetIpAddrInfo will read adapter's IP/NetMask/Gateway information.

```
U32 BmapiGetIpAddrInfo(  
    IN U32 handle,  
    IN U32 uType,  
    OUT U8 *pBuf,  
    IN OUT U32 *pLen );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uType

Type of information to retrieve. It could be BMAPI_IPINFO_IP_LIST, BMAPI_IPINFO_SUBNETMASK_LIST or BMAPI_IPINFO_GATEWAY_LIST.

pBuf

Buffer to for retrieved information.

pLen

Pointer to a U32 number. On input, it is the length of buffer. On return, it is the required buffer length.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All NDIS adapters including NetLink, NetXtreme and BASP created virtual adapters

Remarks:

1. On input, uLen will be the length of buffer.
2. If 'pBuf' is NULL, 'uLen' will be the required length of buffer.
3. If the function failed due to buffer too short, 'uLen' will be the required length of buffer.
4. Returned data in 'pBuf' is in REG_MULTI_SZ format. (An array of null-terminated strings, terminated by two null characters.)

5.45 BmapiGetBRCMNicInfoEx

BmapiGetBRCMNicInfoEx will retrieve proprietary Broadcom network adapter information.

```
U32 BmapiGetBRCMNicInfoEx(  
    IN U32 handle,  
    OUT BM_BRCM_ADAPTER_INFO_EX *pBRCMNicInfoEx );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pBRCMNicInfoEx

On input, applications pass a pointer to a BM_BRCM_ADAPTER_INFO_EX structure. On output, data will be filled out in the BM_BRCM_ADAPTER_INFO_EX structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
Not all fields are applicable to 4401.

5.46 BmapiWriteFirmwareInfo

BmapiWriteFirmwareInfo will update firmware information.

```
U32 BmapiWriteFirmwareInfo(  
    IN U32 handle,  
    IN BM_FW_INFO *pFWInfo,  
    IN U32 uOption );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pFWInfo

Pointer to BM_FW_INFO buffer will be copied to EEPROM.

uOption

Which block to update.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink (exclude 4401 and selfboot), NetXtreme I

Remarks:

Currently, the function only supports BMAPI_WR_FW_MANUFAC option to update manufactory data block.

This API will go in diagnostics mode during execution if the application had not called BmapiInitDiag() prior to call this API.

5.47 BmapiRetrieveLinkStatusEx

The BmapiRetrieveLinkStatusEx() will retrieve link status and link negotiation result for Broadcom made network adapters.

```
U32 BmapiRetrieveLinkStatusEx(  
    IN U32 handle,  
    OUT BM_LINK_STATUS_EX *pLinkStatusEx );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pLinkStatusEx

On input, applications pass a pointer to a BM_LINK_STATUS_EX structure. On output, data will be filled out in the BM_LINK_STATUS_EX structure.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

All physical NICs including NetLink, NetXtreme I and third party NICs. All BASP created virtual adapters.

Remarks:

Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

For all non-Broadcom made network adapters and BASP created virtual adapters, only "link_status", "line_speed_Kbps" and "driver_loaded" are available.

For 4401 network adapters, only "link_status", "duplex_mode", "link_speed" and fields for third party NICs are available.

5.48 BmapiGetBRCMNicStatisticsEx

The BmapiGetBRCMNicStatisticsEx() will return Broadcom network adapters proprietary statistics data.

```
U32 BmapiGetBRCMNicStatisticsEx(  
    IN U32 handle,  
    IN/OUT BM_BRCM_STATISTICS_EX *pBrcmStatisticsEx );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pBrcmStatistics

Pointer to a BM_BRCM_STATISTICS_EX structure for Broadcom network adapter proprietary statistics information. The data will be filled out on return.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

Applications should allocate buffers for the statistics structures and pass pointers to those structures to the function.

The 'version' field of BM_BRCM_STATISTICS_EX must be filled prior to call BmapiGetBRCMNicStatisticsEx(). The latest version is defined as BM_BRCM_STATISTICS_EX_VER in BMAPI.h.

5.49 BmapiGetBrcmNicParamList

The function will retrieve Broadcom network adapter's parameter list that are shown in the advance tab in NIC's property page.

```
U32 BmapiGetBrcmNicParamList(  
    IN U32 handle,  
    IN U8 *pParamList,  
    IN/OUT U32 *pLen );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pParamList

On input, applications pass a pointer to a buffer. On return the buffer will be filled with list of parameters' names in REG_MULTI_SZ format (an array of null-terminated strings, terminated by two null characters).

pLen

It is a pointer to a U32 number. On input, it is the length of buffer. On return, it is the required buffer length.

Return Value:

Return BMAPI_OK if successful, otherwise a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

1. Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
2. If 'pBuf' is NULL, 'pLen' will return the required length of buffer.
3. If the function failed due to buffer too short, 'pLen' will be the required length of buffer.
4. Returned data in 'pParamList' is in REG_MULTI_SZ format. (an array of null-terminated strings, terminated by two null characters.)

5.50 BmapiGetBrcmNicParamInfo

The function will retrieve information of Broadcom network adapter's parameter.

```
U32 BmapiGetBrcmNicParamInfo (
    IN U32 handle,
    IN U8 *pParam,
    IN U8 *pCurVal,
    IN/OUT U32 *pCurValLen,
    IN U8 *pParamInfo,
    IN/OUT U32 *pParamInfoLen,
    IN U8 *pParamEnum,
    IN/OUT U32 *pParamEnumLen );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pParam

pointer to the name of the parameter

pCurVal

On input, applications pass a pointer to a buffer that will be filled with current value of the parameter.

pCurValLen

It is a pointer to a U32 number. On input, it is the length of buffer for 'pCurVal'. On return, it is the required buffer length for 'pCurVal'.

pParamInfo

On input, applications pass a pointer to a buffer. On return the buffer will be filled with list of the parameter's information in REG_MULTI_SZ format (an array of null-terminated strings, terminated by two null characters).

pParamInfoLen

It is a pointer to a U32 number. On input, it is the length of buffer for 'pParamInfo'. On return, it is the required buffer length for 'pParamInfo'.

pParamEnum

On input, applications pass a pointer to a buffer. On return the buffer will be filled with list of the parameter's enum information in REG_MULTI_SZ format (an array of null-terminated strings, terminated by two null characters).

pParamEnumLen

It is a pointer to a U32 number. On input, it is the length of buffer for 'pParamEnum'. On return, it is the required buffer length for 'pParamEnum'.

Return Value:

Return BMAPI_OK if successful, otherwise a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

1. Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
2. If 'pCurVal' is NULL, 'pCurValLen' will return the required length of 'pCurVal'.
3. If 'pParamInfo' is NULL, 'pParamInfoLen' will return the required length of 'pParamInfo'.
4. If 'pParamEnum' is NULL, 'pParamEnumLen' will return the required length of 'pParamEnum'.
5. If the function failed due to buffer too short, 'pCurValLen', 'pParamInfoLen' and 'pParamEnumLen' will be the required length of their corresponding buffer.
6. Returned data in 'pParamInfo' and 'pParamEnum' are in REG_MULTI_SZ format (An array of null-terminated strings, terminated by two null characters.) with pairs of name and value information. For example, 'pParamInfo' of parameter 'Enable8021p' could be "ParamDesc\0802.1p QOS\0type\0enum\0\0" which means that 'ParamDesc' is set to value '802.1p QOS' (the string to display in GUI) and 'type' is set to value 'enum'. Another example, 'pParamEnum' of parameter 'Enable8021p' could be "0\0Disable\01\0Enable\0\0" which means that 'Enable8021p' could set to value 0 which means 'disable' and set to 1 which means 'enable'. One more example, 'pParamInfo' of parameter 'NetworkAddress' could be "Default\0\0type\0edit\0\0" which means that 'Default' is set to value "" (empty string) and 'type' is set to value 'edit'. Application MUST be able to handle the empty string case.
7. 'pParamEnum' information is available only when 'Type' in 'pParamInfo' is set to 'enum'.
8. The complete information including possible name and value of the information can be found in Microsoft's Windows 2000 DDK, 'Specifying Configuration Parameters for the Advanced Properties Page'.
9. If, currently, no 'pParam' is set in registry, '*pCurValLen' will be set to 0.
10. If pParamInfoLen is NULL, pParamInfo, pParamInfoLen, pParamEnum and pParamEnumLen will all NOT be filled with information. If pParamInfoLen is not NULL and pParamEnumLen is NULL, pParamEnum and pParamEnumLen will NOT be filled.

5.51 BmapiSetBrcmNicParam2

The function will retrieve information of Broadcom network adapter's parameter.

```
U32 BmapiSetBrcmNicParam2 (
    IN U32 handle,
    IN U8 *pParam,
    IN pNewVal );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pParam

pointer to the name of the parameter

pNewVal

the new value to set for 'pParam'

Return Value:

Return BMAPI_OK if successful, otherwise a nonzero error code.

Supported Network Adapters:

NetLink and NetXtreme I

Remarks:

1. Applications must call BmapiGetAllPhyNic(), etc. to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
2. If pNewVal is NULL, the 'pParam' will be deleted from registry.

5.52 BmapiGetLicenseKey

The API is obsolete.

5.53 BmapiSetLicenseKey

The API is obsolete.

5.54 BmapiGet5706FwInfo

The API is obsolete.

5.55 BmapiGet57710FwInfo

The API is obsolete.

5.56 BmapiGetMgmtProcessors

BmapiGetMgmtProcessors() will return the type of management processors on the controllers.

```
U32 BmapiGetMgmtProcessors(  
    IN U32 handle,  
    OUT U32 *pProc );
```

Parameters:

handle

handle to a NIC port

pProc

pointer to a U32 number. The number is a bit definition to various types of management processors in our controllers.

Return Value:

Return BMAPI_OK if information was retrieved successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

NetXtreme I family

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.57 BmapiGetMgmtEnableState

BmapiGetMgmtEnableState() will return whether management FW is enabled or not.

```
U32 BmapiGetMgmtEnableState(  
    IN U32 handle,  
    OUT U32 *pEnable );
```

Parameters:

handle

handle to a NIC port

pEnable

pointer to a U32 number. Set to '0' if management FW is disabled in NVRAM. Non-zero value if it is enabled in NVRAM.

Return Value:

Return BMAPI_OK if information was retrieved successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

NetXtreme I family

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.58 BmapiSetMgmtEnableState

BmapiSetMgmtEnableState() will set management FW to enable or disable.

```
U32 BmapiSetMgmtEnableState(  
    IN U32 handle,  
    IN U32 bEnable );
```

Parameters:

handle

handle to a NIC port

bEnable

'0' to disable management FW and non-zero to enabled.

Return Value:

Return BMAPI_OK if configuration was set successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

NetXtreme I family

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.59 BmapiAssertMgmtEvent

BmapiAssertMgmtEvent() will assert an APE event.

```
U32 BmapiAssertMgmtEvent (  
    IN U32 handle,  
    IN U8 uEventID,  
    IN U8 uEventData,  
    IN U8 Reserved,  
    IN void *pMsg,  
    IN U32 uMsgLen );
```

Parameters:

handle

handle to a NIC port

uEventID

Event ID

uEventData

Event data

Reserved

reserved

**pMsg*

Pointer to the message for the event

uMsgLen

length of the message in 'pMsg'

Return Value:

Return BMAPI_OK if event was asserted successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.60 BmapiGetMgmtOTPKeys

BmapiGetMgmtOTPKeys() will read OTP keys.

```
U32 BmapiGetMgmtOTPKeys (  
    IN U32 handle,  
    OUT U32 *pKey1,  
    OUT U32 *pKey2 );
```

Parameters:

handle

Handle to a NIC port

pKey1

Pointer to the first part of the key on return

pKey2

Pointer to the second part of the key on return

Return Value:

Return BMAPI_OK if keys were read retrieved successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.61 BmapiGetMgmtDataLength

BmapiGetMgmtDataLength() will get the length of APE data.

```
U32 BmapiGetMgmtDataLength(  
    IN U32 handle,  
    OUT U32 *pLength );
```

Parameters:

handle

Handle to a NIC port

pLength

Pointer to the APE data length (filled on return)

Return Value:

Return BMAPI_OK if length was read successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.62 BmapiGetMgmtData

BmapiGetMgmtData() will get APE data.

```
U32 BmapiGetMgmtData (  
    IN U32 handle,  
    IN U32 uStart,  
    OUT void *pBuf,  
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

uStart

Starting offset to read APE data (from beginning of APE data)

pBuf

Pointer to the APE data (filled on return)

uBufLen

Length of buffer ('pBuf')

Return Value:

Return BMAPI_OK if APE data is read successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.63 BmapiSetMgmtData

BmapiSetMgmtData() will set APE data.

```
U32 BmapiSetMgmtData (  
    IN U32 handle,  
    IN U32 uStart,  
    IN void *pBuf,  
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

uStart

Starting offset to write APE data (from beginning of APE data)

pBuf

Pointer to the APE data that will write to the NIC

uBufLen

Length of buffer ('pBuf') to write to NIC

Return Value:

Return BMAPI_OK if APE data is written successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.64 BmapiGetMgmtConfigLength

BmapiGetMgmtConfigLength() will get the length of management configuration data.

```
U32 BmapiGetMgmtConfigLength (
    IN U32 handle,
    OUT U32 *pLength );
```

Parameters:

handle

Handle to a NIC port

pLength

Pointer to the management configuration data length (filled on return)

Return Value:

Return BMAPI_OK if the length is retrieved successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

NetXtreme I family

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.65 BmapiGetMgmtConfig

BmapiGetMgmtConfig() will read management configuration data.

```
U32 BmapiGetMgmtConfig(  
    IN U32 handle,  
    OUT void *pBuf,  
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

pBuf

Pointer to the management configuration data (filled on return)

uBufLen

Length of the buffer for management configuration data ('pBuf')

Return Value:

Return BMAPI_OK if the management configuration data is read successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

NetXtreme I family

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.66 BmapiSetMgmtConfig

BmapiSetMgmtConfig() will write management configuration data.

```
U32 BmapiSetMgmtConfig(  
    IN U32 handle,  
    IN void *pBuf,  
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

pBuf

Pointer to the management configuration data that will write to the NIC

uBufLen

Length of buffer for management configuration data to write

Return Value:

Return BMAPI_OK if the management configuration data is written to the NIC successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

NetXtreme I family

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.67 BmapiGetMgmtSharedMem

BmapiGetMgmtSharedMem() will read APE shared memory.

```
U32 BmapiGetMgmtSharedMem(  
    IN U32 handle,  
    IN U32 uStart,  
    IN void *pBuf,  
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

uStart

Starting offset to read APE shared memory

pBuf

Pointer to the APE shared memory data (filled on return)

uBufLen

Length of buffer ('pBuf') in bytes, must be multiple of DWORD

Return Value:

Return BMAPI_OK if APE shared memory is read successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.68 BmapiWritePhyFirmware

The API is obsolete.

5.69 BmapiCreateMgmtData

BmapiCreateMgmtData() will create APE_DATA section in NVRAM. If the APE_DATA exists, it will be resized to 'uLength'. The content of APE_DATA should be zeroed-out.

```
U32 BmapiCreateMgmtData (  
    IN U32 handle,  
    IN U32 uLength );
```

Parameters:

handle

Handle to a NIC port

uLength

length of APE_DATA

Return Value:

Return BMAPI_OK if APE_DATA created, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.70 BmapiGetMgmtWebDataLength

BmapiGetMgmtWebDataLength() will get the length of APE WEB data (if it exists).

```
U32 BmapiGetMgmtWebDataLength (  
    IN U32 handle,  
    OUT U32 *pLength );
```

Parameters:

handle

Handle to a NIC port

pLength

pointer to the APE WEB data length (filled on return)

Return Value:

Return BMAPI_OK if length was read successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.71 BmapiGetMgmtWebData

BmapiGetMgmtWebData() will get APE WEB data.

```
U32 BmapiGetMgmtWebData (
    IN U32 handle,
    IN U32 uStart,
    OUT void *pBuf,
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

uStart

Starting offset to read APE WEB data (from beginning of APE WEB data)

pBuf

Pointer to the APE WEB data (filled on return)

uBufLen

Length of buffer ('pBuf')

Return Value:

Return BMAPI_OK if APE WEB data is read successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.72 BmapiSetMgmtWebData

BmapiSetMgmtWebData() will set APE WEB data.

```
U32 BmapiSetMgmtWebData (  
    IN U32 handle,  
    IN U32 uStart,  
    IN void *pBuf,  
    IN U32 uBufLen );
```

Parameters:

handle

Handle to a NIC port

uStart

Starting offset to write APE WEB data (from beginning of APE WEB data)

pBuf

Pointer to the APE WEB data that will write to the NIC

uBufLen

Length of buffer ('pBuf') to write to NIC

Return Value:

Return BMAPI_OK if APE WEB data is written successfully, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

1. Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
2. 'uBufLen' must be in multiple of 4 bytes.

5.73 BmapiCreateMgmtWebData

BmapiCreateMgmtWebData() will create APE_WEB_DATA section in NVRAM. If the APE_WEB_DATA exists, it will be resized to 'uLength'. The content of APE_WEB_DATA should be zeroed-out.

```
U32 BmapiCreateMgmtWebData(  
    IN U32 handle,  
    IN U32 uLength );
```

Parameters:

handle

Handle to a NIC port

uLength

length of APE_WEB_DATA

Return Value:

Return BMAPI_OK if APE_WEB_DATA created, otherwise a nonzero error code will be returned.

Supported Network Adapters:

DASH supported chips

Remarks:

Applications must call BmapiGetAllPhyNic() or BmapiGetPhyNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.

5.74 BmapiRetrieveMultiLinkStatus

BmapiRetrieveMultiLinkStatus() will return multiple link information.

```
U32 BmapiRetrieveMultiLinkStatus(  
    IN/OUT BM_LINK_STATUS_EX **ppLinkStatusEx,  
    IN U32 uNumOfStruct );
```

Parameters:

ppLinkStatusEx

Pointer to the pointer array of BM_LINK_STATUS_EX for all link status structures that need to be filled.

uNumOfStruct

Number of BM_LINK_STATUS_EX structures in 'ppLinkStatusEx' array.

Return Value:

Return BMAPI_OK if all information were collected, otherwise a nonzero error code will be returned.

Supported Network Adapters:

All

Remarks:

1. Applications must call BmapiGetAllPhyNic() or BmapiGetAllUnassignedNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
2. On input, 'version' and 'handle' fields must be provided.

5.75 BmapiGetIscsiRuntimeIPCount

The API is obsolete.

5.76 BmapiGetIscsiRuntimeIP

The API is obsolete.

5.77 BmapiGetIscsiRuntimeStatistics

The API is obsolete.

5.78 BmapiGetIscsiSessionStatistics

The API is obsolete.

5.79 BmapiWriteFirmware2

BmapiWriteFirmware2() will write firmware information starting at specified offset with provided data and len to write to. This API is not supported by BMAPILNX.

```
U32 BmapiWriteFirmware2(  
    IN U32 handle,  
    IN U32 uOffset,  
    IN U32 *pDataBuf,  
    IN U32 uBufLen,  
    IN U8 *pChk,  
    IN U32 uTarget );
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uOffset

Starting offset to write data to.

pDataBuf

Data to write to firmware. Data are in 32-bit array.

uBufLen

Number of elements in data buffer array. For example, 2 means 2 32-bit data in the data buffer.

uTarget

BMAPI_NVRAM_TARGET_ACTIVE,
BMAPI_NVRAM_TARGET_NVRAM or
BMAPI_NVRAM_TARGET_OTP.

Return Value:

Always return BMAPI_FEATURE_NOT_AVAILABLE error code.

Supported Network Adapters:

NetLink (exclude 4401), NetXtreme I

Remarks:

1. Applications need to call BmapiInitDiag() prior to call this and call BmapiUnInitDiag() after everything is done.
2. The newly programmed firmware will not be effective till:
 - The machine reboot.
 - The driver is disabled and re-enabled. Can be done manually or through BmapiEnableDevice().
 - Call BmapiSuspendDriverEx() followed by BmapiResumeDriverEx().

This is the most preferable method.

3. 'uTarget' is applicable only to NetLink and NetXtreme.
4. BMAPI_NVRAM_TARGET_OTP is applicable only to some NetXtreme devices only.

5.80 BmapiReadFirmware2

BmapiReadFirmware2() will read firmware information starting at specified offset with provided data buffer and length to read. This API is not supported by BMAPILNX.

```
U32 BmapiReadFirmware2(  
    IN U32 handle,  
    IN U32 uOffset,  
    IN U32 *pDataBuf,  
    IN U32 uBufLen,  
    IN U8 *pChk,  
    IN U32 uTarget )
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

uOffset

Starting offset to read data from.

pDataBuf

Data buffer from firmware data. Data are in 32-bit array.

uBufLen

Number of elements in data buffer array. For example, 2 means 2 32-bit data in the data buffer.

uTarget

BMAPI_NVRAM_TARGET_ACTIVE,
BMAPI_NVRAM_TARGET_NVRAM or
BMAPI_NVRAM_TARGET_OTP.

Return Value:

Always return BMAPI_FEATURE_NOT_AVAILABLE error code.

Supported Network Adapters:

NetLink (exclude 4401), NetXtreme I

Remarks:

1. Applications need to call BmapiInitDiag() prior to call this and call BmapiUnInitDiag() after everything is done.
2. 'uTarget' is applicable only to NetLink and NetXtreme.
3. BMAPI_NVRAM_TARGET_OTP is applicable only to some NetXtreme devices only.

5.81 BmapiGetNicStatisticsV3

BmapiGetNicStatisticsV3() will return statistics information for a NDIS driver.

```
U32 BmapiSetReverseNWay(  
    IN U32 handle,  
    IN/OUT BM_GENERAL_STATISTICS_EX *pGenStatistics,  
    IN/OUT BM_ETHERNET_STATISTICS_EX *pEthStatistics )
```

Parameters:

handle

Handle to an adapter BM_ADAPTER_INFO structure.

pGenStatistics

Pointer to a BM_GENERAL_STATISTICS_EX structure for general statistics information of a network adapter. The data will be filled out on return.

pEthStatistics

Pointer to a BM_ETHERNET_STATISTICS_EX structure for ethernet statistics information of a network adapter. The data will be filled out on return.

Return Value:

Return BMAPI_OK if successful; otherwise return a nonzero error code.

Supported Network Adapters:

NetLink, NetXtreme I

Remarks:

1. Applications should allocate buffers for the statistics structures and pass pointers to those structures to the function. Application should be aware that some information might not be available.
2. "version" in the structure should be initialized on input.

5.82 BmapiGetLldpParams

BmapiGetLldpParams() will retrieve LLDP parameters.

```
U32 BmapiGetLldpParams (  
    IN U32 handle,  
    IN/OUT BM_LLDP_PARAMS *pLldpParams )
```

Parameters:

handle

Handle to the NIC information that should be retrieved.

pLldpParams

Pointer to BM_LLDP_PARAMS for LLDP parameters.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

FCoE supported adapters

Remarks:

1. Applications should allocate buffers for the structures and pass the pointer to the structure to the function. Application should be aware that some information might not be available.

5.83 BmapiGetDcbxParams

BmapiGetDcbxParams() will retrieve DCBX parameters.

```
U32 BmapiGetDcbxParams (  
    IN U32 handle,  
    IN/OUT BM_DCBX_PARAMS *pDcbxParams )
```

Parameters:

handle

Handle to the NIC information that should be retrieved.

pDcbxParams

Pointer to BM_DCBX_PARAMS for DCBX parameters.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

FCoE supported adapters

Remarks:

1. Applications should allocate buffers for the structures and pass the pointer to the structure to the function. Application should be aware that some information might not be available.

5.84 BmapiGetIscsiCfg

BmapiGetIscsiCfg() will read iSCSI boot configuration block from NVRAM.

```
U32 BmapiGetIscsiCfg(  
    IN U32 handle,  
    IN/OUT U8 *pBuf,  
    IN/OUT U32 *pLen )
```

Parameters:

handle

Handle to the NIC to read iSCSI boot configuration.

pBuf

Buffer for iSCSI boot configuration

pLen

Pointer to the buffer length on input and actual iSCSI configuration length on return.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

All adapters that have iSCSI boot programmed

Remarks:

1. If 'pBuf' is NULL, the required buffer length will be in 'pLen'. If 'pBuf' is not NULL, the iSCSI CFG block length will be in 'pLen'.

5.85 BmapiSetIscsiCfg

BmapiSetIscsiCfg() will update iSCSI boot configuration block in NVRAM.

```
U32 BmapiSetIscsiCfg(  
    IN U32 handle,  
    IN U8 *pBuf,  
    IN U32 uLen )
```

Parameters:

handle

Handle to the NIC to write iSCSI boot configuration.

pBuf

Buffer for iSCSI boot configuration

uLen

How many bytes from the 'pBuf' to write to NVRAM.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

All adapters that have iSCSI boot programmed

Remarks:

1. The 'uLen' must be the same length as existing iSCSI boot configuration block.

5.86 BmapiGetFcoeCfg

BmapiGetFcoeCfg() will read FCoE boot configuration block from NVRAM.

```
U32 BmapiGetFcoeCfg(  
    IN U32 handle,  
    IN/OUT U8 *pBuf,  
    IN/OUT U32 *pLen )
```

Parameters:

handle

Handle to the NIC to read iSCSI boot configuration.

pBuf

Buffer for FCoE boot configuration

pLen

Pointer to the buffer length on input and actual FCoE configuration length on return.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

All adapters that have FCoE boot programmed

Remarks:

1. If 'pBuf' is NULL, the required buffer length will be in 'pLen'. If 'pBuf' is not NULL, the FCoE boot configuration block length will be in 'pLen'.

5.87 BmapiSetFcoeCfg

BmapiSetFcoeCfg() will update FCoE boot configuration block in NVRAM.

```
U32 BmapiSetFcoeCfg(  
    IN U32 handle,  
    IN U8 *pBuf,  
    IN U32 uLen )
```

Parameters:

handle

Handle to the NIC to write FCoE boot configuration.

pBuf

Buffer for FCoE boot configuration

uLen

How many bytes from the 'pBuf' to write to NVRAM.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

All adapters that have FCoE boot programmed

Remarks:

1. The 'uLen' must be the same length as existing iSCSI boot configuration block.

5.88 BmapiGetMBAParams

BmapiGetMBAParams() will retrieve MBA parameters from NVRAM.

```
U32 BmapiGetMBAParams (
    IN U32 handle,
    IN/OUT BM_MBA_PARAMS *pMbaParams )
```

Parameters:

handle

Handle to the NIC information that should be retrieved.

pMbaParams

Pointer to BM_MBA_PARAMS for MBA parameters.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

All NXI adapters

Remarks:

The 'handle' should be to a physical NIC/VBD.

5.89 BmapiSetMBAParams

BmapiSetMBAParams() will update MBA parameters in NVRAM.

```
U32 BmapiSetMBAParams (
    IN U32 handle,
    IN BM_MBA_PARAMS *pMbaParams )
```

Parameters:

handle

Handle to the NIC information that should be updated.

pMbaParams

Pointer to BM_MBA_PARAMS for MBA parameters.

Return Value:

Returns BMAPI_OK if successful; otherwise returns a nonzero error code.

Supported Network Adapters:

All NXI adapters

Remarks:

The 'handle' should be to a physical NIC/VBD.

5.90 BmapiGetNicPartCfg

The API is obsolete.

5.91 BmapiSetNicPartCfg

The API is obsolete.

5.92 BmapiGetExtVPD

BmapiGetExtVPD() will read extended VPD into buffer.

```
U32 BmapiGetExtVPD(  
    IN U32 handle,  
    IN/OUT U8 *pBuf,  
    IN/OUT U32 *pLen )
```

Parameters:

handle

Handle to the device interface.

pBuf

Buffer for extended VPD data

pLen

Pointer to the buffer length on input and actual extended VPD data length on return.

Return Value:

Returns BMAPI_OK if the result is collected successfully; otherwise returns a nonzero error code.

Supported Network Adapters:

All adapters that have extended VPD programmed

.

Remarks:

1. If 'pBuf' is NULL, the required buffer length will be in 'pLen'. If 'pBuf' is not NULL, the extended VPD data block length will be in 'pLen'.
2. For NXII, the 'handle' must be a VBD handle.

5.93 BmapiSetExtVPD

BmapiSetExtVPD() will write extended VPD into NVRAM.

```
U32 BmapiGetExtVPD(  
    IN U32 handle,  
    IN U8 *pBuf,  
    IN U32 uLen )
```

Parameters:

handle
Handle to the device interface.

pBuf
Buffer for extended VPD data

uLen
VPD buffer length

Return Value:

Returns BMAPI_OK if the result is collected successfully; otherwise returns a nonzero error code.

Supported Network Adapters:

All adapters that have extended VPD programmed

Remarks:

1. Applications must call BmapiGetPhyNic(), BmapiGetAllPhyNic() or GetAllUnassignedNic() to get adapter information. Applications will find the 'handle' in BM_ADAPTER_INFO and use it to call the function.
2. The 'uLen' must be DWORD aligned.

5.94 BmapiGetDcbNvramCfg

The API is obsolete.

5.95 BmapiSetDcbNvramCfg

The API is obsolete.

5.96 BmapiGetNivCfg

The API is obsolete.

5.97 BmapiSetNivCfg

The API is obsolete.

5.98 BmapiGetNivPortProfile

The API is obsolete.

5.99 BmapiGetFLRCfg

The API is obsolete.

5.100 BmapiSetFLRCfg

The API is obsolete.

5.101 BmapiGetSRIOVforSF

The API is obsolete.

5.102 BmapiSetSRIOVforSF

The API is obsolete.

5.103 BmapiGetSRIOVSwitchInfo

The API is obsolete.

5.104 BmapiGetSRIOVSwitchStats

The API is obsolete.

5.105 BmapiGetSRIOVVFStats

The API is obsolete.

5.106 BmapiGetSRIOVVFInfo

The API is obsolete.

CONFIDENTIAL

6. Quick Programming Guide

6.1 How to get physical network adapter information?

Make sure BMAPI is initialized. And, do the following:

- Call BmapiGetNumPhyNic() to get the number of physical network adapters installed (with driver) in the system.
- Allocate enough buffer to host all data.
- Call BmapiGetAllPhyNic() to get all information for all physical network adapters.

or

- Call BmapiGetNumPhyNicEx() to get the number of physical network adapters in the system. In case of Windows 2000, the number will include network adapters not installed with driver.
- Allocate enough buffer that can host all handles for all NICs.
- Call BmapiGetAllPhyNicHandles() to get all handles for all NICs.
- Call BmapiGetPhyNic() to get detail information of a specific physical network adapter.

6.2 How to get handle to a physical network adapter?

Please follow section 5.1 to get each NIC's information. Handle can be found in every return data structure.

If the service name of a NIC is known to the application, the application can also call BmapiGetHandleByServiceName() to retrieve a handle to the NIC.

6.3 How to do diagnostics?

Make sure BMAPILNX is initialized. And, do the following:

- Call BmapiInitDiag() (required step).
- Call Diag APIs such as BmapiTestControlRegisters(), BmapiTestMIIRegisters(), etc.
- Call BmapiUnInitDiag() (required step).

Please note that BmapiInitDiag() and diagnostic APIs must be called from the same thread.

6.4 How to do ASF?

- Make sure BMAPI is initialized.
- Call BmapiGetASFTable() to get ASF table from a NIC.
- Call BmapiSetASFTable() to write ASF table to a NIC.

6.5 How to tell which type of NIC it is?

- Use 'nic_type' in BM_ADAPTER_INFO.