**X-Ways Python Command Reference**

collected from [www.x-ways.net/forensics/x-tensions/XWF\_functions.html](http://www.x-ways.net/forensics/x-tensions/XWF_functions.html) and c++ source code for the public python x-tension.

no guarantee of completeness!

All commands and information are thought by X-Ways AG, Germany

|  |  |  |  |
| --- | --- | --- | --- |
| ***AddComment*** | ***BOOL XWF\_AddComment(*** ***LONG***nItemID,***LPWSTR****lpComment,****DWORD***nFlagsHowToAdd***);*** | | *Adds the specified comment to the specified item.*  nFlagsHowToAdd: *0x01: append to any existing comment, do not replace it 0x02: append to any existing comment, do not replace it, and insert a line break as a delimiter* |
| ***AddToReportTable*** | ***LONG XWF\_AddToReportTable(*** ***LONG***nItemID,***LPWSTR****lpReportTableName,****DWORD***nFlags***);*** | | *Associates the specified file with the specified report table. If the report table does not exist yet in the currently active case, it will be created. Returns 1 if the file was successfully and newly associated with the report table, 2 if that association existed before, or 0 in case of failure, for example if no case is active or if the volume that the file is contained in is not an evidence object in the active case.*  nFlags: *Flags for the newly created report table. They have no effect if the report table already existed before. 0x01: show as created by application, not by examiner 0x02: select for inclusion in report 0x04: select for filtering 0x08: select for future manual report table associations* |
| ***CreateItem*** | ***LONG XWF\_CreateItem(*** ***LPWSTR****lpName,****DWORD***nCreationFlags ***);*** | | *Creates a new item (file or directory) in the volume snapshot. May be called when refining the volume snapshot. Returns the ID of the newly created item, or -1 if an error occurred (e.g. out of memory). Should be followed by calls to XWF\_SetItemParent, XWF\_SetItemSize, XWF\_SetItemInformation, and/or XWF\_SetItemOfs. If via XWF\_SetItemParent you make the new file a child object of a file (not directory), you are responsible for setting the parent's XWF\_ITEM\_INFO\_FLAG\_HASCHILDREN flag.*  *For example, if you are creating a file carved from the sectors of the evidence object, you can specify the file size using XWF\_SetItemSize and the start offset via the*nDefOfs*parameter (must be negative) using XWF\_SetItemOfs.*  nCreationFlags:0x00000001: for performance reasons, set if many more items are expected to be created |
| ***GetComment*** | ***LPWSTR******XWF\_GetComment(*** ***LONG***nItemID***);*** | | *Retrieves a pointer to the comment of an item, if any, otherwise NULL. This pointer is guaranteed to be valid only at the time when you retrieve it.* |
| ***GetItemCount*** | ***DWORD XWF\_GetItemCount(*** ***LPVOID***pReserved ***);*** | | *Retrieves the number of items in the current volume snapshot (files and directories). Item IDs are consecutive 0-based. That means the ID of the first item has the ID 0 and the last item in a volume snapshot has the ID (GetItemCount-1). You address each and every item in that range, be it a file or directory, by specifying its ID.* |
| ***GetItemFirstDataSector*** | ***VOID XWF\_GetItemOfs(*** ***LONG***nItemID,***LPINT64***lpDefOfs,***LPINT64***lpStartSector***);*** | | *Retrieves the offset of the file system data structure (e.g. NTFS FILE record) where the item is defined. If negative, the absolute value is the offset where a carved file starts on the volume. 0 if an error occurred. 0xFFFFFFFF if not available/not applicable.*  *Also retrieves the number of the sector from the point of the volume in which the data of the item starts.* |
| ***GetFileSystemInfoOffset*** |
| ***All two commands above refer to the same X-Ways Command*** | | | |
| ***GetItemIDForSector*** | | ***BOOL XWF\_GetSectorContents(*** ***HANDLE***hVolume,***INT64***nSectorNo,***LPWSTR***lpDescr,***LPLONG***lpItemID ***);*** | *Retrieves information about a certain sector on a volume. Returns FALSE if the sector belongs to an unused/free cluster, otherwise TRUE.*  *lpDescr: Retrieves a textual description of what this sector is used for. Can be the name and path of a file or something like "FAT 1". May be language specific. Use a buffer that has space for 511 characters and a terminating null.*  *lpItemID: Optional. Retrieves the ID of the item in the volume snapshot that the sector is allocated to, if any, otherwise -1.* |
| ***GetItemInformation*** | | ***INT64 XWF\_GetItemInformation(*** ***LONG***nItemID,***LONG***nInfoType,***LPBOOL***lpSuccess, ***);*** | *Returns information about an item (file or directory) as stored in the volume snapshot, such as the original ID or attributes that the item had in its defining file system. What information is actually returned depends on nInfoType. The function indicates success or failure via lpSuccess. This parameter may be NULL if not required. All timestamps are transferred in Windows FILETIME format.*  *#define XWF\_ITEM\_INFO\_ORIG\_ID 1 #define XWF\_ITEM\_INFO\_ATTR 2 #define XWF\_ITEM\_INFO\_FLAGS 3 #define XWF\_ITEM\_INFO\_DELETION 4 #define XWF\_ITEM\_INFO\_CLASSIFICATION 5 // e.g. extracted e-mail message, alternate data stream, etc. #define XWF\_ITEM\_INFO\_LINKCOUNT = 6 // hard-link count #define XWF\_ITEM\_INFO\_COLORANALYSIS\* = 7 // v17.2 and later, skin color percentage, <0: n/a, -2: error, -3: b/w or grayscale, -4: irrelevant #define XWF\_ITEM\_INFO\_PIXELINDEX\* = 8 // v18.9 and later, more information below #define XWF\_ITEM\_INFO\_FILECOUNT = 11 // how many child objects exist recursively that are files #define XWF\_ITEM\_INFO\_EMBEDDEDOFFSET = 16 // v17.7 and later, for a file linearly embedded within another file, offset in that file #define XWF\_ITEM\_INFO\_CREATIONTIME = 32 #define XWF\_ITEM\_INFO\_MODIFICATIONTIME = 33 #define XWF\_ITEM\_INFO\_LASTACCESSTIME = 34 #define XWF\_ITEM\_INFO\_ENTRYMODIFICATIONTIME = 35 #define XWF\_ITEM\_INFO\_DELETIONTIME = 36 #define XWF\_ITEM\_INFO\_INTERNALCREATIONTIME = 37*   |  |  | | --- | --- | | *Flags that are returned for XWF\_ITEM\_INFO\_FLAGS: 0x00000001: is a directory 0x00000002: has child objects (for files only)  0x00000004: has subdirectories (for directories only) 0x00000008: is a virtual item  0x00000010: hidden by examiner 0x00000020: tagged 0x00000040: tagged partially 0x00000080: viewed by examiner*  *0x00000100: file system timestamps not in UTC 0x00000200: internal creation timestamp not in UTC  0x00000400: FAT timestamps  0x00000800: originates from NTFS 0x00001000: UNIX world attributes 0x00002000: has examiner comment 0x00004000: has extracted metadata 0x00008000: file contents totally unknown*  *0x00010000: file contents partially unknown 0x00020000: reserved 0x00040000: hash 1 already computed 0x00080000: has duplicates 0x00100000: hash 2 already computed (since v18.0) 0x00200000: known good hash category 0x00400000: known bad hash category 0x00600000: known, either good or bad (both flags!, v18.9+) 0x00800000: found in volume shadow copy*  *0x01000000: deleted files with known original contents 0x02000000: file format consistency OK 0x04000000: file format consistency not OK 0x10000000: file archive already explored (v17.6+) 0x20000000: e-mail archive or video already processed (v17.6+) 0x40000000: embedded data already uncovered (v17.6+) 0x80000000: metadata extraction already applied (v17.6+)*  *0x100000000: file embedded in other file linearly (v17.7+)\* 0x200000000: file whose contents is stored externally (v17.7+)\* 0x400000000: alternative data /a via XWF\_OpenItem (v18.9+)\** | *Deletion status returned for XWF\_ITEM\_INFO\_DELETION: 0 = existing >0 = not existing 1 = previously existing, possibly recoverable 2 = previously existing, first cluster overwritten or unknown  3 = renamed/moved, possibly recoverable 4 = renamed/moved, first cluster overwritten or unknown 5 = carved file (since v19.3 SR-3, used to be 1)*  *Classification values for XWF\_ITEM\_INFO\_CLASSIFICATION: 0x00: normal file 0x04: HFS resource fork 0x08: NTFS alternate data stream 0x0A: NTFS non-directory index 0x0B: NTFS bitmap attribute 0x10: NTFS general logged utility stream 0x11: NTFS EFS logged utility stream 0xF5: e-mail related 0xF6: excerpt 0xF7: manually attached 0xF8: video still 0xF9: e-mail attachment 0xFA: e-mail message 0xFD: INDX record remnant 0xFE: session root directory in CDFS/UDF* |   *XWF\_ITEM\_INFO\_PIXELINDEX: This is in indicator of the pixel count of a raster image. It is the square root of width × height in pixels, divided by 20. 0: not yet computed or not a picture. 1: <= 0,02 KP. 254 = 16.5 MP. 255 (maximum) = even larger.* |
| ***GetOpenFileName*** | | *unknown* | *Open a window to choose a filename* |
| ***GetReportTableAssocs*** | | ***DWORD XWF\_GetReportTableAssocs(*** ***LONG***nItemID*,****LPWSTR****lpBuffer,****LONG****nBufferLen* ***);*** | *Retrieves the names of the report tables that the specified item is associated with. The names are delimited with comma and space. If the buffer was filled completely, that likely means that the specified buffer length was insufficient. In v17.6 SR-7 and later, returns the total number of associations of that item, and lpBuffer may be NULL.* |
| ***GetSaveFileName*** | | *unknown* | *Open a window to choose a filename* |
| ***GetSectorContentsString*** | | ***BOOL XWF\_GetSectorContents(*** ***HANDLE***hVolume,***INT64***nSectorNo,***LPWSTR***lpDescr,***LPLONG***lpItemID ***);*** | *Retrieves information about a certain sector on a volume. Returns FALSE if the sector belongs to an unused/free cluster, otherwise TRUE.*  *lpDescr: Retrieves a textual description of what this sector is used for. Can be the name and path of a file or something like "FAT 1". May be language specific. Use a buffer that has space for 511 characters and a terminating null.*  *lpItemID: Optional. Retrieves the ID of the item in the volume snapshot that the sector is allocated to, if any, otherwise -1.* |
| ***GetVolumeBytesPerSector*** | | ***VOID XWF\_GetVolumeInformation(*** ***HANDLE***hVolume,***LPLONG***lpFileSystem,***LPDWORD***lpBytesPerSector,***LPDWORD***lpSectorsPerCluster,***PINT64***lpClusterCount,***PINT64***lpFirstClusterSectorNo ***);*** | *Retrieves various information about the volume. All parameters are optional.*   |  |  | | --- | --- | | *nFileSystem: 9=main memory 8=CDFS 7=opened through OS 6=XWFS 5=UDF 4=exFAT 3=FAT32 2=FAT16 1=FAT12 0=Unknown* | *-1=NTFS -2=HPFS -3=Ext2 -4=Ext3 -5=ReiserFS -6=Reiser4 -7=Ext4 -9=JFS -10=XFS -11=UFS -12=HFS -13=HFSPlus -15=NTFS Bitlocker -16=physical disk, potentially partitioned* | |
| ***GetVolumeClusterCount*** | |
| ***GetVolumeFileSystem*** | |
| ***GetVolumeFirstClusterSectorNo*** | |
| ***GetVolumeName*** | |
| ***All five commands above refer to the same X-Ways Command*** | | | |
| ***GetVolumeSectorsPerCluster*** | ***VOID XWF\_GetVolumeName(*** ***HANDLE***hVolume,***LPWSTR****lpString,****DWORD****nType* ***);*** | | *Retrieves the name of the volume in UTF-16, 255 characters at most. 3 types of names are available (1, 2 or 3). For example, 3 can be more generic than 2 ("Hard disk 1" instea* |
| ***HideProgress*** | ***VOID XWF\_HideProgress( );*** | | *Closes the progress indicator window.* |
| ***OutputMessage*** | ***VOID XWF\_OutputMessage(*** ***LPWSTR****lpMessage,****DWORD***nFlags***);*** | | *Outputs the specified message in the Messages window. You may use this function for example to alert the user of errors or to output debug information.*  *nFlags: 0x00000001: append without line break (will be delimited from the previous message with a space instead) 0x00000002: don't log this error message in msglog.txt even if logging is active by default 0x00000004: lpMessage points to an ANSI string, not a Unicode string (v16.5 and later) 0x00000010: output the message as an entry in the case log, not in the Messages window (v19.4 and later), flag is ignored if no case is active, may be combined with the 0x4 flag* |
| ***ProcessMessages*** | ***VOID XWF\_OutputMessage(*** ***LPWSTR****lpMessage,****DWORD***nFlags***);*** | | *Outputs the specified message in the Messages window. You may use this function for example to alert the user of errors or to output debug information.*  *nFlags: 0x00000001: append without line break (will be delimited from the previous message with a space instead) 0x00000002: don't log this error message in msglog.txt even if logging is active by default 0x00000004: lpMessage points to an ANSI string, not a Unicode string (v16.5 and later) 0x00000010: output the message as an entry in the case log, not in the Messages window (v19.4 and later), flag is ignored if no case is active, may be combined with the 0x4 flag* |
| ***Read*** | ***DWORD XWF\_Read(*** ***HANDLE***hVolumeOrItem*,****INT64***nOffset*,****LPVOID***lpBuffer*,****DWORD***nNumberOfBytesToRead,***);*** | | *Reads the specified number of bytes from the specified position in the specified volume or item into the specified buffer. Returns the number of bytes read.* |
| ***SetItemOfs*** | ***VOID XWF\_SetItemOfs(*** ***LONG***nItemID,***INT64***nDefOfs,***INT64***nStartSector***);*** | | *Sets the above-mentioned offset and sector number.* |
| ***SetItemParent*** | ***VOID XWF\_SetItemParent(*** ***LONG***nChildItemID,***LONG***nParentItemID***);*** | | *Sets the parent of the specified child item. You may specify -1 for the virtual "Path unknown" directory as the parent, or -2 for the "Carved files" directory. If the parent is a file that does not have child objects yet, you should use XWF\_SetItemInformation to mark it has having child objects.* |
| ***SetItemSize*** | ***VOID XWF\_SetItemSize(*** ***LONG***nItemID,***INT64****nSize* ***);*** | | *Sets the size of the item in bytes. -1 means unknown size.* |
| ***GetItemName*** | ***LPWSTR XWF\_GetItemName(*** ***LONG***nItemID***);*** | | *Retrieves a pointer to the null-terminated name of the specified item (file or directory) in UTF-16. You may call XWF\_GetItemName and XWF\_GetItemParent repeatedly until XWF\_GetItemParent returns -1 and concatenate the item names to get the path of an item.* |
| ***GetItemParent*** | ***LONG XWF\_GetItemParent(*** ***LONG***nItemID***);*** | | *Returns the ID of the parent of the specified item, or -1 if the item is the root directory or if for some strange reason no parent object is assigned.* |
| ***GetItemSize*** | ***INT64 XWF\_GetItemSize(*** ***LONG***nItemID***);*** | | *Retrieves the size of the item (file or directory) in bytes. -1 means unknown size.* |