CBIOS DO API Reference

Document: 0-23May010ks(CBIOS DO API Reference).odt
Last update: 1 September 2017 by Steffen Kaetsch
Environment: C++ (Microsoft Visual Studio), Delphi

Executive summary

Smarx®OS DataObjects API provides convenient access to various data objects, like expiration date, usage counter, password, self-defined objects, etc., stored in CRYPTO-BOX® memory partitions. MARX® provides tools for configuring the CRYPTO-BOX with pre-defined DataObject settings with can be queried via API later. See Smarx Compendium, chapter 4 for more information on CRYPTO-BOX configuration.

Table of Contents

1. Smarx®OS Data Objects API

1.1. Overview

Smarx OS Data Objects API (DO API) is based on top of the CBIOS API (see Smarx Compendium, chapter 12 and separate CBIOS API Reference).
Smarx OS DataObjects API provides convenient access to various objects, like expiration date, usage counter, password, self-defined objects, etc., stored in CRYPTO-BOX® memory partitions. The Smarx OS DO API is one of the basements for the automatic protection (AutoCrypt). MARX provides tools for configuring the CRYPTO-BOX with pre-defined DataObject settings with can be queried via API later. See Smarx Compendium, chapter 4.5 for more information on CRYPTO-BOX configuration.

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 under Support → Documents → White Papers.

This document contains the CBIOS DO 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 12 and 14 in the Smarx Compendium first before working with this document – it will help you to understand the basic CBIOS API, the CBIOS call sequence and gives an introduction to the available CBIOS DataObject types.

1.2. Smarx®OS Data Objects API Calls - detailed description

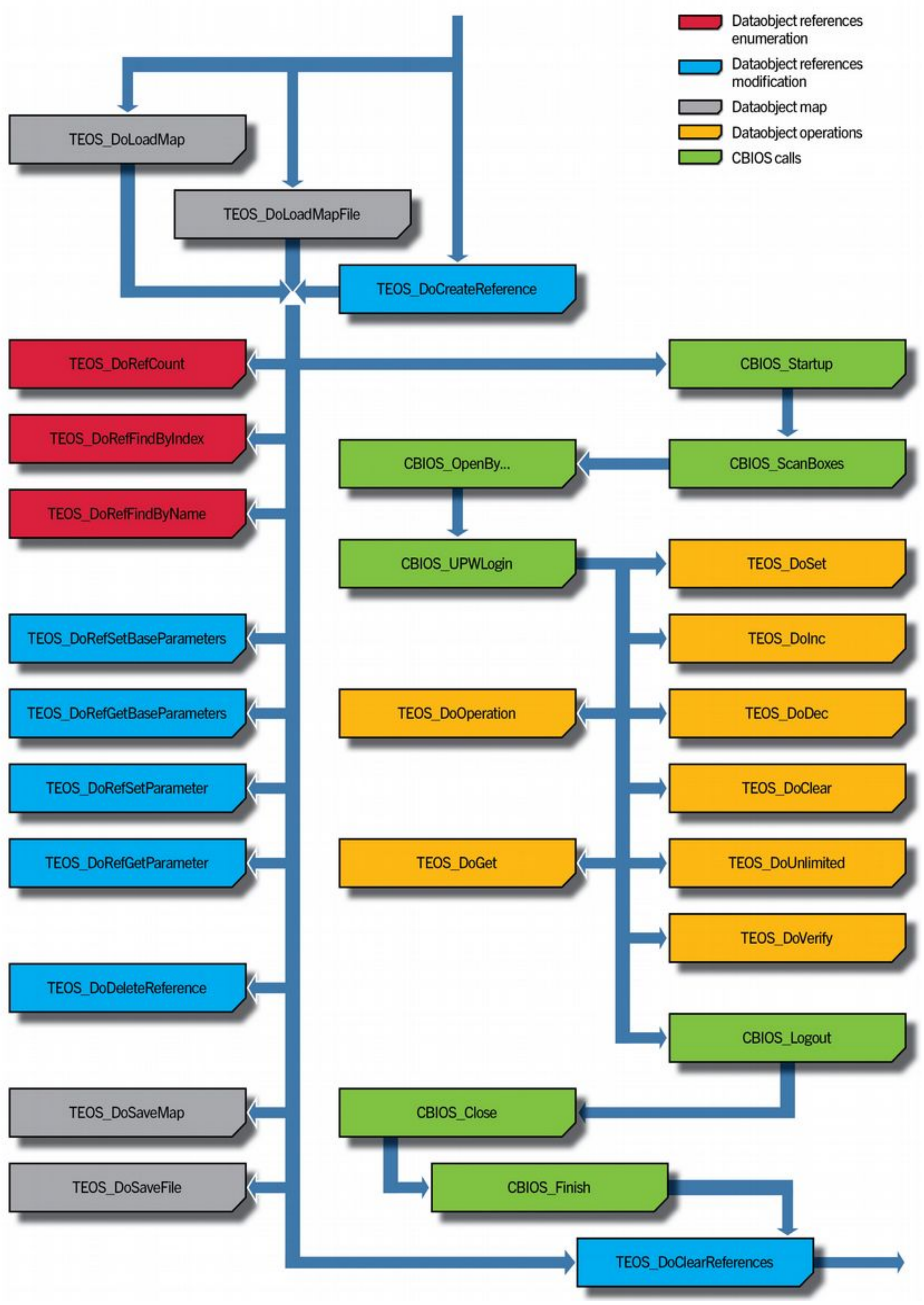

Fig. 1.1:
Smarx®OS Data Object API Calls – overview

DWORD WINAPI TEOS_DoCreateReference(DWORD dwDoID, DWORD dwDoType, DWORD dwRAMNumber, DWORD dwOffset, DWORD dwReserved);

Delphi syntax: **function TEOS_DoCreateReference(dwDoID: DWORD; dwDoType: DWORD; dwRAMNumber: DWORD; dwOffset: DWORD; dwReserved: DWORD): DWORD; stdcall;**

Creates the Data Object reference in memory.

Parameters:

DWORD dwDoID	IN: Data Object's ID, must be > 1
DWORD dwDoType	IN: Data Object's type
DWORD dwRAMNumber	IN: RAM1/RAM2/RAM3
DWORD dwOffset	IN: memory offset in partition
DWORD dwReserved	reserved

Return:

0	Successful
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoDeleteReference(DWORD dwDoID);

Delphi syntax: **function TEOS_DoDeleteReference(dwDoID: DWORD): DWORD; stdcall;**

Deletes the Data Object reference in memory.

Parameters:

DWORD dwDoID	IN: Data Object's ID, must be > 1

Return:

0	Successful
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoClearReferences();

Delphi syntax: **function TEOS_DoLoadMap(pData: PBYTE; dwSize: DWORD): DWORD; stdcall;**

Deletes all references in memory.

Parameters: None

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoLoadMap(BYTE* pData, DWORD dwSize);

<u>*Delphi syntax:*</u> **function TEOS_DoLoadMap(pData: PBYTE; dwSize: DWORD): DWORD; stdcall;**

Loads Data Object's Map from buffer.

Parameters:

BYTE * pData	IN: Pointer to Data Object buffer
DWORD dwSize	IN: Buffer size

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoLoadMapFile(const char* szFileName);

<u>*Delphi syntax:*</u> **function TEOS_DoLoadMapFile(szFileName: pchar): DWORD; stdcall;**

Loads Data Object's Map from file.

Parameters:

const CHAR * szFileName	IN: Data Object's Map file name

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoSaveMap(BYTE* pData, DWORD * pdwSize);

<u>*Delphi syntax:*</u> **function TEOS_DoSaveMap(pData: PBYTE; pdwSize: PDWORD): DWORD; stdcall;**

Saves Data Object's Map into buffer.

Parameters:

BYTE * pData	OUT: Pointer to Data Object buffer: if NULL: required buffer size value is returned in * pdwSize
DWORD * pdwSize	IN/OUT: Pointer to buffer size value

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoSaveMapFile(const char* szFileName);

<u>*Delphi syntax:*</u> **function TEOS_DoSaveMapFile(szFileName: pchar): DWORD; stdcall;**

Saves Data Object's Map into file.

Parameters:

const CHAR * szFileName	IN: Data Object's Map file name

Return:

0	Success
error code	See error code description in chapter *1.3*

INT WINAPI TEOS_DoRefCount();

<u>*Delphi syntax:*</u> **function TEOS_DoRefCount(): integer; stdcall;**

Returns number of references in the memory.
Parameters: None

Return:

0	No references available
N	Number of references in the memory

DWORD WINAPI TEOS_DoRefFindByIndex(INT iRefIndex, DWORD *pdwDoID);

<u>*Delphi syntax:*</u> **function TEOS_DoRefFindByIndex(iRefIndex: integer {1...n}; pdwDoID: PDWORD): DWORD; stdcall;**

Gets ID of Data Object in the memory, referenced by index.

Parameters:

INT iRefIndex	IN: Data Object's index from 1 to <RefCount> value, returned by the preceding TEOS_DoRefCount() call.
DWORD * pdwDoID	OUT: Pointer on ID value

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoRefFindByName(char *szName, DWORD *pdwDoID);

<u>*Delphi syntax:*</u> **function TEOS_DoRefFindByName(szName: pchar;
pdwDoID: PDWORD): DWORD; stdcall;**

Gets ID of Data Object in the memory, referenced by name.

Parameters:

char * szName	IN: Data Object's name - set by preceding TEOS_DoRefSetParameter() call.
DWORD * pdwDoID	OUT: Pointer to ID value

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoRefSetBaseParameters(DWORD dwDoID, DWORD dwRAMNumber, DWORD dwOffset, DWORD dwReserved);

<u>*Delphi syntax:*</u> **function TEOS_DoRefSetBaseParameters(dwDoID: DWORD; dwRAMNumber: DWORD; dwOffset: DWORD; dwReserved: DWORD): DWORD; stdcall;**

Sets basic parameters of a Data Object in the CRYPTO-BOX memory.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
DWORD dwRAMNumber	IN: RAM1/RAM2/RAM3
DWORD dwOffset	IN: memory offset in partition
DWORD dwReserved	reserved

Return:

0	Success

error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoRefGetBaseParameters(DWORD dwDoID, DWORD *pdwDoType, DWORD *pdwRAMNumber, DWORD *pdwOffset, DWORD *pdwReserved);

<u>*Delphi syntax:*</u> **function TEOS_DoRefGetBaseParameters(dwDoID: DWORD; pdwDoType: PDWORD; pdwRAMNumber: PDWORD; pdwOffset: PDWORD; pdwReserved: PDWORD): DWORD; stdcall;**

Retrieves basic parameters of a Data Object in the CRYPTO-BOX memory.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
DWORD * pdwDoType	OUT: Pointer to Data Object Type value
DWORD * pdwRAMNumber	OUT: Pointer to Data Object memory value (RAM1/RAM2/RAM3)
DWORD * pdwOffset	OUT: Pointer to Data Object memory offset in partition
DWORD * pdwReserved	reserved

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoRefSetParameter(DWORD dwDoID, DWORD dwParameterID, BYTE *pData, DWORD dwSize);

<u>*Delphi syntax:*</u> **function TEOS_DoRefSetParameter(dwDoID: DWORD; dwParameterID: DWORD; pData: PBYTE; dwSize: DWORD): DWORD; stdcall;**

Sets/changes parameter of a Data Object in memory.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
DWORD dwParameterID	IN: Parameter type **TEOSDO_ID** **TEOSDO_TYPE** **TEOSDO_RAM** **TEOSDO_OFFSET** **TEOSDO_SIZE** **TEOSDO_NAME**

BYTE * pData	IN: Pointer to parameter data
DWORD dwSize	IN: Size of parameter data

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoRefGetParameter(DWORD dwDoID, DWORD dwParameterID, BYTE *pData, DWORD *pdwSize);

<u>*Delphi syntax:*</u> **function TEOS_DoRefGetParameter(dwDoID: DWORD; dwParameterID: DWORD; pData: PBYTE; pdwSize: PDWORD): DWORD; stdcall;**

Retrieves a parameter of a Data Object in memory.

Parameters:

DWORD dwDoID	IN: Data Object's ID, must be > 1
DWORD dwParameterID	IN: Parameter type **TEOSDO_ID** **TEOSDO_TYPE** **TEOSDO_RAM** **TEOSDO_OFFSET** **TEOSDO_SIZE** **TEOSDO_NAME**
BYTE * pData	OUT: Pointer to parameter data
DWORD * pdwSize	OUT: Pointer of parameter data size

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoSet(DWORD dwDoID, BYTE* pData, DWORD dwDataSize, DWORD dwParameter, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoSet(dwDoID: DWORD; pData: PBYTE; dwDataSize: DWORD; dwParameter: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Sets/programs a Data Object value into a CRYPTO-BOX partition.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1

BYTE * pData	IN: Pointer to Data Object value
DWORD dwDataSize	IN: DatObject value size
DWORD dwParameter	IN: additional parameter (reserved); 0 - not used
PASSW bPass	IN: Admin Password (APW) required for **TEOSDO_NET_License**, for the rest: NULL

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoGet(DWORD dwDoID, BYTE* pData, DWORD * pdwDataSize, DWORD dwParameter, PASSW bPass);

Delphi syntax: **function TEOS_DoGet(dwDoID: DWORD; pData: PBYTE; pdwDataSize: PDWORD; dwParameter: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Retrieves Data Object value from a CRYPTO-BOX partition. This function is valid for all Data Object types except **TEOSDO_PSW_HASH**, **TEOSDO_APP_CS**, **TEOSDO_APP_HASH**. For security reasons, hash and checksum values can only be set (calculated on the fly) and verified (calculated on the fly and compared with the values stored in the CRYPTO-BOX). There is no way to read these values from the CRYPTO-BOX directly.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
BYTE * pData	OUT: Pointer to Data Object value
DWORD dwDataSize	OUT: Pointer to DatObject value size
DWORD dwParameter	IN: additional parameter: 0 – not used **TEOSDO_DATE_AS_STRING** - gets expiration value as string, e.g. “19 DEC 2011” **TEOSDO_DATE_AS_DWORD** - gets expiration value as DWORD (days left), for instance 352 (days left)
PASSW bPass	IN: NULL – reserved

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoInc(DWORD dwDoID, BYTE* pData, DWORD dwDataSize, DWORD dwParameter, PASSW bPass);

Delphi syntax: **function TEOS_DoInc(dwDoID: DWORD; pData: PBYTE; dwDataSize: DWORD; dwParameter: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Increments a Data Object value in the CRYPTO-BOX partition. This function is valid for Data Object types containing an expiration counter such as **TEOSDO_EXPIRATION_DATE, TEOSDO_NUMBER_OF_DAYS, TEOSDO_TIME_ALLOWED, TEOSDO_COUNTER**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
BYTE * pData	IN: Pointer to increment value
DWORD dwDataSize	IN: Size of increment value
DWORD dwParameter	IN: additional parameter: 0 - not used
PASSW bPass	IN: NULL – reserved

Return:

0	Successful
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoDec(DWORD dwDoID, BYTE* pData, DWORD dwDataSize, DWORD dwParameter, PASSW bPass);

Delphi syntax: **function TEOS_DoDec(dwDoID: DWORD; pData: PBYTE; dwDataSize: DWORD; dwParameter: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Decrements a Data Object value in the CRYPTO-BOX partition. This function is valid for Data Object types containing an expiration counter such as **TEOSDO_EXPIRATION_DATE, TEOSDO_NUMBER_OF_DAYS, TEOSDO_TIME_ALLOWED, TEOSDO_COUNTER**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
BYTE * pData	IN: Pointer to decrement value
DWORD dwDataSize	IN: Size of decrement value
DWORD dwParameter	IN: additional parameter: 0 - not used
PASSW bPass	IN: NULL - reserved

Return:

0	Successful
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoClear(DWORD dwDoID, PASSW bPass);

Delphi syntax: **function TEOS_DoClear(dwDoID: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Clears a Data Object value from the CRYPTO-BOX partition. This function is valid for Data Object types containing an expiration counter such as **TEOSDO_EXPIRATION_DATE, TEOSDO_NUMBER_OF_DAYS, TEOSDO_TIME_ALLOWED, TEOSDO_COUNTER, TEOSDO_NET_License**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
PASSW bPass	IN: Admin Password (APW) required for **TEOSDO_NET_License**, for the rest - NULL

Return:

0	Successful
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoUnlimited(DWORD dwDoID, PASSW bPass);

Delphi syntax: **function TEOS_DoUnlimited(dwDoID: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Sets a Data Object value in the CRYPTO-BOX partition to UNLIMITED (no expiration). This function is valid for the following Data Object types: **TEOSDO_EXPIRATION_DATE, TEOSDO_NUMBER_OF_DAYS, TEOSDO_TIME_ALLOWED, TEOSDO_COUNTER, TEOSDO_NET_License**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
PASSW bPass	IN: Admin Password (APW) required for **TEOSDO_NET_License**, for the rest - NULL

Return:

0	Successful
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoVerify(DWORD dwDoID, BYTE* pData, DWORD dwDataSize, DWORD dwParameter, PASSW bPass);

Delphi syntax: **function TEOS_DoVerify(dwDoID: DWORD; pData: PBYTE; dwDataSize: DWORD; dwParameter: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Verifies a Data Object value in the CRYPTO-BOX partition. This function is valid for the following Data Object types: **TEOSDO_EXPIRATION_DATE, TEOSDO_NUMBER_OF_DAYS, TEOSDO_TIME_ALLOWED, TEOSDO_PSW_HASH**, **TEOSDO_APP_CS**, **TEOSDO_APP_HASH.**

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
BYTE * pData	IN: Verification data • for **TEOSDO_TIME_ALLOWED** - decrement data (in seconds) for real-time verification • for **TEOSDO_PSW_HASH** - password value (string) • NULL - not used
DWORD dwDataSize	IN: Size of verification data
DWORD dwParameter	IN: 0 - reserved
PASSW bPass	IN: NULL - reserved

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoOperation(DWORD dwDoID, DWORD dwOperation, BYTE* pData, DWORD dwDataSize, DWORD dwParameter, PASSW bPass);

Delphi syntax: **function TEOS_DoOperation(dwDoID: DWORD; dwOperation: DWORD; pData: PBYTE; dwDataSize: DWORD; dwParameter: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Universal function for operation executed on a Data Object in the CRYPTO-BOX partition.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be > 1
DWORD dwOperation	IN: Data Object operation/method: **TEOSDO_SET** **TEOSDO_INC** **TEOSDO_DEC** **TEOSDO_CLEAR**

	TEOSDO_UNLIMITED
	TEOSDO_VERIFY
	TEOSDO_BIND (=**TEOSDO_INC**)
BYTE * pData	IN/OUT: Operation data
DWORD dwDataSize	IN: Size of operation data
DWORD dwParameter	IN: Operation parameter
PASSW bPass	IN: NULL - reserved

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoSetKey(DWORD dwDoID, PVOID pKey, PVOID pKeyInfo, PASSW bPass);

Delphi syntax: **function TEOS_DoSetKey(dwDoID: DWORD; pKey: POINTER; pKeyInfo: POINTER; bPass: PTPasswd): DWORD; stdcall;**

Sets encryption key. This function is only valid for **TEOSDO_RSA**, **TEOSDO_AES**, **TEOSDO_AES_PRIVATE** and **TEOSDO_AES_SESSION**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
PVOID pKey	IN: Pointer to key value. Depending on Data Object type it's actual type must be: – CBIOS_RSA_KEY* for **TEOSDO_RSA** – CBIOS_AES_KEY* for **TEOSDO_AES** – BYTE [0x20] for **TEOSDO_AES_PRIVATE** and **TEOSDO_AES_SESSION**
PVOID pKeyInfo	IN: Pointer to key info. Depending on Data Object type it's actual type must be: – CBIOS_RSA_KEY_INFO* for **TEOSDO_RSA** – CBIOS_AES_KEY_INFO* for **TEOSDO_AES** – Null (not used) for **TEOSDO_AES_PRIVATE** and **TEOSDO_AES_SESSION** Can be omitted (Null passed). In this case default key info value is assigned.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoClearKey(DWORD dwDoID, PVOID pKeyInfoNew, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoClearKey(dwDoID: DWORD; pKeyInfoNew: POINTER; bPass: PTPasswd): DWORD; stdcall;**

Clears encryption key. This function is only valid for **TEOSDO_RSA**, **TEOSDO_AES**, **TEOSDO_AES_PRIVATE** and **TEOSDO_AES_SESSION**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
PVOID pKeyInfoNew	IN: Pointer to key info. Depending on Data Object type it's actual type must be: – CBIOS_RSA_KEY_INFO* for **TEOSDO_RSA** – CBIOS_AES_KEY_INFO* for **TEOSDO_AES** – Null (not used) for **TEOSDO_AES_PRIVATE** and **TEOSDO_AES_SESSION** Can be omitted (Null passed). In this case default key info value is assigned.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoGenerateEx(DWORD dwDoID, DWORD dwBitsQty , PVOID pRSAPublicKey, PVOID pRSAKeyInfo, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoGenerateEx(dwDoID: DWORD; dwBitsQty: DWORD; pRSAPublicKey: POINTER; pRSAKeyInfo: POINTER; bPass: PTPasswd): DWORD; stdcall;**

Generates encryption key and stores it in the CRYPTO-BOX memory. Public key can be retrieved. This function is only valid for **TEOSDO_RSA**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwBitsQty	IN: Sets key size (in bits). Default is 1024.
PVOID pRSAPublicKey	OUT: Pointer to public RSA key pair (CBIOS_RSA_KEY*). Optional.

PVOID pKeyInfo	IN: Pointer to RSA key info (CBIOS_RSA_KEY_INFO*). Can be omitted (Null passed). In this case default key info value is assigned.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoEncryptRSA(DWORD dwDoID, DWORD dwMode, BYTE* pInBuffer, DWORD dwInBufferLen, BYTE* pOutBuffer, DWORD* pdwOutBufferLen, PASSW bPass);

Delphi syntax: **function TEOS_DoEncryptRSA(dwDoID: DWORD; dwMode: DWORD; pInBuffer: PBYTE; dwInBufferLen: DWORD; pOutBuffer: PBYTE; pdwOutBufferLen: PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Encrypts given buffer with RSA algorithm. This function is only valid for **TEOSDO_RSA**, **TEOSDO_RSA_CLIENT** and **TEOSDO_RSA_DISTRIBUTOR**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwMode	IN: Encryption mode: key mode (CBIOS_RSA_PUBL_KEY or CBIOS_RSA_PRIV_KEY) bitwise summed with padding mode (CBIOS_RSA_MARX_PADDING or CBIOS_RSA_RSAREF_PADDING).
BYTE * pInBuffer	IN: Source buffer
DWORD dwInBufferLen	IN: Source buffer size
BYTE * pOutBuffer	IN/Out: Receiving buffer.
DWORD * pdwInBufferLen	IN/Out: Pointer to Receiving buffer size. If specified size is not enough function returns CBIOS_ERR_WRONG_PARAM and pdwInBufferLen returns needed buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoDecryptRSA(DWORD dwDoID, DWORD dwMode, BYTE* pInBuffer, DWORD dwInBufferLen, BYTE* pOutBuffer, DWORD* pdwOutBufferLen, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoDecryptRSA(dwDoID: DWORD; dwMode: DWORD; pInBuffer: PBYTE; dwInBufferLen: DWORD; pOutBuffer: PBYTE; pdwOutBufferLen: PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Decrypts given buffer with RSA algorithm. This function is only valid for **TEOSDO_RSA**, **TEOSDO_RSA_CLIENT** and **TEOSDO_RSA_DISTRIBUTOR**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwMode	IN: Encryption mode: key mode (CBIOS_RSA_PUBL_KEY or CBIOS_RSA_PRIV_KEY) bitwise summed with padding mode (CBIOS_RSA_MARX_PADDING or CBIOS_RSA_RSAREF_PADDING).
BYTE * pInBuffer	IN: Source buffer
DWORD dwInBufferLen	IN: Source buffer size
BYTE * pOutBuffer	IN/Out: Receiving buffer.
DWORD * pdwInBufferLen	IN/Out: Pointer to Receiving buffer size. If specified size is not enough function returns CBIOS_ERR_WRONG_PARAM and pdwInBufferLen returns needed buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoCryptAES(DWORD dwDoID, DWORD dwMode, BYTE* pIV, BYTE* pInBuffer, BYTE* pOutBuffer, DWORD dwBufferLen, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoCryptAES(dwDoID: DWORD; dwMode: DWORD; pIV: PBYTE; pInBuffer: PBYTE; pOutBuffer: PBYTE; dwBufferLen: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Encrypts (decrypts) given buffer with AES algorithm. This function is only valid for **TEOSDO_AES, TEOSDO_AES_FIXED, TEOSDO_AES_PRIVATE** and **TEOSDO_AES_SESSION**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwMode	IN: Key mode (CBIOS_AES_OFB, CBIOS_AES_CBC_ENCRYPT or CBIOS_AES_CBC_DECRYPT).
BYTE * pIV	IN: Initialization vector (BYTE [0x10])
BYTE * pInBuffer	IN: Source buffer. It's size is dwBufferLen.
BYTE * pOutBuffer	IN/Out: Receiving buffer. The same size as source buffer.
DWORD dwBufferLen	IN: Buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoCalculateFileSignature(DWORD dwDoID, DWORD dwMode, PVOID pRSAKey, const char* szSourceFileName, const char* szSignatureFileName, PASSW bPass);

Delphi syntax: **function TEOS_DoCalculateFileSignature(dwDoID: DWORD; dwMode: DWORD; pRSAKey: PVOID; szSourceFileName : pchar; szSignatureFileName : pchar; bPass: PTPasswd): DWORD; stdcall;**

Calculates signature – calculates MD5 hash and encrypts (private) – of a file given the source file name and saves it to another file with signature file name. This function is only valid for **TEOSDO_RSA**, **TEOSDO_RSA_CLIENT** and **TEOSDO_RSA_DISTRIBUTOR**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwMode	IN: Padding mode: CBIOS_RSA_MARX_PADDING or CBIOS_RSA_RSAREF_PADDING.
PVOID pRSAKey	IN: RSA key (pointer to CBIOS_RSA_KEY). Optional. If it is specified (not Null) given RSA key is used as encryption key and dwDoID is ignored (i.e. function can be used in "offline" mode, whithout creating reference).
const char* szSourceFileName	IN: Source file name
const char* szSignatureFileName	IN: Signature file name. Optional. If this parameter is omitted (Null) signature file name will be source file name + ".sig".
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoValidateFileSignature(DWORD dwDoID, DWORD dwMode, const char* szSourceFileName, const char* szSignatureFileName, PASSW bPass);

Delphi syntax: **function TEOS_DoValidateFileSignature(dwDoID: DWORD; dwMode: DWORD; szSourceFileName : pchar; szSignatureFileName : pchar; bPass: PTPasswd): DWORD; stdcall;**

Validates signature – decrypts (public) data from signature file name and compares it with MD5 hash of a file given the source file name. This function is only valid for **TEOSDO_RSA**, **TEOSDO_RSA_CLIENT** and **TEOSDO_RSA_DISTRIBUTOR**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwMode	IN: Padding mode: CBIOS_RSA_MARX_PADDING or CBIOS_RSA_RSAREF_PADDING.
const char* szSourceFileName	IN: Source file name
const char* szSignatureFileName	IN: Signature file name. Optional. If this parameter is omitted (Null) signature file name will be source file name + ".sig".
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
TEOS_ERR_CORRUPTED_DATA	Signature mismatch
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoEncryptRSAEx(DWORD dwDoID, BYTE* pInBuffer, DWORD dwInBufferLen, BYTE* pOutBuffer, DWORD* pdwOutBufferLen, PASSW bPass);

Delphi syntax: **function TEOS_DoEncryptRSAEx(dwDoID: DWORD; pInBuffer: PBYTE; dwInBufferLen: DWORD; pOutBuffer: PBYTE; pdwOutBufferLen: PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Encrypts given buffer with RSA algorithm. This function is only valid for **TEOSDO_RSA_EX**

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1

BYTE * pInBuffer	IN: Source buffer
DWORD dwInBufferLen	IN: Source buffer size
BYTE * pOutBuffer	IN/Out: Receiving buffer.
DWORD * pdwInBufferLen	IN/Out: Pointer to Receiving buffer size. If specified size is not enough function returns CBIOS_ERR_WRONG_PARAM and pdwInBufferLen returns needed buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoDecryptRSAEx(DWORD dwDoID, BYTE* pInBuffer, DWORD dwInBufferLen, BYTE* pOutBuffer, DWORD* pdwOutBufferLen, PASSW bPass);

Delphi syntax: **function TEOS_DoDecryptRSAEx(dwDoID: DWORD; pInBuffer: PBYTE; dwInBufferLen: DWORD; pOutBuffer: PBYTE; pdwOutBufferLen: PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Decrypts given buffer with RSA algorithm. This function is only valid for **TEOSDO_RSA_EX**

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
BYTE * pInBuffer	IN: Source buffer
DWORD dwInBufferLen	IN: Source buffer size
BYTE * pOutBuffer	IN/Out: Receiving buffer.
DWORD * pdwInBufferLen	IN/Out: Pointer to Receiving buffer size. If specified size is not enough function returns CBIOS_ERR_WRONG_PARAM and pdwInBufferLen returns needed buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoCryptAESEx(DWORD dwDoID, BYTE* pIV, BYTE* pInBuffer, BYTE* pOutBuffer, DWORD dwBufferLen, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoCryptAESEx(dwDoID: DWORD; pIV: PBYTE; pInBuffer: PBYTE; pOutBuffer: PBYTE; dwBufferLen: DWORD; bPass: PTPasswd): DWORD; stdcall;**

Encrypts (decrypts) given buffer with AES algorithm. This function is only valid for **TEOSDO_AES_EX**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
BYTE * pIV	IN: Initialization vector (BYTE [0x10]) It's usage dependents on **TEOSDO_AES_EX** value.
BYTE * pInBuffer	IN: Source buffer. It's size is dwBufferLen.
BYTE * pOutBuffer	IN/Out: Receiving buffer. The same size as source buffer.
DWORD dwBufferLen	IN: Buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoSetKeys(DWORD dwDoID, PVOID pKeyA, PVOID pKeyAInfo, PVOID pKeyB, PVOID pKeyBInfo, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoSetKeys(dwDoID: DWORD; pKeyA: POINTER; pKeyAInfo: POINTER; pKeyB: POINTER; pKeyBInfo: POINTER; bPass: PTPasswd): DWORD; stdcall;**

Sets encryption keys (Side A and Side B). This function is only valid for **TEOSDO_SIGNATURE**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
PVOID pKeyA, pKeyB	IN: Pointer to key structure CBIOS_RSA_KEY. If NULL – ignored.
PVOID pKeyAInfo, pKeyBInfo	IN: Pointer to key info structure CBIOS_RSA_KEY_INFO. Can be omitted (Null passed). In this case default key info value is assigned.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoGenerateA(B)(DWORD dwDoID, DWORD dwBitsQty, PVOID pRSAPublicKey, PVOID pRSAKeyInfo, PASSW bPass);

Delphi syntax: **function TEOS_DoGenerateA(B)(dwDoID: DWORD; dwBitsQty: DWORD; pRSAPublicKey: POINTER; pRSAKeyInfo: POINTER; bPass: PTPasswd): DWORD; stdcall;**

Generates encryption key Side A (or Side B for B-function) and stores it in CRYPTO-BOX memory. Public key can be retrieved. This function is only valid for **TEOSDO_SIGNATURE** .

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
DWORD dwBitsQty	IN: Sets key size (in bits). Default is 1024.
PVOID pRSAPublicKey	OUT: Pointer to public RSA key pair (CBIOS_RSA_KEY*). Optional.
PVOID pKeyInfo	IN: Pointer to RSA key info (CBIOS_RSA_KEY_INFO*). Can be omitted (Null passed). In this case default key info value is assigned.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoCalculateDigitalSignatureF(DWORD dwDoID, const char* szSourceFileName, const char* szSignatureFileName, PASSW bPass);

Delphi syntax: **function TEOS_DoCalculateDigitalSignatureF(dwDoID: DWORD; szSourceFileName : pchar; szSignatureFileName : pchar; bPass: PTPasswd): DWORD; stdcall;**

Calculates signature – calculates MD5 hash and encrypts with up to two RSA keys (as specified by **TEOSDO_SIGNATURE** value) – of a file given the source file name and saves it to another file with signature file name. This function is only valid for **TEOSDO_SIGNATURE**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1

const char* szSourceFileName	IN: Source file name
const char* szSignatureFileName	IN: Signature file name. Optional. If this parameter is omitted (Null) signature file name will be source file name + ".sig".
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoValidateDigitalSignatureF(DWORD dwDoID, const char* szSourceFileName, const char* szSignatureFileName, DWORD* pdwTimeStamp, PASSW bPass);

Delphi syntax: **function TEOS_DoValidateDigitalSignatureF(dwDoID: DWORD; szSourceFileName : pchar; szSignatureFileName : pchar; pdwTimeStamp :PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Validates signature – extracts MD5 hash (using combination of up to two RSA keys as specified by **TEOSDO_SIGNATURE** value) from signature file name and compares it with MD5 hash of a file given the source file name. This function is only valid for **TEOSDO_SIGNATURE**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
const char* szSourceFileName	IN: Source file name
const char* szSignatureFileName	IN: Signature file name. Optional. If this parameter is omitted (Null) signature file name will be source file name + ".sig".
DWORD* pdwTimeStamp	OUT: Buffer receiving time stamp – for additional validation
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
TEOS_ERR_CORRUPTED_DATA	Signature mismatch
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoCalculateDigitalSignature(DWORD dwDoID, BYTE* pInBuffer, DWORD dwInBufferLen, BYTE* pOutBuffer, DWORD* pdwOutBufferLen, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoCalculateDigitalSignature(dwDoID: DWORD; pInBuffer: PBYTE; dwInBufferLen: DWORD; pOutBuffer: PBYTE; pdwOutBufferLen: PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Calculates signature – calculates MD5 hash and encrypts with up to two RSA keys (as specified by **TEOSDO_SIGNATURE** value) – of a given (pInBuffer) buffer and places it to another (pOutBuffer) buffer. This function is only valid for **TEOSDO_SIGNATURE**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
BYTE * pInBuffer	IN: Source buffer to calculate signature from
DWORD dwInBufferLen	IN: Source buffer size
BYTE * pOutBuffer	IN/Out: Receiving buffer for signature calculated.
DWORD * pdwInBufferLen	IN/Out: Pointer to Receiving buffer size. If specified size is not enough function returns CBIOS_ERR_WRONG_PARAM and pdwInBufferLen returns needed buffer size.
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
error code	See error code description in chapter *1.3*

DWORD WINAPI TEOS_DoValidateDigitalSignature(DWORD dwDoID, BYTE* pInBuffer, DWORD dwInBufferLen, BYTE* pInBuffer2, DWORD dwInBuffer2Len, DWORD* pdwTimeStamp, PASSW bPass);

<u>*Delphi syntax:*</u> **function TEOS_DoValidateDigitalSignature(dwDoID: DWORD; pInBuffer: PBYTE; dwInBufferLen: DWORD; pInBuffer2: PBYTE; dwInBuffer2Len: DWORD; pdwTimeStamp :PDWORD; bPass: PTPasswd): DWORD; stdcall;**

Validates signature – extracts MD5 hash (using combination of up to two RSA keys as specified by **TEOSDO_SIGNATURE** value) from **pInBuffer** buffer and compares it with MD5 hash of a **pInBuffer2**. This function is only valid for **TEOSDO_SIGNATURE**.

Parameters:

DWORD dwDoID	IN: Data Object ID, must be >= 1
BYTE * pInBuffer	IN: Source buffer to extract MD5 hash from

DWORD dwInBufferLen	IN: pInBuffer size
BYTE * pInBuffer2	IN: Source buffer to compare MD5 hash
DWORD dwInBuffer2Len	IN: pInBuffer2 size
DWORD* pdwTimeStamp	OUT: Buffer receiving time stamp – for additional validation
PASSW bPass	IN: Password (user or admin). Optional.

Return:

0	Success
TEOS_ERR_CORRUPTED_DATA	Signature mismatch
error code	See error code description in chapter *1.3*

1.3. DO API Error Codes (see also teosdo.h)

For standard CBIOS API error codes, see cbios.h.

API Constant	Error Codes		Description
	dec	hex	
TEOS_ERR_NOT_IMPLEMENTED	65	41	Function is not implemented
TEOS_ERR_WRONG_PARAMETER	66	42	Input parameter is incorrect
TEOS_ERR_BUFFER_TOO_SHORT	67	43	Buffer is too small to receive DataObject value
TEOS_ERR_CORRUPTED_DATA	68	44	Data Object is corrupted or not initialized
TEOS_ERR_MODIFIED_DATE	69	45	PC system date was modified
TEOS_ERR_EXPIRED	70	46	Data Object is expired
TEOS_ERR_NO_HANDLES	71	47	Too many Data Object's references
TEOS_ERR_NO_MEMORY	72	48	Memory allocation error
TEOS_ERR_DO_NOT_FOUND	73	49	Data Object not found (unknown DO identifier)
TEOS_ERR_NO_USAGE_COUNTER	74	4A	Usage Counter limit reached
TEOS_ERR_ALREADY_EXISTS	75	4B	Data Object's reference already exists
TEOS_ERR_CRC_ERROR	76	4C	Application CRC or password hash mismatch or invalid data (Data Object's map)
TEOS_ERR_FILE_NOT_FOUND	77	4D	File not found (Data Object's map file)
TEOS_ERR_FILE_ACCESS_ERROR	78	4E	File access error (Data Object's map file)
TEOS_ERR_REF_NO_BASE_FIELD	79	4F	Invalid data (Data Object's map)
TEOS_ERR_READ_ONLY	80	50	Data Object is read only (e.g. result of SET for TEOSDO_AES_FIXED)
TEOS_ERR_NOT_SUPPORTED	81	51	Function is not supported

2. 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
Fax:	(+1) 678-730-1804
E-Mail:	contact@marx.com

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

Sales:	sales-de@marx.com
Support:	support-de@marx.com
Phone:	+49 (0) 8403 9295-0
Fax:	+49 (0) 8403 9295-40
E-Mail:	contact-de@marx.com

3. Alphabetical Index

0-23May010ks(CBIOS DO API Reference).odt