CBIOS 1.7 API Reference (local and network)

Document: 0-20May010ks(CBIOS API Reference).odt
Last update: 11 November 2021 by Steffen Kaetsch
Environment: C++ (Microsoft Visual Studio), Delphi, Visual Basic

Executive summary

This document describes the most important part of the CRYPTO-BOX API for developers: CBIOS API. It provides functions for identification and access to the internal memory and encryption functions of a connected CRYPTO-BOX. To realize an effective protection scheme, CBIOS is always required while all other SmarxOS APIs (DO, RFP, DP) provide extended functionality on top of it.

CBIOS version 1.5 and up covers the following types of USB hardware:

- *CRYPTO-BOX® SC (CBU SC);*
- *CRYPTO-BOX® XS/Versa (CBU XS/Versa).*

The CRYPTO-BOX SC is fully backward compatible to the CRYPTO-BOX XS and Versa models.

If you are looking for C#.NET support, please refer to our separate "CBIOS4NET Developer's Guide" (available at www.marx.com -> Support -> White Papers).

Quick and efficient hardware based software protection!

Software and information piracy costs billions of dollars in annual losses to software vendors, distributors and content providers worldwide. The internets role in software and data distribution is growing rapidly and increases the importance of the situation dramatically. Hardware based protection can be used for creating robust and reliable secure demo versions of applications in a straightforward manner. While benefiting from strong and effective application protection provided by a hardware based approach you can create flexible and secure demo versions with ease. The CRYPTO-BOX makes it happen!

Order your CRYPTO-BOX Evaluation Kit now:

www.marx.com/eval

MARX Software Security GmbH
Vohburger Strasse 68
85104 Wackerstein, Germany
Phone: +49 (0) 8403 / 9295-0
contact-de@marx.com

MARX CryptoTech LP
489 South Hill Street
Buford, GA 30518 U.S.A.
Phone: (+1) 770 904 0369
contact@marx.com

www.marx.com

Table of Contents

1. Smarx®OS CBIOS API Reference

1.1. Overview

This chapter describes the most important part of the CRYPTO-BOX API for developers: CBIOS API. It provides functions for CRYPTO-BOX identification and access to the internal memory and encryption functions of a connected CRYPTO-BOX and is mainly targeted to C/C++, Delphi and Visual Basic developers.

For .NET developers we provide a separate Developer's Guide which explains implementation details and syntax of our object oriented component based SmarxOS API for .NET platform: CBIOS4NET. The CBIOS4NET Developer's Guide is available on www.marx.com -> Support -> White Papers.

This document contains the CBIOS API reference only. If you need more information first on how to start implementing the CRYPTO-BOX with API:

- Our White Paper “Implementation with API” provides a general introduction and overview about all available APIs for the CRYPTO-BOX, including the new object oriented Smarx API.
- Read chapter 14 in the Smarx Compendium first before working with this document – it will help you to understand the CBIOS call sequence.
- For an introduction to network usage, please refer to Smarx Compendium, chapter 12.
- CRYPTO-BOX SC specific API functions (extended hardware based AES and RSA functions) are described in the Smarx Compendium, chapter 12.9.

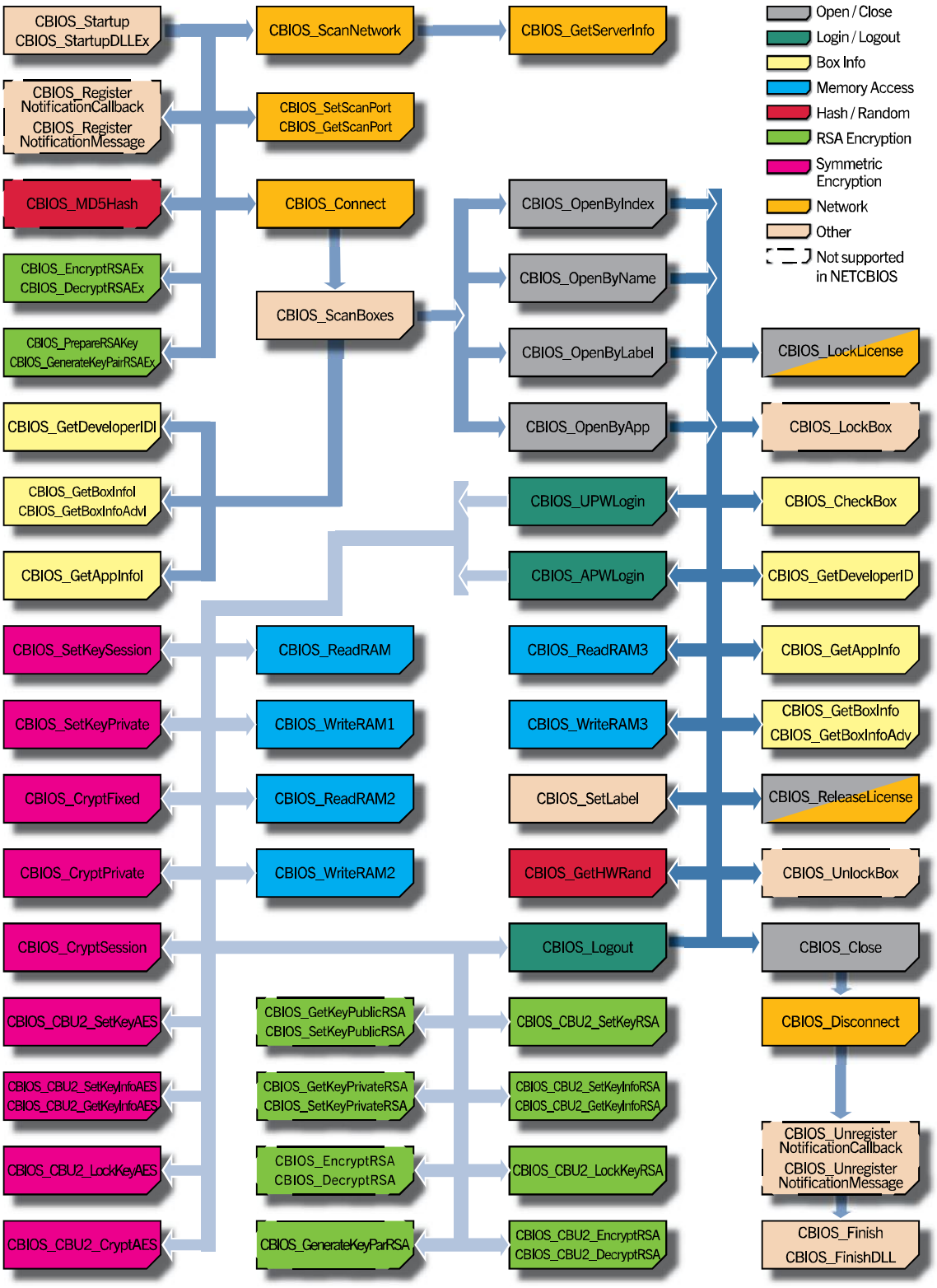

Fig. 1.1: CBIOS API calls – overview

1.2. Smarx®OS system brackets

DWORD WINAPI CBIOS_Startup()
DWORD WINAPI CBIOS_StartupDLLEx(HMODULE hDll);
(Only if called within a DLL in DLL_PROCESS_ATTACH)

Delphi syntax: **function CBIOS_Startup: Longint; stdcall;**
function CBIOS_StartupDllEx(hDll : THandle): Longint; stdcall;

Visual Basic syntax: **Function CBIOS_Startup () As Long**

This function initializes CBIOS API for this application. It is not necessary to call this function explicitly, it will be called in background if the first CBIOS call is CBIOS_ScanBoxes(). But you always have to make sure to call corresponding CBIOS_Finish before closing the application. If you need to use CBIOS from within a DLL it is strongly recommended to create two special functions in your DLL, like: Begin_CBIOS_Support() and End_CBIOS_Support() calling CBIOS_Startup() and CBIOS_Finish() correspondingly. The first one must be called before any other CBIOS related DLL function call, while the second must be the last one.
If for some reason this is not feasible, then the following "special" DLL implementation of CBIOS system brackets has to be used:

```
//**********************************
BOOL APIENTRY DllMain (
       HANDLE hModule
       , DWORD ul_reason_for_call
       , LPVOID lpReserved )
{
       switch (ul_reason_for_call) {
              case DLL_PROCESS_ATTACH:
              // Important!!! Pass hModule into CBIOS_StartupDLLEx.
              // If none then this call will be equivalent to CBIOS_Startup ().
              // i.e. memory leaks.
                     CBIOS_StartupDLLEx (hModule);
                     break;
              case DLL_PROCESS_DETACH:
                     CBIOS_FinishDLL();
                     break;
       }
       return TRUE;
}
//**********************************
```

It is not recommended to do it from within DLLMain, because thread and window creation is not supported at that point. That's why special non-standard functions are provided, bypassing standard Windows limitations – CBIOS_StartupDLLEx() / CBIOS_FinishDLL() used in the above sample of DLLMain entry point.

Return:

CBIOS_ERR_NOT_INITIALIZED	CBIOS initialization error. Further CBIOS work is not possible
CBIOS_ERR_INIT_IN_PROGRESS	CBIOS initialization process is not yet complete.

CBIOS_WRN_CTM_NOT_INITIALIZED	Warning: CTM2048 support (deprecated. CBIOS works properly, CTM2048 support doesn't work
CBIOS_WRN_INIT_BOX_NOTIFIC	Warning: CRYPTO-BOX insert-remove notification subsystem initialization failed. CBIOS will work OK, however event notifications may not come to applications.

DWORD WINAPI CBIOS_Finish()
DWORD WINAPI CBIOS_FinishDLL()
(If called within a DLL)

Delphi syntax: **function CBIOS_Finish: Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_Finish () As Long**

Releases memory, allocated by CBIOS_Startup(). It has to be called before application exits to eliminate possible memory leaks.
If CBIOS initialization is performed in DLL (DllMain() function), then CBIOS_StartupDLLEx(), CBIOS_FinishDLL() has to be used (see example in CBIOS_Startup() description).

DWORD WINAPI CBIOS_GetLastError()

Delphi syntax: **function CBIOS_GetLastError: Longint; stdcall;**

Visual Basic syntax: **Function GetLastError () As Long**

Returns CBIOS last error code

Parameters: None

Return:
Additionally to CBUSB standard return codes:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode

1.3. Getting information about the attached CRYPTO-BOX®

INT32 WINAPI CBIOS_ScanBoxes()

Delphi syntax: **function CBIOS_ScanBoxes: Integer; stdcall;**

Visual Basic syntax: **Function CBIOS_ScanBoxes () As Integer**

Scans USB local ports and determines attached CRYPTO-BOX (if any).

Parameters: None

Return:
Number of attached CRYPTO-BOXes. In case of CBIOS Extended Mode – returns zero.
Returns "-1" in case of CBIOS initialization error.

Although it is sufficient to call this function only once (for one application), we recommend you to register a callback notification handling function **CBIOS_RegisterNotificationCallback()** or **CBIOS_RegisterNotificationMessage()** to keep track of hardware configuration changes (i.e. after attaching/detaching a CRYPTO-BOX). Alternatively, CBIOS_ScanBoxes() needs to be called each time the application wants to refresh information about the hardware configuration.

DWORD WINAPI CBIOS_GetBoxInfoI(INT32 iBoxIndex, CBIOS_BOX_INFO* pBoxInfo)
DWORD WINAPI CBIOS_GetBoxInfo(CBIOS_BOX_INFO* pBoxInfo)
(this function is included for compatibility purposes, use GetBoxInfoAdvI() / GetBoxInfoAdv() instead)

Delphi syntax: **function CBIOS_GetBoxInfoI(iBoxIndex: Integer;**
pBoxInfo: PCBIOS_BOX_INFO
):Longint; stdcall;
function CBIOS_GetBoxInfo(pBoxInfo: PCBIOS_BOX_INFO): Longint; stdcall;

Visual Basic syntax: **Function CBIOS_GetBoxInfoI (ByVal iCount As Integer,**
ByRef stInfo As CBIOS_BOX_INFO) As Long
Function CBIOS_GetBoxInfo(ByRef pBoxInfo As CBIOS_BOX_INFO) As Long

Returns CRYPTO-BOX info based on its Index.

CBIOS_GetBoxInfoI() allows to to get information about any one of attached CRYPTO-BOX units before opening it. CBIOS_GetBoxInfo assumes that the CRYPTO-BOX was already opened.

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # from 1 to <Box Quantity> value, Returned by the preceding CBIOS_ScanBoxes() call
CBIOS_BOX_INFO* pBoxInfo	IN/OUT: Pointer to the structure which contains CRYPTO-BOX information.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox)

CBIOS_ERR_BOX_NOT_FOUND CRYPTO-BOX not found (invalid iBoxIndex)

Or standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h)

CBIOS_BOX_INFO structure contains the following information (also see cbios.h):

DWORD	**dwBoxName**	dwBoxName
DWORD	**dwModel**	CRYPTO-BOX Model (Legacy/VERSA/XS/XL – see cbios.h)
DWORD	**dwApp**	Number of partitions (applications) registered in this CRYPTO-BOX (**CBIOS_GetBoxInfoAdvI**) or currently opened partition (**CBIOS_GetBoxInfoAdv**)
WORD	**wCBIOS**	CRYPTO-BOX CBIOS version info: 0 – means not a CBIOS configured CRYPTO-BOX
BYTE	**bAppPolicy**	0 – demo CRYPTO-BOX or digital signature is not valid
BYTE	**bLogin**	CRYPTO-BOX current status: UPW_LOGIN = 0x01, APW_LOGIN = 0x02,
BYTE	**bHiVersion**	Firmware version High (2 for 2.3)
BYTE	**bLoVersion**	Firmware version Low (3 for 2.3)
DWORD	**dwRAMLen**	Full RAM size in bytes
DWORD	**dwSize_RAM1**	Full RAM1 size in bytes
DWORD	**dwSize_RAM2**	Full RAM2 size in bytes
DWORD	**dwSize_RAM3**	Full RAM3 size in bytes
BYTE	**bBoxLabel[CBIOS_LABEL_LEN]**	Volume Label

DWORD WINAPI CBIOS_GetBoxInfoAdvI(INT32 iBoxIndex, CBIOS_BOX_INFO_ADV* pBoxInfo)
DWORD WINAPI CBIOS_GetBoxInfoAdv(CBIOS_BOX_INFO_ADV* pBoxInfo)

Delphi syntax: **function CBIOS_GetBoxInfoAdvI(iBoxIndex: Integer;**
pBoxInfo: PCBIOS_BOX_INFO_ADV
):Longint; stdcall;
function CBIOS_GetBoxInfoAdv(pBoxInfo: PCBIOS_BOX_INFO_ADV
):Longint; stdcall;

Visual Basic syntax: **Function CBIOS_GetBoxInfoAdvI (ByVal iCount As Integer,**
ByRef stInfo As CBIOS_BOX_INFO_ADV) As Long
Function CBIOS_GetBoxInfoAdv (ByRef stInfo As CBIOS_BOX_INFO_ADV) As Long

Returns extended information:
CBIOS_GetBoxInfoAdvI – for the CRYPTO-BOX with the specified Index;
CBIOS_GetBoxInfoAdv – for the currently opened CRYPTO-BOX.

CBIOS_GetBoxInfoAdvI/CBIOS_GetBoxInfoAdv have been added to provide advanced

(extended) information on attached hardware. In case of non-Smarx OS (MPI) formatted CRYPTO-BOX USB CBIOS_GetBoxInfoAdvI/CBIOS_GetBoxInfoAdv calls return proper flag set to ON. It will allow better diagnostics and handling for MPI formatted CRYPTO-BOX USB by Smarx OS applications.

CBIOS_GetBoxInfoAdvI() allows to to get information about any one of attached CRYPTO-BOX units before opening it. CBIOS_GetBoxInfoAdv assumes that the CRYPTO-BOX was already opened.

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # from 1 to <Box Quantity> value, Returned by the preceding CBIOS_ScanBoxes() call
CBIOS_BOX_INFO_ADV* pBoxInfo	OUT: Pointer to the structure which will contain CRYPTO-BOX information. **dwStructSize** must be set to **sizeof(CBIOS_BOX_INFO_ADV)**

Return:

CBIOS_ERR_WRONG_PARAM	dwStructSize value is wrong
CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox)
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (invalid iBoxIndex)

Or standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h)

CBIOS_BOX_INFO_ADV structure contains the following info:

DWORD	**dwStructSize**	IN: sizeof(CBIOS_BOX_INFO_ADV)
DWORD	**dwBoxName**	OUT: dwBoxName
DWORD	**dwModel**	CRYPTO-BOX Model Legacy/Versa/XS/XL – see cbios.h
DWORD	**dwApp**	OUT: Number of partitions (applications) registered in this CRYPTO-BOX (**CBIOS_GetBoxInfoAdvI**) or currently opened partition (**CBIOS_GetBoxInfoAdv**)
WORD	**wCBIOS**	OUT: CRYPTO-BOX CBIOS version info: 0 – means not a SmarxOS formatted CRYPTO-BOX
BYTE	**bAppPolicy**	OUT: 0 – demo CRYPTO-BOX or digital signature is not valid
BYTE	**bLogin**	OUT: CRYPTO-BOX current status: UPW_LOGIN = 0x01, APW_LOGIN = 0x02,
BYTE	**bHiVersion**	OUT: Firmware version High (1 for 1.6)
BYTE	**bLoVersion**	OUT: Firmware version Low (6 for 1.6)
DWORD	**dwRAMLen**	OUT: Full RAM size in bytes
DWORD	**dwSize_RAM1**	OUT: Full RAM1 size in bytes
DWORD	**dwSize_RAM2**	OUT: Full RAM2 size in bytes
DWORD	**dwSize_RAM3**	OUT: Full RAM3 size in bytes
BYTE	**bBoxLabel[CBIOS_LABEL_LEN]**	OUT: CRYPTO-BOX Label

DWORD	**dwDeveloperID**	OUT: DeveloperID (unique value for every MARX customer - see CBIOS_GetDeveloperIDI for more details)
BYTE	**bMPI**	OUT: CRYPTO-BOX is MPI formatted
WORD	**wType**	OUT: CRYPTO-BOX type CBIOS_BOX_UNKNOWN - Unknown type CBIOS_BOX_CBU - CRYPTO-BOX XS/Versa (CBU) CBIOS_BOX_CBU2 - CRYPTO-BOX SC (CBU SC)
DWORD	**dwSize_RAM4**	OUT: Full RAM4 size in bytes (CBU SC)
DWORD	**dwSize_RAM5**	OUT: Full RAM5 size in bytes (CBU SC)

DWORD WINAPI CBIOS_GetDeveloperIDI (I NT iBoxIndex, DWORD* pdwDeveloperID)
DWORD WINAPI CBIOS_GetDeveloperID (DWORD* pdwDeveloperID)

Delphi syntax: **function CBIOS_GetDeveloperIDI (iBoxIndex: integer;**
pdwDeveloperID: PLongint): Longint; stdcall;
function CBIOS_GetDeveloperID(pdwDeveloperID: PLongint): Longint; stdcall;

Visual Basic syntax: **Function CBIOS_GetDeveloperIDI(ByRef pdwDeveloperID As Long)**
As Long
Function CBIOS_GetDeveloperID(ByRef pdwDeveloperID As Long) As Long

Returns CRYPTO-BOX Developer ID:
CBIOS_GetDeveloperIDI – for the CRYPTO-BOX with the specified Index;
CBIOS_GetDeveloperID – for the currently opened CRYPTO-BOX.

Since July 2005 CRYPTO-BOX USB hardware distributed to customers contain hard coded Developer ID - a unique value for every MARX customer (all CRYPTO-BOX modules of this customer have the same value of DeveloperID).
The returned value (DWORD) includes: unique Developer ID for every MARX customer all over the world.
For CRYPTO-BOX USB formatted before July 2005 zero value will be returned as Developer ID.

CBIOS_GetBoxInfoAdvI() allows to to get information about any one of attached CRYPTO-BOX units before opening it. CBIOS_GetBoxInfoAdv assumes that the CRYPTO-BOX was already opened.

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # from 1 to <Box Quantity> value, returned by the preceding CBIOS_ScanBoxes() call
DWORD* pdwDeveloperID	OUT: Points to the DWORD, where CRYPTO-BOX DeveloperID is placed

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox)
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (invalid iBoxIndex)

Or standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h)

CBIOS_GetBoxInfoAdvI/CBIOS_GetBoxInfoAdv calls also return Developer ID included to the BOX_INFO_ADV structure.

DWORD WINAPI CBIOS_GetSerialNumI(INT32 iBoxIndex, BYTE bSerNum[16])
DWORD WINAPI CBIOS_GetSerialNum(BYTE* pSerNum[16])

Delphi syntax: **function CBIOS_GetSerialNumI(iBoxIndex: Integer; SerNum: PTSerNum): Longint; stdcall;**
function CBIOS_GetSerialNum(SerNum: PTSerNum): Longint; stdcall;

Visual Basic syntax: **Function CBIOS_GetSerialNumI(ByVal iBoxIndex As Integer, ByVal SerNum As CBIOS_BYTEARRAY16) As Long**
Function CBIOS_GetSerialNum (ByRef bSerNum As CBIOS_BYTEARRAY16) As Long

Returns the Serial Number of the CRYPTO-BOX:
CBIOS_GetSerialNumI – for the CRYPTO-BOX with the specified Index;
CBIOS_GetSerialNum – for the currently opened CRYPTO-BOX.

CBIOS_GetSerialNumI() allows to to get the Serialnumber from any one of attached CRYPTO-BOX units before opening it. CBIOS_GetSerialNum() assumes that the CRYPTO-BOX was already opened.

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # from 1 to <Box Quantity> value, returned by the preceding CBIOS_ScanBoxes() call
BYTE bSerNum[16]	OUT: Points to the structure, where the CRYPTO-BOX Serial Number will be stored

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox)
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (invalid iBoxIndex)
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened (CBIOS_Open).

Or standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h)

DWORD WINAPI CBIOS_GetAppInfoI(INT32 iBoxIndex, INT32 iAppIndex,
CBIOS_APP_INFO* pAppInfo)
DWORD WINAPI CBIOS_GetAppInfo(INT32 iAppIndex, CBIOS_APP_INFO* pAppInfo)

Delphi syntax: **function CBIOS_GetAppInfoI(iBoxIndex: Integer; iAppIndex: Integer;**
pAppInfo: PCBIOS_APP_INFO): Longint; stdcall;
function CBIOS_GetAppInfo(iAppIndex: Integer; pAppInfo: PCBIOS_APP_INFO): Longint; stdcall;

Visual Basic syntax: **Function CBIOS_GetAppInfoI (ByVal iBoxIndex As Integer,**
ByVal iAppIndex As Integer,
ByRef pAppInfo As CBIOS_APP_INFO) As Long
Function CBIOS_GetAppInfo (ByVal iAppIndex As Integer,
ByRef pAppInfo As CBIOS_APP_INFO) As Long

Returns information about the specified application on the CRYPTO-BOX:
CBIOS_GetAppInfoI – for the CRYPTO-BOX with the specified Index;
CBIOS_GetAppInfo – for the currently opened CRYPTO-BOX.
It allows to find out if an application-related partition is defined in the CRYPTO-BOX or not (otherwise, it permits browsing through defined partitions).

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # starting from 1 to <Box Quantity> value, returned by the preceding CBIOS_ScanBoxes() call
INT32 iAppIndex	IN: Application number starting from 1 to the <wApp> value, returned by the previous CBIOS_GetBoxInfoI() call
CBIOS_APP_INFO* pAppInfo	OUT: Points to the structure, where application information will be placed

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox)
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (invalid iBoxIndex)
CBIOS_ERR_OUT_OF_RANGE	Invalid iAppIndex

DWORD WINAPI CBIOS_GetDriverLastError()

Delphi syntax: **function CBIOS_GetDriverLastError: Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_GetDriverLastError () As Long**

Returns the last error code for the currently opened CRYPTO-BOX.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.4. Opening and closing the CRYPTO-BOX®

DWORD WINAPI CBIOS_OpenByIndex(INT32 iBoxIndex, WORD wAppID)

Delphi syntax: **function CBIOS_OpenByIndex(iBoxIndex: Integer; wAppID: Word): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_OpenByIndex(ByVal iBoxIndex As Integer, ByVal wAppID As Integer) As Long**

Opens CRYPTO-BOX for specified application (wAppID), using Index.
Using the **Open by Index** approach makes only sense, if application logic assumes that only one CRYPTO-BOX is attached to the PC at a time. Application logic should not rely on the Index number returned alone: after hardware detach/attach events, the number may change. Use **CBIOS_OpenByName()** if it is necessary to open a CRYPTO-BOX with a known name (dwBoxName).

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # starting from 1 to <Box Quantity> value, returned by the preceding CBIOS_ScanBoxes() call
WORD wAppID	IN: Application Identifier. The corresponding partition must exist in this CRYPTO-BOX. It is possible to specify “0”. This permits working with the CRYPTO-BOX (for example, to change BoxLabel, etc.) , but there is no access to its memory.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (invalid iBoxIndex)
CBIOS_ERR_BOX_ALREADY_OPENED	This thread has already opened a CRYPTO-BOX. To open this one it is necessary to close currently opened CRYPTO-BOX (CBIOS_Close).
CBIOS_ERR_APP_NOT_FOUND	This CRYPTO-BOX (iBoxIndex) contains no partition with the specified application (wAppID)
CBIOS_ERR_TOO_MANY_BOXES_OPEN	Too many CRYPTO-BOXes are opened (more than 32 simultaneously on this computer)

After opening the CRYPTO-BOX, Smarx OS records the fact that this CRYPTO-BOX is open for the current thread and specified application (wAppID). That means that all further CBIOS calls from the thread will work only with this application partition.

DWORD WINAPI CBIOS_OpenByName(DWORD dwBoxName, WORD wAppID)

Delphi syntax: **function CBIOS_OpenByName(dwBoxName: Longint; wAppID: Word): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_OpenByName (ByVal dwBoxName As Long, ByVal wAppID As Integer) As Long**

Opens CRYPTO-BOX for specified application (wAppID), using dwBoxName.

Parameters:

DWORD dwBoxName	IN: dwBoxName
WORD wAppID	IN: Application Identifier. The corresponding partition must exist in this CRYPTO-BOX. It is possible to specify “0”. This permits working with the CRYPTO-BOX (for example, to change BoxLabel, etc.) , but there is no access to its memory.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (there is no CRYPTO-BOX with specified dwBoxName)
CBIOS_ERR_BOX_ALREADY_OPENED	This thread has already opened a CRYPTO-BOX. To open this one it is necessary to close currently opened CRYPTO-BOX (CBIOS_Close).

CBIOS_ERR_APP_NOT_FOUND	There is no partition in this CRYPTO-BOX (dwBoxName) for the specified application (wAppID)
CBIOS_ERR_TOO_MANY_BOXES_OPEN	Too many CRYPTO-BOX are opened (more than 32 simultaneously on this computer)

After opening the CRYPTO-BOX, Smarx OS records the fact that this CRYPTO-BOX is open for the current thread and specified application (wAppID). That means that all further CBIOS calls from the thread will work only with this application partition.

DWORD WINAPI CBIOS_OpenByLabel(const BYTE bBoxLabel[CBIOS_LABEL_LEN], WORD wAppID)

Delphi syntax: **function CBIOS_OpenByLabel(const bBoxLabel: PTBoxLabel; wAppID: Word): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_OpenByLabel (ByRef bBoxLabel As CBIOS_BYTEARRAY16, ByVal wAppID As Integer) As Long**

Opens CRYPTO-BOX for specified application (wAppID), using Volume Label.

Parameters:

BYTE bBoxLabel[CBIOS_LABEL_LEN]	IN: bBoxLabel - Volume Label of the CRYPTO-BOX to open
WORD wAppID	IN: Application Identifier. The corresponding partition must exist in this CRYPTO-BOX. It is possible to specify “0”. This permits working with the CRYPTO-BOX (for example, to change BoxLabel, etc.) , but there is no access to its memory.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (there is no CRYPTO-BOX with specified Volume Label)
CBIOS_ERR_BOX_ALREADY_OPENED	This thread has already opened a CRYPTO-BOX. To open this one it is necessary to close currently opened CRYPTO-BOX (CBIOS_Close).
CBIOS_ERR_APP_NOT_FOUND	There is no partition in this CRYPTO-BOX (dwBoxName) for the specified application (wAppID)
CBIOS_ERR_TOO_MANY_BOXES_OPEN	Too many CRYPTO-BOX are opened (more than 32 simultaneously on this computer)

If more than one CRYPTO-BOX with the same BoxLabel is attached, then the CRYPTO-BOX with the lowest index will be open (the index is defined by operating system and depends, for

example, on the sequence of CRYPTO-BOX attachments).

After opening the CRYPTO-BOX, Smarx OS records the fact that this CRYPTO-BOX is open for the current thread and specified application (wAppID). That means that all further CBIOS calls from the thread will work only with this application partition.

DWORD WINAPI CBIOS_OpenByApp(WORD wAppID)

Delphi syntax: **function CBIOS_OpenByApp(wAppID: Word): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_OpenByApp(ByVal wAppID As Integer) As Long**

Opens CRYPTO-BOX for the application partition specified (wAppID)

Parameters:

WORD wAppID	IN: Application Identifier. The corresponding partition must exist in this CRYPTO-BOX. It is possible to specify "0". This permits working with the CRYPTO-BOX (for example, to change BoxLabel, using encryption functions, etc.), but there is no access to its memory.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_BOX_NOT_FOUND	CRYPTO-BOX not found (there is no CRYPTO-BOX with specified wAppID).
CBIOS_ERR_BOX_ALREADY_OPENED	This thread has already opened a CRYPTO-BOX. To open this one, the currently opened CRYPTO-BOX has to be closed first (CBIOS_Close).
CBIOS_ERR_TOO_MANY_BOXES_OPEN	Too many CRYPTO-BOXes are opened (more than 32 simultaneously on this computer).

If more than one attached CRYPTO-BOX has a partition for the specified application, then the one with the lower index number will be opened (the index is defined by operating system and depends, for example, on the sequence in which the CRYPTO-BOXes were attached).

After opening the CRYPTO-BOX, Smarx OS records the fact that this CRYPTO-BOX is open for the current thread and specified application (wAppID). That means that all further CBIOS calls from the thread will work only with this application partition.

DWORD WINAPI CBIOS_Close()

Delphi syntax: **function CBIOS_Close: Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_Close() As Long**

Closes the CRYPTO-BOX for the current thread.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened.

1.5. Locking an open CRYPTO-BOX®

DWORD WINAPI CBIOS_LockBox()

Delphi syntax: **function CBIOS_LockBox(): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_LockBox () As Long**

Locks the CRYPTO-BOX (opened in the current thread). Other threads/processes will wait in the queue with 20 seconds (timeout) and will return CBIOS_ERR_LOCK_TIMEOUT if the CRYPTO-BOX is not unlocked before timeout is reached.
To unlock CRYPTO-BOX call CBIOS_UnlockBox.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds.
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened

DWORD WINAPI CBIOS_LockBoxEx(DWORD dwTimeoutMsec);

Delphi syntax: **function CBIOS_LockBoxEx(dwTimeoutMsec:Longint): Longint; stdcall;**

Visual Basic syntax: **not implemented**

Locks the CRYPTO-BOX (opened in the current thread). Other threads/processes will wait in the queue with **dwTimeoutMsec milli**seconds (timeout) and will return CBIOS_ERR_LOCK_TIMEOUT if the CRYPTO-BOX is not unlocked before timeout is reached. To unlock CRYPTO-BOX call CBIOS_UnlockBox.

Parameters:

DWORD dwTimeoutMsec	IN: Timeout in milliseconds

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than **dwTimeoutMsec** milliseconds.
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened

DWORD WINAPI CBIOS_UnlockBox()

Delphi syntax: **function CBIOS_UnlockBox(): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_UnlockBox () As Long**

Unlocks the CRYPTO-BOX locked by CBIOS_LockBox.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened

BOOL WINAPI CBIOS_IsBoxLockedByOthersI(INT32 iBoxIndex);
BOOL WINAPI CBIOS_IsBoxLockedByOthers(void);

*Delphi syntax:***function CBIOS_IsBoxLockedByOthersI(iBoxIndex: Integer) : BOOL; stdcall;**
function CBIOS_IsBoxLockedByOthers() : BOOL; stdcall;

Visual Basic syntax: **not implemented**

Checks if the CRYPTO-BOX is locked or not by other thread or process.

CBIOS_IsBoxLockedByOthersI – on the CRYPTO-BOX Index;

CBIOS_IsBoxLockedByOthers – on the currently opened CRYPTO-BOX.

Parameters:

INT32 iBoxIndex	IN: CRYPTO-BOX # starting from 1 to <Box Quantity> value, returned by the preceding CBIOS_ScanBoxes() call

Return:
TRUE – Locked, FALSE – Unlocked

DWORD WINAPI CBIOS_LockLicence()

Delphi syntax: **function CBIOS_LockLicence(): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_LockLicence() As Long**

CBIOS_LockLicence() call gives exclusive access to the open partition, locking it for any other applications/threads. After this call no other application can open the partition using CBIOS_OpenBy###(), any attempt to open it will result in CBIOS_ERR_NO_FREE_LICENCE error code. Exclusive access mode will be canceled only after the application that uses this partition exclusively issues CBIOS_ReleaseLicense() or closes the session using CBIOS_Close().
CBIOS_LockLicence()/CBIOS_ReleaseLicence() can be used to prevent the launch of several copies of a protected application trough Terminal Server (see *SmarxOS Compendium, chapter 11.2.8* for more details).
CBIOS_LockLicence() locks only the currently open partition. Other partitions of the CRYPTO-BOX are still available for other applications/processes, the same is true for hardware encryption functionality.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds.
CBIOS_ERR_BOX_ALREADY_OPENED	This thread has already opened a CRYPTO-BOX.

DWORD WINAPI CBIOS_LockLicenceExt()

Delphi syntax: **function CBIOS_LockLicenceExt(): Longint; stdcall;**

Visual Basic syntax: ***not implemented***

Note: CBIOS_LockLicence is strongly recommended !

CBIOS_LockLicenceExt() is similar to CBIOS_LockLicence function. It is more flexible, allowing usage of extended network licensing rules (pre-programmed to the CRYPTO-BOX) for

license locking. From network licensing standpoint this function is less secure comparing to the standard one. Be careful with extended network licensing and use its rules only when it is required by your licensing approach. The extended rules are described in **CBIOS_SetRuleAppLicences** function.

Parameters: None

Return:

CBIOS_ERR_OUT_OF_RANGE	Invalid rule ID

DWORD WINAPI CBIOS_ReleaseLicence()

Delphi syntax: **function CBIOS_ReleaseLicence(): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_ReleaseLicence () As Long**

Unlocks the current partition so that other applications (processes/threads) can access it.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds.
CBIOS_ERR_BOX_ALREADY_OPENED	This thread has already opened a CRYPTO-BOX.

1.6. Check if the CRYPTO-BOX® is still attached

1.6.1. Check presence of the currently opened CRYPTO-BOX®

DWORD WINAPI CBIOS_CheckBox()

Delphi syntax: **function CBIOS_CheckBox: Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_CheckBox () As Long**

Checks for the presence of the currently opened CRYPTO-BOX.

CBIOS.LIB does not automatically control CRYPTO-BOX attachments/detachments. This function permits checking, if the CRYPTO-BOX opened with CBIOS_Open call is still attached.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened (CBIOS_Open).

1.6.2. Using event notifications to check the CRYPTO-BOX® presence

DWORD WINAPI CBIOS_RegisterNotificationCallback(F_CBIOS_NOTIFY_CALLBACK* fNotify, void* pParam)

Delphi syntax: **function CBIOS_RegisterNotificationCallback(fNotify: PF_CBIOS_NOTIFY_CALLBACK; dwParam: longint): longint; stdcall;**

Visual Basic syntax: **not supported**

This call registers the CBIOS notification handler: **fNotify** callback function. The function will receive notifications on CRYPTO-BOX USB attach/detach events.

Parameters:

F_CBIOS_NOTIFY_CALLBACK* fNotify	IN: Pointer to the callback function
void* pParam	IN: User specific data. This value will be sent to the fNotify callback function for every notification call.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_WRONG_PARAM	fNotify pointer is NULL

The following is an example of such a callback function definition:
DWORD CALLBACK cbios_callback(CBIOS_NOTIFY_DATA NotificationData);
One application may register more than one notification handler. This can be useful for DLLs and static libraries.
Notifications are sent as a special structure **CBIOS_NOTIFY_DATA**:

```
typedef struct {
        void* pNotificationParam;
        DWORD dwNotificationID;
        DWORD dwNotificationType;
} CBIOS_NOTIFY_DATA;
```

DWORD	**dwNotificationID**	Notification identifier - this value is incremented for every notification
DWORD	**dwBNotification Type**	Possible values:

		1 – CBIOS_BOX_ATTACHED for local mode 2 – CBIOS_BOX_REMOVED for local mode 3 – CBIOS_BOXES_CHANGED for network
void*	**pNotificationParam**	This variable contains the pParam value that was submitted during registration of the handler using CBIOS_RegisterNotificationCallback()

Besides receiving notifications to the callback function it is also possible to get them as window messages – see: CBIOS_RegisterNotificationMessage()

DWORD WINAPI CBIOS_UnRegisterNotificationCallback (F_CBIOS_NOTIFY_CALLBACK* fNotify);

Delphi syntax: **function CBIOS_UnRegisterNotificationCallback(fNotify: PF_CBIOS_NOTIFY_CALLBACK): longint; stdcall;**

Visual Basic syntax: **not supported**

This call unregisters CBIOS notification handler: **fNotify** callback function.

Parameters:

F_CBIOS_NOTIFY_CALLBACK* fNotify	IN: Pointer to the callback function

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_WRONG_PARAM	fNotify pointer is NULL

DWORD WINAPI CBIOS_RegisterNotificationMessage(HWND hWnd, UINT message, LPARAM lParam)

Delphi syntax: **function CBIOS_RegisterNotificationMessage (hWnd:HWND; msg, lParam: longint): longint; stdcall;**

Visual Basic syntax: **not supported**

This call registers window (hWnd) to receive notification messages on CRYPTO-BOX USB attach/detach events.

Parameters:

HWND hWnd	IN: Handle of the window to receive notification messages
UNIT message	IN: The message code to be sent, usually

	WM_USER+N, where N is 1 or larger (depending on the application logic)
LPARAM lParam	IN: User specific data. This value will be sent to the hWnd window with every notification message.

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_WRONG_PARAM	hWnd is NULL

One application may register more than one notification. It can be useful for DLLs and static libraries.
The notification message received by the hWnd window includes the following information:

UINT	**message**	Message number specified in **CBIOS_RegisterNotificationMessage()**
DWORD	**wParam**	Possible values: 1 – CBIOS_BOX_ATTACHED 2 – CBIOS_BOX_REMOVED
LPARAM	**lParam**	This variable contains lParam value, specified in CBIOS_RegisterNotificationMessage()

DWORD WINAPI CBIOS_UnRegisterNotificationMessage(HWND hWnd);

Delphi syntax: **function CBIOS_UnRegisterNotificationMessage(hWnd:HWND): longint; stdcall;**

Visual Basic syntax: **not supported**

This call unregisters hWnd window from receiving CBIOS notification messages.

Parameters:

HWND hWnd	IN: Pointer to the callback function

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_WRONG_PARAM	fNotify pointer is NULL

Besides receiving notifications as window messages it is also possible to get them to the callback function – see: CBIOS_RegisterNotificationCallback().

1.7. Working with the open CRYPTO-BOX®

1.7.1. Setting the Label

DWORD WINAPI CBIOS_SetLabel(const BYTE bBoxLabel[CBIOS_LABEL_LEN])

Delphi syntax: **function CBIOS_SetLabel(const bBoxLabel: PTBoxLabel): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetLabel (ByRef ConstLabel As CBIOS_BYTEARRAY_LABEL) As Long**

Sets Volume Label for currently opened CRYPTO-BOX.

Parameters:

BYTE bBoxLabel[CBIOS_LABEL_LEN]	IN: New Volume Label

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.2. Login/Logout

DWORD WINAPI CBIOS_UPWLogin(BYTE bUPW[0x10])

Delphi syntax: **function CBIOS_UPWLogin(bUPW: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_UPWLogin(ByRef bUPW As CBIOS_BYTEARRAY16) As Long**

User level login to the opened CRYPTO-BOX.

In case of successful login, the CRYPTO-BOX will be automatically locked for the current thread, until CBIOS_Logout() is called. All calls outside of this thread will be queued for 20 seconds.

Parameters:

BYTE bUPW[0x10]	IN: User Password (UPW)

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_ALREADY_LOGGED	CRYPTO-BOX is already logged. CBIOS_Logout has to be called first.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_APWLogin(BYTE bAPW[0x10])

Delphi syntax: **function CBIOS_APWLogin(bAPW: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_APWLogin(ByRef bAPW As CBIOS_BYTEARRAY16) As Long**

Administrator level login (APW) to the opened CRYPTO-BOX.

In case of successful login, the CRYPTO-BOX will be automatically locked for the current thread, until CBIOS_Logout() is called. All calls outside of this thread will be queued for 20 seconds.

Parameters:

BYTE bAPW[0x10]	IN: Admin Password(APW)

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_ALREADY_LOGGED	CRYPTO-BOX is already logged. CBIOS_Logout has to be called first.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_Logout()

Delphi syntax: **function CBIOS_Logout: Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_Logout() As Long**

Performs logout after CBIOS_UPWLogin or CBIOS_APWLogin calls.

Parameters: None

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.3. Using the random number generator

DWORD WINAPI CBIOS_GetHWRand(DWORD dwLen, PVOID pBuffer)

Delphi syntax: **function CBIOS_GetHWRand(dwLen: Longint; pBuffer: Pointer): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_GetHWRand (ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY) As Long**

Retrieves hardware random bit stream.

Parameters:

DWORD dwLen	IN: Length of data pool in bytes
PVOID pBuffer	OUT: Random data pool pointer

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open.
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.4. Setting User and Administrator Password

If you change User and/or Administrator Password, you will be no longer able to use the Smarx Application Framework (SxAF) GUI tool for CRYPTO-BOX formatting, since it is using standard values from TRX file! The SmrxProg.exe command line tool can handle modified UPW values by adding the <UPW OPERATION="USE"> tag. More details can be found in the SmrxProg readme file.

DWORD WINAPI CBIOS_SetUPW(BYTE bOldPass[0x10], BYTE bNewPass[0x10])

Delphi syntax: **function CBIOS_SetUPW(bOldPass: PTPasswd; bNewPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetUPW(ByRef bOldPass As CBIOS_BYTEARRAY16, ByRef bNewPass As CBIOS_BYTEARRAY16) As Long**

Sets User Password.

Parameters:

BYTE bOldPass[0x10]	IN: Old User or Admin Password
BYTE bNewPass[0x10]	IN: New User Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open.
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_SetAPW(BYTE bOldPass[0x10], BYTE bNewPass[0x10])

Delphi syntax: **function CBIOS_SetAPW(bOldPass: PTPasswd; bNewPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetAPW(ByRef bOldPass As CBIOS_BYTEARRAY16, ByRef bNewPass As CBIOS_BYTEARRAY16) As Long**

Sets Administrator Password.

Parameters:

BYTE bOldPass[0x10]	IN: Old Admin Password
BYTE bNewPass[0x10]	IN: New Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.5. Symmetric AES encryption

1.7.5.1. Standard functionality (all CRYPTO-BOX® models)

DWORD WINAPI CBIOS_SetKeySession(BYTE bKEY[0x10])

Delphi syntax: **function CBIOS_SetKeySession(bKEY: PTByteArr16): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetKeySession(ByRef bKEY As CBIOS_BYTEARRAY16) As Long**

Sets key for session hardware encryption/decryption.

Parameters:

BYTE bKey[0x10]	IN: Pointer to Key bit data

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_SetIVSession(BYTE bIV[0x10])

Delphi syntax: **function CBIOS_SetIVSession(bIV: PTByteArr16): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetIVSession(ByRef bIV As CBIOS_BYTEARRAY16) As Long**

Sets initialization vector for session hardware encryption/decryption.

Parameters:

BYTE bIV[0x10]	IN: Pointer to Initialization Vector bit data

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_SetKeyPrivate(BYTE bAPW[0x10], BYTE bKEY[0x10])

Delphi syntax: **function CBIOS_SetKeyPrivate(bAPW: PTPasswd; bKEY: PTByteArr16): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetKeyPrivate(ByRef bAPW As CBIOS_BYTEARRAY16, ByRef bKEY As CBIOS_BYTEARRAY16) As Long**

Sets private key for hardware encryption/decryption.

Parameters:

BYTE bKey[0x10]	IN: Pointer to Key bit data

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user

	(CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_SetIVPrivate(BYTE bAPW[0x10], BYTE bIV[0x10])

Delphi syntax: **function CBIOS_SetIVPrivate(bAPW: PTPasswd; bIV: PTByteArr16): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetIVPrivate(ByRef bAPW As CBIOS_BYTEARRAY16, ByRef bIV As CBIOS_BYTEARRAY16) As Long**

Sets private initialization vector for hardware encryption/decryption.

Parameters:

BYTE bIV[0x10]	IN: Pointer to Initialization Vector bit data

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_CryptFixed(DWORD dwLen, PVOID pBuffer)

Delphi syntax: **function CBIOS_CryptFixed(dwLen: Longint; pBuffer: Pointer): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_CryptFixed(ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY) As Long**

Encrypts/decrypts data using internal hardware algorithm with fixed key and fixed initialization vector. The Fixed Key and its IV is hard coded and cannot be reprogrammed. Every MARX customer has the same values for hardware Rijndael keys initially programmed to all his CRYPTO-BOX units.

Parameters:

DWORD dwLen	IN: Data pool length
PVOID pBuffer	IN/OUT: Pointer to data pool

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_CryptPrivate(DWORD dwLen, PVOID pBuffer)

Delphi syntax: **function CBIOS_CryptPrivate(dwLen: Longint; pBuffer: Pointer): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_CryptPrivate (ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY) As Long**

Encrypts/decrypts data using internal hardware algorithm with private key and private initialization vector (IV). The Private Key (and its IV) requires the User Password for usage and the Admin Password for reprogramming.

Parameters:

DWORD dwLen	IN: Data pool length
PVOID pBuffer	IN/OUT: Pointer to data pool

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_BOX_NOT_LOGGED	There is currently no login to CRYPTO-BOX at user (CBIOS_UPWLogin) or administrator (CBIOS_APWLogin) level.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_CryptSession(DWORD dwLen, PVOID pBuffer)

Delphi syntax: **function CBIOS_CryptSession(dwLen: Longint; pBuffer: Pointer): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_CryptSession(ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY) As Long**

Encrypts/decrypts data using internal hardware algorithm with session key and session initialization vector (IV). The Session Key (and its IV) can be used and reprogrammed by any application without password submission.

Parameters:

DWORD dwLen	IN: Data pool length
PVOID pBuffer	IN/OUT: Pointer to data pool

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.5.2. Extended functionality and CBC mode (CRYPTO-BOX® SC only)

DWORD WINAPI CBIOS_CBU2_SetKeyAES(DWORD dwKeyIndex,
CBIOS_AES_KEY* pAESKey,
CBIOS_AES_KEY_INFO * pAESKeyInfo);

dwKeyIndex	[in] Index of AES cell in CBU SC RAM5 zone for this key (starting from 0)
pAESKey	[in] Pointer to AES key value in computer memory
pAESKeyInfo	[in] Pointer to a structure with AES key information (optional)
[return]	CBIOS Error code.

Description: This function writes CBU SC AES key value (**pAESKey**) to one of AES key holding cells (**dwKeyIndex)** in dedicated CBU SC memory zone (RAM5).

The **pAESKeyInfo** structure can optionally set access rights for this key, including: its usage for encryption; obtaining access rights info and/or changing the key value/access rights.

See section *1.7.5.3* for more details on how to define access rights and limitations using CBIOS_AES_KEY_INFO structure.

Example: the *Test_AES ()* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx OS PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_SetKeyInfoAES(DWORD dwKeyIndex, CBIOS_AES_KEY_INFO* pAESKeyInfo);

dwKeyIndex	[in] Index of AES cell in CBU SC RAM5 holding the key (starting from 0)
pAESKeyInfo	[in] Pointer to new AES key information
[return]	CBIOS Error code.

Description: This function changes access rights and limitations for CBU SC AES cell (**dwKeyIndex)** to values defined in the **pAESKeyInfo** structure.

It can change access rights for this key on: its usage for encryption, obtaining access rights info and/or changing the key value/access rights. See section *1.7.5.3* for more details on how to define access rights and limitations in CBIOS_AES_KEY_INFO structure.

Example: the *Test_AES ()* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_GetKeyInfoAES(DWORD dwKeyIndex, CBIOS_AES_KEY_INFO* pAESKeyInfo);

dwKeyIndex	[in] Index of AES cell in CBU SC RAM5 zone storing the key (starting from 0)
pAESKeyInfo	[in] Pointer to the structure that will receive AES key information
[return]	CBIOS Error code.

Description: This function retrieves access rights and limitations data for CBU SC AES cell (**dwKeyIndex)** to the **pAESKeyInfo** structure. See section *1.7.5.3* for more details on how to define/interpret access rights and limitations in CBIOS_AES_KEY_INFO structure.

Example: the *Test_AES (void)* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_LockKeyAES(DWORD dwKeyIndex);

dwKeyIndex	[in] Index of CBU SC AES cell storing the key (starting from 0)
[return]	CBIOS Error code.

Description: This function locks the CBU SC AES cell (**dwKeyIndex),** so this key can not be used for further AES encryption/decryption operations.

DWORD WINAPI CBIOS_CBU2_CryptAES(DWORD dwKeyIndex, DWORD dwMode,
PVOID pIV,
PVOID pInBuffer,
PVOID pOutBuffer,
DWORD dwBufferLen);

dwKeyIndex	[in] Index of AES key storage cell in CBU SC RAM5 (starting from 0)
dwMode	[in] Defines encryption mode. One of the following values: CBIOS_AES_OFB – Encrypt/decrypt with OFB mode CBIOS_AES_CBC_ENCRYPT – Encrypt with CBC mode CBIOS_AES_CBC_DECRYPT – Decrypt with CBC mode
pIV	Pointer to AES initialization vector value in computer memory
pInBuffer	[in] Pointer to input buffer in computer memory (points to byte array to encrypt).
pOutBuffer	[in] Pointer to a buffer that will receive encrypted data from pInBuffer.
dwBufferLen	[in] Size in bytes of pInBuffer/pOutBuffer .
[return]	CBIOS Error code.

Description: This function performs CBU SC hardware based AES encryption using **dwKeyIndex** cell in RAM5 CBU SC memory as AES key holder.

The **dwMode** parameter defines which mode of encryption/decryption should be used: CBC of OFB.
The input data (byte array **pInBuffer** of length **dwBufferLen**) will be encrypted and the result of this operation will be placed to the **pOutBuffer**.

The AES algorithm logic assumes that encrypted buffer size is always the same as source buffer size. When using the CBC mode this size must be multiple of the key size (16 bytes).

Example: The *Test_AES ()* and *Test_AES_SW ()* functions of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) demonstrate this call usage.

1.7.5.3. New CBIOS structures related to extended AES

CBIOS_AES_KEY

This structure contains AES key or AES public part.

```
typedef struct {
        DWORD dwStructSize;
    BYTE  ubKey[CBIOS_AES_KEY_LEN];
} CBIOS_AES_KEY;
```

dwStructSize	[in] Size of structure in bytes.
ubKey	[in,out] AES key

The **CBIOS_AES_KEY** structure is used in the following CBIOS call:
CBIOS_CBU2_SetKeyAES()

CBIOS_AES_KEY_INFO

This structure defines access rights and limitations for CBU SC AES key: its usage for encryption, obtaining access rights info and/or changing the key value/access rights.

```
typedef struct {
        DWORD dwStructSize;
        DWORD dwAccess;   // low nibble:   SetKey/SetKeyInfo access rights,
                          // high nibble:  Encrypt/Decrypt and GetKeyInfo
        WORD  wBits;
} CBIOS_AES_KEY_INFO;
```

dwStructSize	[in] Size of structure in bytes.
dwAccess	[in,out] AES key access flags. Lower 4 bits define access rights for changing AES key (SetKey and SetKeyInfo operations). Higher 4 bits define access rights for performing AES encryption, decryption and obtaining key information. Supported values: CBIOS_CBU2_ACCESS_NEVER CBIOS_CBU2_ACCESS_ALWAYS CBIOS_CBU2_ACCESS_UPW CBIOS_CBU2_ACCESS_APW CBIOS_CBU2_ACCESS_LOCK
wBits	[out] AES key size in bits (key strength).

Comment: The **dwAccess** member of this structure defines required access rights or returns current access rights for AES key. For any CBU SC AES key access rights can be set for:

- encryption/decryption and obtaining information on this key (key strength and current

access rights)
- changing the key value

Supported values and their meaning:

- **CBIOS_CBU2_ACCESS_NEVER** – this value (if being set) can not be reset even by MARX distributor. MARX distributor can only reprogram the whole RAM4 zone of this CBU SC unit, so all current values of AES keys (if any) will be lost
- **CBIOS_CBU2_ACCESS_ALWAYS** – this value means free access (no limitations).
- **CBIOS_CBU2_ACCESS_UPW** – UPW login is required
- **CBIOS_CBU2_ACCESS_APW** – APW login is required
- **CBIOS_CBU2_ACCESS_LOCK** – the access will be locked, only MARX distributor can unlock it.

The **CBIOS_AES_KEY_INFO** structure is used in the following CBIOS calls:
- **CBIOS_CBU2_SetKeyAES()**
- **CBIOS_CBU2_SetKeyInfoAES()**
- **CBIOS_CBU2_GetKeyInfoAES()**

1.7.6. Read/write CRYPTO-BOX® internal memory

DWORD WINAPI CBIOS_ReadRAM1(DWORD dwAddress, DWORD dwLen, PVOID pBuffer, BYTE bPass[0x10])

Delphi syntax: **function CBIOS_ReadRAM1(dwAddress: Longint; dwLen: Longint; pBuffer: Pointer; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_ReadRAM1(ByVal dwAddress As Long, ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Reads data from RAM1.

Parameters:

DWORD dwAddress	IN: data address in RAM1
DWORD dwLen	IN: Data length in RAM1
PVOID pBuffer	OUT: Pointer to data pool
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the

	application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_WriteRAM1(DWORD dwAddress, DWORD dwLen, PVOID pBuffer, BYTE bPass[0x10])

Delphi syntax: **function CBIOS_WriteRAM1(dwAddress: Longint; dwLen: Longint; pBuffer: Pointer; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_WriteRAM1(ByVal dwAddress As Long, ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Writes data to RAM1.

Parameters:

DWORD dwAddress	IN: Data address in RAM1
DWORD dwLen	IN: Data length in RAM1
PVOID pBuffer	IN: Pointer to data pool
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_ReadRAM2(DWORD dwAddress, DWORD dwLen, PVOID pBuffer, BYTE bPass[0x10])

Delphi syntax: **function CBIOS_ReadRAM2(dwAddress: Longint; dwLen: Longint; pBuffer: Pointer; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_ReadRAM2(ByVal dwAddress As Long, ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Reads data from RAM2.

Parameters:

DWORD dwAddress	IN: Data address in RAM2
DWORD dwLen	IN: Data length in RAM2
PVOID pBuffer	OUT: Pointer to data pool
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_WriteRAM2(DWORD dwAddress, DWORD dwLen, PVOID pBuffer, BYTE bPass[0x10])

Delphi syntax: **function CBIOS_WriteRAM2(dwAddress: Longint; dwLen: Longint; pBuffer: Pointer; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_WriteRAM2(ByVal dwAddress As Long, ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Writes data to RAM2.

Parameters:

DWORD dwAddress	IN: Data address in RAM2
DWORD dwLen	IN: Data length in RAM2
PVOID pBuffer	IN: Pointer to data pool
BYTE bPass[0x10]	IN: Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see

cbios.h).

In contrast to the User Password (UPW), the Admin Password (APW) is not cached, therefore the APW needs to be submitted for each function call requiring it (e.g. CBIOS_WriteRAM2).

We strongly recommend to consider RAM2 as read-only area to keep the the Admin Password (APW) secret (never included to the protected application).

DWORD WINAPI CBIOS_ReadRAM3(DWORD dwAddress, DWORD dwLen, PVOID pBuffer, BYTE bPass[0x10])

Delphi syntax: **function CBIOS_ReadRAM3(dwAddress: Longint; dwLen: Longint; pBuffer: Pointer; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_ReadRAM3(ByVal dwAddress As Long, ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Reads data from RAM3.

Parameters:

DWORD dwAddress	IN: Data address in RAM3
DWORD dwLen	IN: Data length in RAM3
PVOID pBuffer	OUT: Pointer to data pool
BYTE bPass[0x10]	IN: - reserved

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_WriteRAM3(DWORD dwAddress, DWORD dwLen, PVOID pBuffer, BYTE bPass[0x10])

Delphi syntax: **function CBIOS_WriteRAM3(dwAddress: Longint; dwLen: Longint; pBuffer: Pointer; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_WriteRAM3(ByVal dwAddress As Long, ByVal dwLen As Long, ByRef pBuffer As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Writes data to RAM3.

Parameters:

DWORD dwAddress	IN: Data address in RAM3
DWORD dwLen	IN: Data length in RAM3
PVOID pBuffer	IN: Pointer to data pool
BYTE bPass[0x10]	IN: - reserved

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.7. Asymmetric RSA encryption

1.7.7.1. Standard functionality (all CRYPTO-BOX® models)

DWORD WINAPI CBIOS_EncryptRSA(DWORD dwKeyMemory, DWORD dwKeyOffset, PVOID pInBuffer, DWORD dwInBufferLen, PVOID pOutBuffer, DWORD *dwOutBufferLen, BYTE bPass[16])

Delphi syntax: **function CBIOS_EncryptRSA(dwKeyMemory: Longint; dwKeyOffset: Longint; pInBuffer: Pointer; dwInBufferLen: Longint; pOutBuffer: Pointer; dwOutBufferLen: PLongint; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_EncryptRSA(ByVal dwKeyMemory As Long, ByVal dwKeyOffset As Long, ByRef pInBuffer As CBIOS_BYTEARRAY, ByVal dwInBufferLen As Long, ByRef pOutBuffer As CBIOS_BYTEARRAY, ByVal dwOutBufferLen As Long, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Encrypts data with RSA algorithm.

The CRYPTO-BOX XS and Versa models provide RSA support in software (on driver level). The CRYPTO-BOX SC provides hardware-based RSA (key never leaves the hardware). For more details and CRYPTO-BOX SC specific API functions please refer section *1.7.7.3*.

Parameters:

DWORD dwKeyMemory	IN: Type of memory in which RSA-Key is stored RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwKeyOffset	IN: Key offset in RAM
PVOID pInBuffer	IN: Pointer to original data
DWORD dwInBufferLen	IN: Length of original data
PVOID pOutBuffer	OUT: Pointer to encrypted data
DWORD * dwOutBufferLen	OUT: Pointer to length of encrypted data
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_DecryptRSA(DWORD dwKeyMemory,DWORD dwKeyOffset, PVOID pInBuffer, DWORD dwInBufferLen, PVOID pOutBuffer, DWORD *dwOutBufferLen, BYTE bPass[16])

Delphi syntax: **function CBIOS_DecryptRSA(dwKeyMemory: Longint; dwKeyOffset: Longint; pInBuffer: Pointer; dwInBufferLen: Longint; pOutBuffer: Pointer; dwOutBufferLen: PLongint; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_DecryptRSA(ByVal dwKeyMemory As Long, ByVal dwKeyOffset As Long, ByRef pInBuffer As CBIOS_BYTEARRAY, ByVal dwInBufferLen As Long, ByRef pOutBuffer As CBIOS_BYTEARRAY, ByVal dwOutBufferLen As Long, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Decrypts data with RSA algorithm

The CRYPTO-BOX XS and Versa models provide RSA support in software (on driver level). The CRYPTO-BOX SC provides hardware-based RSA (key never leaves the hardware). For more details and CRYPTO-BOX SC specific API functions please refer section *1.7.7.3*.

Parameters:

DWORD dwKeyMemory	IN: Type of memory in which RSA-Key is stored RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwKeyOffset	IN: Key offset in RAM
PVOID pInBuffer	IN: Pointer to encrypted data
DWORD dwInBufferLen	IN: Length of encrypted data
PVOID pOutBuffer	OUT: Pointer to decrypted data
DWORD * dwOutBufferLen	OUT: Pointer to length of decrypted data
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_GetKeyPublicRSA(DWORD dwKeyMemory, DWORD dwKeyOffset, CBIOS_RSA_PUBLIC_KEY *pRSAKey, BYTE bPass[16])

Delphi syntax: **function CBIOS_GetKeyPublicRSA(dwKeyMemory: Longint; dwKeyOffset: Longint; pRSAKey: PCBIOS_RSA_PUBLIC_KEY; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_GetKeyPublicRSA(ByVal dwKeyMemory As Long, ByVal dwKeyOffset As Long, ByRef pRSAKey As CBIOS_RSA_PUBLIC_KEY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Retrieves public RSA key stored in RAM for software RSA encryption.

Parameters:

DWORD dwKeyMemory	IN: Type of memory where RSA-Key is stored: RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwKeyOffset	IN: Key offset in RAM
CBIOS_RSA_PUBLIC_KEY *pRSAKey	OUT: Pointer to public RSA key structure
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_GetKeyPrivateRSA(DWORD dwKeyMemory, DWORD dwKeyOffset, CBIOS_RSA_PRIVATE_KEY *pRSAKey, BYTE bPass[16])

Delphi syntax: **function CBIOS_GetKeyPrivateRSA(dwKeyMemory: Longint; dwKeyOffset: Longint; pRSAKey:PCBIOS_RSA_PRIVATE_KEY; bPass: PTPasswd): Longint; stdcall;**

*Visual Basic syntax:***Function CBIOS_GetKeyPrivateRSA(ByVal dwKeyMemory As Long, ByVal dwKeyOffset As Long, ByRef pRSAKey As CBIOS_RSA_PRIVATE_KEY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Retrieves private RSA key stored in RAM for software RSA encryption.

Parameters:

DWORD dwKeyMemory	IN: Type of memory where RSA-Key is stored RAM1 = 1 RAM2 = 2 RAM3 = 3
DWORD dwKeyOffset	IN: Key offset in RAM
CBIOS_RSA_PRIVATE_KEY *pRSAKey	OUT: Pointer to private RSA key structure
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_SetKeyPublicRSA(DWORD dwKeyMemory, DWORD dwKeyOffset, CBIOS_RSA_PUBLIC_KEY *pRSAKey, BYTE bPass[16])

Delphi syntax: **function CBIOS_SetKeyPublicRSA(dwKeyMemory: Longint; dwKeyOffset: Longint; pRSAKey:PCBIOS_RSA_PUBLIC_KEY; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetKeyPublicRSA (ByVal dwKeyMemory As Long, ByVal dwKeyOffset As Long, ByRef pRSAKey As CBIOS_RSA_PUBLIC_KEY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Stores public RSA key into RAM for software RSA encryption.

Parameters:

DWORD dwKeyMemory	IN: Type of memory where RSA-Key is stored RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwKeyOffset	IN: Key offset in RAM
CBIOS_RSA_PUBLIC_KEY *pRSAKey	OUT: Pointer to public RSA key structure
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_SetKeyPrivateRSA(DWORD dwKeyMemory, DWORD dwKeyOffset, CBIOS_RSA_PRIVATE_KEY *pRSAKey, BYTE bPass[16])

Delphi syntax: **function CBIOS_SetKeyPrivateRSA(dwKeyMemory: Longint; dwKeyOffset: Longint; pRSAKey:PCBIOS_RSA_PRIVATE_KEY; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetKeyPrivateRSA(ByVal dwKeyMemory As Long, ByVal dwKeyOffset As Long, ByRef pRSAKey As CBIOS_RSA_PRIVATE_KEY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Stores private RSA key into RAM for software RSA encryption.

Parameters:

DWORD dwKeyMemory	IN: Type of memory where RSA-Key is stored

	RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwKeyOffset	IN: Key offset in RAM
CBIOS_RSA_PRIVATE_KEY *pRSAKey	OUT: Points to private RSA key structure
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_GenerateKeyPairRSA(DWORD dwPublicKeyMemory, DWORD dwPublicKeyOffset, DWORD dwPrivateKeyMemory, DWORD dwPrivateKeyOffset, WORD bits,BYTE *randomPool, BYTE bPass[16])

Delphi syntax: **function CBIOS_GenerateKeyPairRSA(dwPublicKeyMemory: Longint; dwPublicKeyOffset: Longint; dwPrivateKeyMemory: Longint; dwPrivateKeyOffset: Longint; bits: Word; randomPool: PByte; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_GenerateKeyPairRSA(ByVal dwPublicKeyMemory As Long, ByVal dwPublicKeyOffset As Long, ByVal dwPrivateKeyMemory As Long, ByVal dwPrivateKeyOffset As Long, ByVal bits As Integer, ByRef randomPool As CBIOS_BYTEARRAY, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Generates the private and the public RSA keys and stores them into corresponding RAM.

Parameters:

DWORD dwPublicKeyMemory	IN: Type of memory where public RSA Key is stored RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwPublicKeyOffset	IN: Public Key offset in RAM
DWORD dwPrivateKeyMemory	IN: Type of memory where private RSA Key is stored RAM1 = 1 / RAM2 = 2 / RAM3 = 3
DWORD dwPrivateKeyOffset	IN: Private Key offset in RAM
WORD bits	IN: Key length
BYTE * randomPool	IN: Pointer to random data, used for key generation
BYTE bPass[0x10]	IN: User or Admin Password

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls

	are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_OUT_OF_RANGE	Submitted address and/or size is/are out of the application's partition area.

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

void WINAPI CBIOS_PrepareRSAKey(WORD bits, BYTE* pbModulus, BYTE* pbExponent, BYTE* pbRSAkey])

This function converts a RSA key from modulus and exponent format into internal CBIOS RSA key representation to be used by CBIOS RSA encryption calls.

Parameters:

WORD bits	IN: Key length
BYTE * pbModulus	IN: Pointer to buffer, containing modulus part of RSA key
BYTE * pbExponent	IN: Pointer to buffer, containing exponent part of RSA key
BYTE * pbRSAKey	OUT: Pointer to buffer, where RSA key in internal format will be output

Return: None

1.7.7.2. Smarx®OS internal RSA keys

The following CBIOS API calls operate with Smarx OS internal RSA keys. They are meant to be used for establishing secure communication:

DWORD WINAPI CBIOS_EncryptInternalRSA(DWORD dwKeyID, PVOID pInBuffer, DWORD dwInBufferLen, PVOID pOutBuffer, DWORD *dwOutBufferLen, BYTE bPass[16])

Delphi syntax: **function CBIOS_EncryptInternalRSA(dwKeyID: Longint; pInBuffer: Pointer; dwInBufferLen: Longint; pOutBuffer: Pointer; pdwOutBufferLen: PLongint; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_EncryptInternalRSA(ByVal dwKeyID As Long, ByRef pInBuffer As CBIOS_BYTEARRAY, ByVal dwInBufferLen As Long, ByRef pOutBuffer As CBIOS_BYTEARRAY, ByVal pdwOutBufferLen As Long, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Encrypts data with Smarx OS internal RSA key (used for secure communication).

The CRYPTO-BOX XS and Versa models provide RSA support in software (on driver level). The CRYPTO-BOX SC provides hardware-based RSA (key never leaves the hardware). For this device type, hardware-based RSA encryption/decryption is used for all operations with Smarx OS internal RSA keys.

Parameters:

DWORD dwKeyID	IN: Shows which RSA key will be used - your Public or CRYPTO-BOX's Private: (CBIOS_INTERNAL_RSA_ID_PUBL1 or CBIOS_INTERNAL_RSA_ID_PRIV2)
PVOID pInBuffer	IN: Pointer to original data
DWORD dwInBufferLen	IN: Length of original data
PVOID pOutBuffer	OUT: Pointer to encrypted data
DWORD *dwOutBufferLen	OUT: Pointer to length of encrypted data
BYTE bPass[16]	IN: User Password, in case if CBIOS_INTERNAL_RSA_ID_PRIV2 is used

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_WRONG_PARAM	Invalid dwKeyID value

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

DWORD WINAPI CBIOS_DecryptInternalRSA(DWORD dwKeyID, PVOID pInBuffer, DWORD dwInBufferLen, PVOID pOutBuffer, DWORD *dwOutBufferLen, BYTE bPass[16])

Delphi syntax: **function CBIOS_DecryptInternalRSA(dwKeyID: Longint; pInBuffer: Pointer; dwInBufferLen: Longint; pOutBuffer: Pointer; pdwOutBufferLen: PLongint; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_DecryptInternalRSA(ByVal dwKeyID As Long, ByRef pInBuffer As CBIOS_BYTEARRAY, ByVal dwInBufferLen As Long, ByRef pOutBuffer As CBIOS_BYTEARRAY, ByVal pdwOutBufferLen As Long, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Decrypts data with Smarx OS internal RSA key (used for secure communication).

Parameters:

DWORD dwKeyID	Shows which RSA key will be used - your Public or CRYPTO-BOX's Private: (CBIOS_INTERNAL_RSA_ID_PUBL1 or CBIOS_INTERNAL_RSA_ID_PRIV2)
PVOID pInBuffer	IN: Pointer to encrypted data
DWORD dwInBufferLen	IN: Length of encrypted data
PVOID pOutBuffer	OUT: Pointer to decrypted data
DWORD *dwOutBufferLen	OUT: Pointer to length of decrypted data
BYTE bPass[16]	User Password, in case if CBIOS_INTERNAL_RSA_ID_PRIV2 is used

Return:

CBIOS_ERR_EXTENDED_MODE	CBIOS is in extended mode, regular function calls are not allowed in this mode.
CBIOS_ERR_LOCK_TIMEOUT	CRYPTO-BOX could not be reached for more than 20 seconds (see CBIOS_LockBox).
CBIOS_ERR_BOX_NOT_OPENED	CRYPTO-BOX was not opened with CBIOS_Open
CBIOS_ERR_WRONG_PARAM	Invalid dwKeyID value

Otherwise, standard error code will be returned (codes from 0x01 to 0x0A, 0x10XX, see cbios.h).

1.7.7.3. Extended RSA functionality (CRYPTO-BOX®SC only)

DWORD WINAPI CBIOS_GenerateKeyPairRSAEx(CBIOS_RSA_KEY* pRSAKeyPair, CBIOS_RSA_KEY* pRSAPublicKey, WORD bits, BYTE* randomPool);

pRSAKeyPair [in] Pointer to a structure receiving generated RSA keypair

pRSAPublicKey	[in] Pointer to a structure receiving public part of generated RSA keypair
bits	[in] RSA key size
randomPool	[in] Array of random bytes. Length of array must be at least CBIOS_RAND_POOL_SIZE(bits)
[return]	CBIOS Error code

Description: This function generates CBU SC RSA keypair of 512/1024/2048 bits size (**bits**) and saves the resulting keypair to the **pRSAKeyPair** structure.

In addition it saves the public part of the keypair separately to the **pRSAPublicKey** structure. The generation is software based and is supposed to be done on trusted computer. If **pRSAPublicKey** is null, then this parameter is ignored.

Further the resulting keypair can be saved to one of the cells in CBU SC dedicated memory (RAM4 zone) with the **CBIOS_CBU2_SetKeyRSA()** call. If necessary its access rights can be adjusted with the **CBIOS_CBU2_SetKeyInfoRSA()** function. Then it can be used for CBU SC RSA encryption: **CBIOS_EncryptRSAEx()**/**CBIOS_DecryptRSAEx()**.

Example: the *Test_RSA_HW ()* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_SetKeyRSA(DWORD dwKeyIndex,
CBIOS_RSA_KEY* pRSAKeyPair,
CBIOS_RSA_KEY_INFO* pRSAKeyInfo);

dwKeyIndex	[in] Index of RSA cell in CBU SC RAM4 zone for this keypair (starting from 0)
pRSAKeyPair	[in] Pointer to RSA keypair to write
pRSAKeyInfo	[in] Pointer to RSA keypair information (optional)
[return]	CBIOS Error code

Description: This function writes CBU SC RSA keypair (**pRSAKeyPair**) to one of CBU SC cells (**dwKeyIndex)** in dedicated memory (RAM4 zone).

The **pRSAKeyInfo** structure can optionally set access rights on this keypair, including:
- its usage for encryption,
- obtaining access rights info and/or changing the keypair value/access rights.

See section *1.7.7.4* for more details on how to define access rights and limitations in CBIOS_RSA_KEY_INFO structure.

Example: the *Test_RSA_HW ()* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_SetKeyInfoRSA(DWORD dwKeyIndex,
CBIOS_RSA_KEY_INFO* pRSAKeyInfo);

dwKeyIndex	[in] Index of RSA cell in CBU SC RAM4 holding the keypair (starting from 0)
pRSAKeyInfo	[in] Pointer to new RSA keypair information
[return]	CBIOS Error code.

Description: This function changes access rights and limitations for CBU SC RSA cell (**dwKeyIndex)** to values defined in the **pRSAKeyInfo** structure.

It can change access rights for this keypair on: its usage for encryption, obtaining access rights info and/or changing the keypair value/access rights. See section *1.7.7.4* for more details on how to define access rights and limitations in CBIOS_RSA_KEY_INFO structure.

Example: the *Test_RSA_HW ()* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_GetKeyInfoRSA(DWORD dwKeyIndex,
CBIOS_RSA_KEY_INFO* pRSAKeyInfo);

dwKeyIndex	[in] Index of RSA cell in CBU SC RAM4 storing the keypair (starting from 0)
pRSAKeyInfo	[in] Pointer to structure that will receive RSA keypair information
[return]	CBIOS Error code.

Description: This function retrieves access rights and limitations data for CBU SC RSA cell (**dwKeyIndex)** to the **pRSAKeyInfo** structure. See section *1.7.7.4* for more details on how to define/interpret access rights and limitations in CBIOS_RSA_KEY_INFO structure.

Example: the *Test_RSA_HW (void)* function of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

DWORD WINAPI CBIOS_CBU2_LockKeyRSA(DWORD dwKeyIndex);

dwKeyIndex	[in] Index of CBU SC RSA cell storing the keypair
[return]	CBIOS Error code.

Description: This function locks the CBU SC RSA cell (**dwKeyIndex),** so its keypair can not be used for further RSA encryption/decryption operations.


```
DWORD WINAPI CBIOS_CBU2_EncryptRSA( DWORD    dwKeyIndex,
                                    DWORD    dwMode,
                                    PVOID    pInBuffer,
                                    DWORD    dwInBufferLen,
                                    PVOID    pOutBuffer,
                                    DWORD*   pdwOutBufferLen  );
```

dwKeyIndex	[in] Index of RSA keypair storage cell in CBU SC RAM4 (starting from 0)
dwMode	[in] Defines key type and encryption padding mode, the value is bitwise combination of: CBIOS_RSA_PUBL_KEY – encrypt with public key CBIOS_RSA_PRIV_KEY – encrypt with private key and CBIOS_RSA_MARX_PADDING - use CBU RSA padding CBIOS_RSA_RSAREF_PADDING – use PKCS#1 compliant padding
pInBuffer	[in] Pointer to input buffer in computer memory (points to byte array to encrypt)
dwInBufferLen	[in] Size in bytes of pInBuffer
pOutBuffer	[in] Pointer to buffer that will receive encrypted data from pInBuffer.
pdwOutBufferLen	[in, out] Pointer to a variable containing the pOutBuffer size (in bytes). On return it contains number of encrypted bytes. Or required size of pOutBuffer
[return]	CBIOS Error code

Description: This function performs CBU SC hardware based RSA encryption using **dwKeyIndex** cell in RAM4 CBU SC memory as RSA key holder.

The **dwMode** parameter defines

1) which part of the keypair should be used: public or private, and
2) RSA padding mode: CBU RSA/PKCS#1.

The input data (byte array **pInBuffer** of length **dwInBufferLen**) will be encrypted and the result of this operation will be placed to the **pOutBuffer** of length defined by **pdwOutBufferLen**. On return the **pdwOutBufferLen** will contain the number of encrypted bytes or required size of pOutBuffer (if size of the pOutBuffer is not enough).

This function should be considered in combination with CBU SC hardware or software based RSA decryption. In other words some data being encrypted with public/private RSA key with this function can be decrypted with private/public RSA key of the same keypair using CBU SC hardware (**CBIOS_CBU2_DecryptRSA** – see below) or software (**CBIOS_DecryptRSAEx**) RSA decryption.

Example: The *Test_RSA_HW ()* and *Test_RSA_HW_SW ()* functions of *CBU2_Sample.c*

(MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

```
DWORD WINAPI CBIOS_CBU2_DecryptRSA( DWORD    dwKeyIndex,
                                    BYTE     bKeyType,
                                    PVOID    pInBuffer,
                                    DWORD    dwInBufferLen,
                                    PVOID    pOutBuffer,
                                    DWORD*   pdwOutBufferLen  );
```

dwKeyIndex	[in] Index of RSA keypair storage cell in CBU SC RAM4 (starting from 0)
dwMode	[in] Defines key type and encryption padding mode, the value is bitwise combination of: CBIOS_RSA_PUBL_KEY – decrypt with public key CBIOS_RSA_PRIV_KEY – decrypt with private key and CBIOS_RSA_MARX_PADDING - use CBU RSA padding CBIOS_RSA_RSAREF_PADDING – use PKCS#1 compliant padding
pInBuffer	[in] Pointer to input buffer in computer memory (points to byte array to decrypt)
dwInBufferLen	[in] Size in bytes of pInBuffer
pOutBuffer	[in] Pointer to buffer that will receive decrypted data from pInBuffer
pdwOutBufferLen	[in, out] Pointer to a variable containing the pOutBuffer size (in bytes). On return it contains number of decrypted bytes, or required size of pOutBuffer
[return]	CBIOS Error code

Description: This function performs CBU SC hardware based RSA decryption using **dwKeyIndex** cell in RAM4 CBU SC memory as RSA key holder.
The **dwMode** parameter defines

1) which part of the keypair should be used: public or private;
2) RSA padding mode: CBU RSA/PKCS#1.

The input data (byte array **pInBuffer** of length **dwInBufferLen**) will be decrypted and the result of this operation will be placed to the **pOutBuffer** of length defined by **pdwOutBufferLen**. On return the **pdwOutBufferLen** will contain the number of decrypted bytes or required size of pOutBuffer (if size of pOutBuffer is not enough).
This function should be considered in combination with CBU SC hardware or software based RSA encryption. In other words some data being decrypted with public/private RSA key with this function can be encrypted with private/public RSA key of the same keypair using CBU SC hardware (**CBIOS_CBU2_EncryptRSA** – see above) or software (**CBIOS_EncryptRSAEx** – see) RSA encryption.

Example: The *Test_RSA_HW ()* and *Test_RSA_HW_SW ()* functions of *CBU2_Sample.c*

(MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

```
DWORD WINAPI CBIOS_EncryptRSAEx( CBIOS_RSA_KEY*    pRSAKey,
                                 DWORD             dwMode,
                                 PVOID             pInBuffer,
                                 DWORD             dwInBufferLen,
                                 PVOID             pOutBuffer,
                                 DWORD*            pdwOutBufferLen );
```

pRSAKeyPair	[in] Points to RSA keypair (for private encryption) or public key (for public encryption).
dwMode	[in] Defines key type and encryption padding mode, the value is bitwise combination of (for compatibility with CBU SC hardware RSA encryption the key type may be specified explicitly): CBIOS_RSA_PUBL_KEY – encrypt with public key CBIOS_RSA_PRIV_KEY – encrypt with private key and CBIOS_RSA_MARX_PADDING - use CBU RSA padding CBIOS_RSA_RSAREF_PADDING – use PKCS#1 compliant padding
pInBuffer	[in] Pointer to input buffer in computer memory (points to byte array to encrypt)
dwInBufferLen	[in] Size in bytes of pInBuffer.
pOutBuffer	[in] Pointer to buffer that will receive encrypted data from pInBuffer
pdwOutBufferLen	[in, out] Pointer to a variable containing the pOutBuffer size (in bytes). On return it contains number of encrypted bytes or required size of pOutBuffer
[return]	CBIOS Error code.

Description: This function performs software based RSA encryption using **pRSAKeyPair** as a pointer to the structure containing RSA private or public key. The software based RSA encryption is 100% compatible with CBU SC hardware based RSA.

The **dwMode** parameter optionally (for compatibility with CBU SC hardware encryption) defines which part of the keypair should be used: public or private. It can also define RSA padding mode: CBU RSA/PKCS#1.

The input data (byte array **pInBuffer** of length **dwInBufferLen**) will be encrypted and the result of this operation will be placed to the **pOutBuffer** of length defined by **pdwOutBufferLen**. On return the **pdwOutBufferLen** will contain the number of encrypted bytes or required size of pOutBuffer (if size of pOutBuffer is not enough).

This function should be considered in combination with CBU SC hardware or software based RSA decryption. In other words some data being encrypted with public/private RSA key with this function can be decrypted with private/public RSA key of the same keypair using CBU SC

hardware (**CBIOS_CBU2_DecryptRSA**) or software (**CBIOS_DecryptRSAEx** – see below) RSA decryption.

Example: The *Test_RSA_HW ()* and *Test_RSA_HW_SW ()* functions of *CBU2_Sample.c* (MSVS 2005 or 2008 static sample included to the Smarx PPK) illustrates this call usage.

```
DWORD WINAPI CBIOS_DecryptRSAEx( CBIOS_RSA_KEY*    pRSAKey,
                                 DWORD             dwMode,
                                 PVOID             pInBuffer,
                                 DWORD             dwInBufferLen,
                                 PVOID             pOutBuffer,
                                 DWORD*            pdwOutBufferLen );
```

pRSAKeyPair	[in] Points to RSA keypair (for private decryption) or public key (for public decryption).
dwMode	[in] Defines key type and decryption padding mode, the value is bitwise combination of (for compatibility with CBU SC hardware RSA encryption the key type may be specified explicitly): CBIOS_RSA_PUBL_KEY – decrypt with public key CBIOS_RSA_PRIV_KEY – decrypt with private key and CBIOS_RSA_MARX_PADDING - use CBU RSA padding CBIOS_RSA_RSAREF_PADDING – use PKCS#1 compliant padding
pInBuffer	[in] Pointer to input buffer in computer memory (points to byte array to decrypt)
dwInBufferLen	[in] Size in bytes of pInBuffer
pOutBuffer	[in] Pointer to buffer that will receive decrypted data from pInBuffer
pdwOutBufferLen	[in, out] Pointer to a variable containing the pOutBuffer size (in bytes). On return it contains number of decrypted bytes or required size of pOutBuffer.
[return]	CBIOS Error code

Description: This function performs software based RSA decryption using **pRSAKeyPair** as a pointer to the structure containing RSA private or public key. The software based RSA encryption is 100% compatible with CBU SC hardware based RSA.

The **dwMode** parameter optionally (for compatibility with CBU SC hardware decryption) defines which part of the keypair should be used: public or private. It can also define RSA padding mode: CBU RSA/PKCS#1.

The input data (byte array **pInBuffer** of length **dwInBufferLen**) will bedecrypted and the result of this operation will be placed to the **pOutBuffer** of length defined by **pdwOutBufferLen**. On return the **pdwOutBufferLen** will contain the number of decrypted bytes or required size of

pOutBuffer (if size of pOutBuffer is not enough).

This function should be considered in combination with CBU SC hardware or software based RSA encryption. In other words some data being decrypted with public/private RSA key with this function can be encrypted with private/public RSA key of the same keypair using CBU SC hardware (**CBIOS_CBU2_EncryptRSA**) or software (**CBIOS_EncryptRSAEx** – see above) RSA decryption.

1.7.7.4. New CBIOS structures related to extended RSA

CBIOS_RSA_KEY

This structure contains RSA keypair or RSA public part.

```
typedef struct {
   DWORD dwStructSize;
   WORD  wModulusLen;      // in bytes, must be a multiple of four bytes
   WORD  wPrivExpLen;      // in bytes, must be a multiple of four bytes
   BYTE  ubModulus[CBIOS_CBU2_MAX_RSA_KEY_LEN]; // most significant byte first
   BYTE  ubPrivExp[CBIOS_CBU2_MAX_RSA_KEY_LEN]; // most significant byte first
} CBIOS_RSA_KEY;
```

dwStructSize	[in] Size of structure in bytes.
wModulusLen	[in,out] RSA keypair modulus length in bytes.
wPrivExpLen	[in,out] RSA keypair exponent length in bytes.
ubModulus	[in,out] Modulus bytes. Most significant byte first.
ubPrivExp	[in,out] Exponent bytes. Most significant byte first.

The **CBIOS_RSA_KEY** structure is used in the following CBIOS calls:

- **CBIOS_GenerateKeyPairRSAEx()**
- **CBIOS_CBU2_SetKeyRSA()**
- **CBIOS_EncryptRSAEx()**
- **CBIOS_DecryptRSAEx()**

CBIOS_RSA_KEY_INFO

This structure defines access rights and limitations for CBU SC RSA keypair: its usage for encryption, obtaining access rights info and/or changing the keypair value/access rights.

```
typedef struct {
        DWORD dwStructSize;
        DWORD dwAccess;   // low nibble:   SetKey/SetKeyInfo access rights,
                          // high nibble:  Encrypt/Decrypt and GetKeyInfo
        WORD  wBits;
} CBIOS_RSA_KEY_INFO;
```


dwStructSize	[in] Size of structure in bytes.
dwAccess	[in,out] RSA keypair access flags. Lower 4 bits define access rights for changing RSA keypair (SetKey and SetKeyInfo operations). Higher 4 bits define access rights for performing RSA encryption, decryption and obtaining keypair information. Supported values: CBIOS_CBU2_ACCESS_NEVER CBIOS_CBU2_ACCESS_ALWAYS CBIOS_CBU2_ACCESS_UPW CBIOS_CBU2_ACCESS_APW CBIOS_CBU2_ACCESS_LOCK
wBits	[out] RSA key size in bits (key strength).

Comment: The **dwAccess** member of this structure defines required access rights or returns current access rights for RSA keypair. For any CBU SC RSA keypair access rights can be set for:

- encryption/decryption and obtaining information on this keypair (key strength and current access rights)
- changing the keypair value

Supported values and their meaning:

- **CBIOS_CBU2_ACCESS_NEVER** – this value (if being set) can not be reset even by MARX distributor. MARX distributor can only reprogram the whole RAM4 zone of this CBU SC unit, so all current values of RSA keypairs (if any) will be lost
- **CBIOS_CBU2_ACCESS_ALWAYS** – this value means free access (no limitations).
- **CBIOS_CBU2_ACCESS_UPW** – UPW login is required
- **CBIOS_CBU2_ACCESS_APW** – APW login is required
- **CBIOS_CBU2_ACCESS_LOCK** – the access will be locked, only MARX distributor can unlock it.

The **CBIOS_RSA_KEY_INFO** structure is used in the following CBIOS calls:

- **CBIOS_CBU2_SetKeyRSA()**
- **CBIOS_CBU2_SetKeyInfoRSA()**
- **CBIOS_CBU2_GetKeyInfoRSA()**

1.7.8. Hash Functions

DWORD WINAPI CBIOS_MD5Hash(PVOID pInBuffer, DWORD dwInBufferLen, PVOID pOutBuffer, DWORD *dwOutBufferLen)

Delphi syntax: **function CBIOS_ MD5Hash (pInBuffer: Pointer; dwInBufferLen: Longint; pOutBuffer: Pointer; pdwOutBufferLen: PLongint): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_MD5Hash(ByRef pInBuffer As CBIOS_BYTEARRAY, ByVal dwInBufferLen As Long, ByRef pOutBuffer As CBIOS_BYTEARRAY, ByVal pdwOutBufferLen As Long) As Long**

Calculates MD5 hash from any byte sequence that is submitted via input buffer. For MD5 hash calculation CRYPTO-BOX presence is not required.

Parameters:

PVOID pInBuffer	IN: Points on input bytes sequence
DWORD dwInBufferLen	IN: Length of input bytes sequence
PVOID pOutBuffer	OUT: Points on calculated bytes sequence
DWORD *dwOutBufferLen	OUT: Points on length of calculated bytes sequence

Return:

0	if success
errorcode	see cbios.h for details

2. Smarx®OS Networking: CBIOS on the Network

2.1. General issues

Smarx®OS Networking allows applications to access CRYPTO-BOX hardware attached to one computer within a network. It also allows to limit/control the number of applications of one type running on the network (seats). A special program, called Smarx OS Network Server, monitors remote connections to the CRYPTO-BOX.

This document contains the network API reference only. Please read our White Paper Network Licensing with the CRYPTO-BOX for an introduction to network usage. This document also describes the administration of the network server.

2.2. Network CBIOS API Calls: detailed description

INT32 WINAPI CBIOS_ScanNetwork(DWORD dwScanTimeMs);

Delphi syntax: ***function CBIOS_ScanNetwork(dwScanTimeMs: Longint): Integer; stdcall;***

Visual Basic syntax: ***not implemented***

Search for Smarx OS Network Servers in the local network

Parameters:
DWORD dwScanTimeMs IN: Scan time in milliseconds

Return:
Number of Smarx OS Network Servers found in the network.

CBIOS looks for the defined scan time whether it finds any servers in the local network and stops searching after this time, regardless if server(s) were found or not. Searching for servers is done via UDP broadcasting, by default port 8766 is used (see also **CBIOS_SetScanPort**. See our White Paper Network Licensing with the CRYPTO-BOX, chapter 4 for more details.

Example:

```
int n = CBIOS_ScanNetwork(2000);
// n - number of found CBIOS Network servers
```

DWORD WINAPI CBIOS_SetScanPort(WORD wUDPPort);

Delphi syntax: ***function CBIOS_SetScanPort(wUDPPort: Word): Longint; stdcall;***

Visual Basic syntax: ***not implemented***

Set UDP port used for network searching (default port 8766 is used if not set)

Parameters:

WORD wUDPPort	IN: UDP Port

Return:
Always 0 – Success

DWORD WINAPI CBIOS_GetScanPort(WORD* pwUDPPort);

Delphi syntax: **function CBIOS_GetScanPort(pwUDPPort : PWord): Longint; stdcall;**

Visual Basic syntax: **not implemented**

Get UDP port used for network searching (default port 8766 is used if not set)

Parameters:

WORD* pwUDPPort	OUT: UDP Port

Return:

0	Success
CBIOS_ERR_WRONG_PARAM	**pwUDPPort** is NULL)

DWORD WINAPI CBIOS_GetServerInfo(INT32 iServerIndex, CBIOS_SERVER_INFO* pServerInfo);

Delphi syntax: **function CBIOS_GetServerInfo(iServerIndex: Integer;**
pServerInfo: PCBIOS_SERVER_INFO): Longint; stdcall;

Visual Basic syntax: **not implemented**

Returns Smarx OS Network Server info based on its Index.

Parameters:

INT32 iServerIndex	IN: Server# from 1 to <Servers Quantity> value, Returned by the preceding **CBIOS_ScanNetwork**() call
CBIOS_SERVER_INFO* pServerInfo	IN/OUT: Pointer to the structure which contains Server information.

Return:

0	Success
CBIOS_ERR_WRONG_PARAM	Invalid **iServerIndex or pServerInfo is NULL**

CBIOS_SERVER_INFO structure contains the following information:

CHAR	**szServerIp[16]**	Server IP address
WORD	**wServerPort**	Server port (for connection)
INT32	**iBoxesNumber**	Number of CRYPTO-BOX units attached to the server

DWORD WINAPI CBIOS_Connect(CHAR *ServerName, USHORT Port);

Delphi syntax: **function CBIOS_Connect(ServerName: PChar; Port: Word): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_Connect (ByRef ServerName As CBIOS_BYTEARRAY, ByVal Port As Integer) As Long**

Connects to Smarx OS Network Server.

Parameters:

CHAR * ServerName	IN: Server Network Name or IP address
USHORT Port	IN: Server Port

Return:

0	Success
CBIOS_ERR_WRONG_PARAM	Wrong server name submitted
CBIOS_ERR_CONN_REFUSED	Remote procedure call on server side produced an error.
Standard CBIOS error	See cbios.h for details

DWORD WINAPI CBIOS_Disconnect();

Delphi syntax: **function CBIOS_Disconnect : Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_Disconnect () As Long**

Disconnects from Smarx OS Network Server.

Parameters: None

Return:

0	Success
CBIOS_ERR_CONN_REFUSED	Remote procedure call on server side produced an error.
Standard CBIOS error	See cbios.h for details

DWORD WINAPI CBIOS_LockLicence();

Delphi syntax: **function CBIOS_LockLicence : Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_LockLicence () As Long**

Locks a network license for the opened application/partition.

Parameters: None

Return:ReleaseLicense

0	Success
CBIOS_ERR_NO_FREE_License	There's no free license for application
CBIOS_ERR_CONN_REFUSED	Error during remote procedure call on server side.
Standard CBIOS error	See cbios.h for details

DWORD WINAPI CBIOS_ReleaseLicence();

Delphi syntax: **function CBIOS_ReleaseLicence : Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_ReleaseLicence () As Long**

Releases the network license of the opened application/partition.

Parameters: None

Return:

0	Success
CBIOS_ERR_CONN_REFUSED	Error during remote procedure call on server side.
Standard CBIOS error	See cbios.h for details

DWORD WINAPI CBIOS_GetAppLicenses(WORD wAppID, DWORD* pdwRuleId, DWORD* pdwNetLic);

Delphi syntax: **function CBIOS_GetAppLicences(wAppID: Word; pdwRuleId: PLongint; pdwNetLic: PLongint): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_GetAppLicences(ByVal wAppID As Integer, ByRef pdwRuleId As Long, ByRef pdwNetLic As Long) As Long**

Reads information about the application's local and network licenses from the LMT.

Parameters:

WORD wAppId	IN: Application/partition number
DWORD * pdwRuleId	OUT: Pointer to license rule identifier
DWORD * pdwNetLic	OUT: Pointer to network license value

Return:

0	Success
CBIOS_ERR_NOT_LICENSED	Application/partition is not licensed
CBIOS_ERR_CONN_REFUSED	Remote procedure call on server side produced an error.
Standard CBIOS error	See cbios.h for details

DWORD WINAPI BIOS_SetAppLicenses(WORD wAppID, DWORD dwReserved, DWORD dwNetLic, BYTE bPass[0x10]);

Delphi syntax: **function CBIOS_SetAppLicences(wAppID: Word; dwReserved: Longint; dwNetLic: Longint; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_SetAppLicences(ByVal wAppID As Long, ByVal dwReserved As Long, ByVal dwNetLic As Long, ByRef bPass As CBIOS_BYTEARRAY16) As Long**

Writes information about the application's local and network licenses to the LMT.

Parameters:

WORD wAppId	IN: Application/partition number
DWORD dwReserved	IN: Reserved (would be 0)
DWORD dwNetLic	IN: Network license value
BYTE bPass[0x10]	IN: Admin Password (APW)

Return:

0	Success
CBIOS_ERR_NOT_LICENSED	Application/partition is not licensed
CBIOS_ERR_CONN_REFUSED	Remote procedure call on server side produced an error.
Standard CBIOS error	see cbios.h for details

DWORD WINAPI CBIOS_SetRuleAppLicences(WORD wAppID, BYTE bRuleId, BYTE bNetLic, BYTE bPass[0x10]);

Delphi syntax: **function CBIOS_SetRuleAppLicences(wAppID: Word; bRuleId: Byte; bNetLic: Byte; bPass: PTPasswd): Longint; stdcall;**

Visual Basic syntax: ***not implemented***

Writes information about the application network licenses and license distribution rule to the LMT.

Licenses rule ID :

CBIOS_NET_LICENCE_RULE_ DEFAULT (0)	Licenses are locked without taking client's IP/Terminal Session into account (default scenario), CBIOS_LockLicenceExt will deduct one license for each application instance, regardless of whether the protected application is started several times on one computer/IP address/Terminal Session or on different computers/IP addresses/Terminal Sessions).
CBIOS_NET_LICENCE_RULE_ IP_1L (1)	"One license per one IP", CBIOS_LockLicenceExt will deduct only one license per IP address (unlimited number of instances can be launched concurrently from one IP). Important Note: This option does not consider Terminal Server Session!
CBIOS_NET_LICENCE_RULE_ MACADD_TS_1L (2)	"One seat per User and MAC address" for dynamic IP, CBIOS_LockLicenceExt will deduct only one license for: – one MAC address (unlimited number of instances can be launched concurrently from one MAC) – one Terminal Session (unlimited number of instances can be launched concurrently in the same TS)
CBIOS_NET_LICENCE_RULE_ IPADD_TS_1L (3)	"One seat per IP, User and MAC address" for static IP, CBIOS_LockLicenceExt will deduct only one license for each Terminal Server Session (no matter how often the application is started in this Terminal Server Session with a particular IP address). The same as CBIOS_NET_LICENCE_RULE_MACADD_TS_1L + IP

Parameters:

WORD wAppId	IN: Application/partition number
BYTE bRuleId	IN: License rule identifier
BYTE bNetLic	IN: Network license value
BYTE bPass[0x10]	IN: Admin Password (APW)

Return:

0	Success
CBIOS_ERR_NOT_LICENSED	Application/partition is not licensed
CBIOS_ERR_OUT_OF_RANGE	Invalid rule id

CBIOS_ERR_CONN_REFUSED	Remote procedure call on server side produced an error.
Standard CBIOS error	see cbios.h for details

DWORD WINAPI CBIOS_CheckAppLicence(WORD wAppID, DWORD * pdwRuleId, DWORD * pdwNetLic);

Delphi syntax: **function CBIOS_CheckAppLicence(wAppID: Word; pdwRuleId: PLongint; pdwNetLic: PLongint): Longint; stdcall;**

Visual Basic syntax: **Function CBIOS_CheckAppLicence(ByVal wAppID As Integer, ByRef pdwRuleId As Long, ByRef pdwNetLic As Long) As Long**

Retrieves information about free (not busy) local and network licenses of an application.

Parameters:

WORD wAppId	IN: Application/partition number
DWORD * pdwRuleId	OUT: Pointer to license rule identifier
DWORD * pdwNetLic	OUT: Pointer to network license value

Return:

0	Success
CBIOS_ERR_NOT_LICENSED	Application/partition is not licensed
CBIOS_ERR_CONN_REFUSED	Remote procedure call on server side produced an error.
Standard CBIOS error	see cbios.h for details

3. Contact and Support

USA
MARX CryptoTech LP
489 South Hill Street
Buford, GA 30518
USA
www.marx.com

Sales: sales@marx.com
Support: support@marx.com
Phone: (+1) 770-904-0369

Germany
MARX Software Security GmbH
Vohburger Str. 68
D-85104 Wackerstein
Germany
www.marx.com

Sales: sales-de@marx.com
Support: support-de@marx.com
Phone: +49 (0) 8403 9295-0

4. Alphabetical Index

