



CndexLink user library documentation

Interface with Cndex Server

INTRODUCTION	6
The Cndex server	8
General description of the functions	9
Compatibility tables	11
Management of the Cndex server and linking up with the CNC	15
ConnectServer_C (obsolete function)	16
ReleaseServer_C (obsolete function)	17
OpenSession_C	18
CloseSession_C	20
Functions for the control of the CNC bootstrap phase	20
BootPhaseEnquiry_C.....	22
BootReboot_C	23
BootShutDown_C	24
BootMode_C	25
BootEnterServer_C (10 Series only)	26
BootLeaveServer_C (10 Series only)	27
GetHWKey_C	28
Variable management	29
ReadVarWord_C.....	30
ReadVarDouble_C	32
WriteVarWord_C	34
WriteVarWordBit_C.....	35
WriteVarDouble_C	36
ReadVarText_C	37
WriteVarText_C	39
PLC oriented functions	40
ReadWarningMsg_C.....	41
GetPLVarWord_C (obsolete function)	42
SetPLVarWord_C (obsolete function)	43
GetPLVarDouble_C (obsolete function)	44
SetPLVarDouble_C (obsolete function)	45
GetPLVarAscii_C (obsolete function)	46
SetPLVarAscii_C (obsolete function).....	47
GetPLStreamWord_C (obsolete function)	48
SetPLStreamWord_C (obsolete function)	50
GetPLStreamDouble_C (obsolete function)	52
SetPLStreamDouble_C (obsolete function)	54
GetAxisTabRecord_C	56
SetAxisTabRecord_C.....	57
GetToolTabRecord_C	58
SetToolTabRecord_C	59
GetOffsetTabRecord_C	60
SetOffsetTabRecord_C.....	61
GetUserTabRecord_C.....	62
SetUserTabRecord_C	63
GetTabFieldDouble_C	64
SetTabFieldDouble_C.....	65
GetTabFieldShort_C.....	66
SetTabFieldShort_C	67
TabSearchDouble_C	68
TabSearchShort_C.....	70
SetTabSem_C	72



CndexLink user library documentation

Interface with Cndex Server

TabSemInfo_C	74
ResetSingleTableII_C	75
LockTableII_C	76
UnLockTableII_C	77
GetOriginTabRecordII_C	78
SetOriginTabRecordII_C	79
GetToolTabRecordII_C	80
SetToolTabRecordII_C	81
GetOffsetTabRecordII_C	82
SetOffsetTabRecordII_C	83
GetUserTabRecordII_C	84
SetUserTabRecordII_C	85
GetMagazineTabRecordII_C	86
SetMagazineTabRecordII_C	87
GetPocketTabRecordII_C	88
SetPocketTabRecordII_C	89
SaveTables_C	90
SaveSingleTable_C	91
RestoreSingleTable_C	92
SaveBackupMemory_C	93
RestoreBackupMemory_C	94
Process dedicated functions	95
Cycle_C	96
SyncroCycle_C	97
Reset_C	98
Hold_C	99
SetFeedManOver_C	100
SetFeedRateOver_C	101
SetFeedRapidOver_C	102
SetSpeedRateOver_C	103
SetManMovDirection_C	104
GetVarJOG_C	105
SetVarJOG_C	106
SetVarUAS_C	107
GetVarRCM_C	108
SetVarRCM_C	109
LoadPTech_C	110
GetPTechSizes_C	112
SetMdiString_C	113
SetProcessMode_C	114
SelectProcess_C	115
GetSelectedProcess_C	116
SelectProcAxis_C	117
SelectPartProgram_C	118
SelectPartProgramFromDrive_C	119
ManagePartProgram_C	120
GetActivePartProgram_C	122
GetActivePartProgramFullPath_C	123
GetPartProgramLines_C	124
GetAxOriginNum_C	125
GetAxesPosition_C	126
GetNcInfo1_C	128



CndexLink user library documentation

Interface with Cndex Server

GetNcInfo2_C	129
GetMarkerInfo_C	130
GetToolNames_C	131
GetProcessStatus_C	132
GetBlkNum_C	133
GetVarE_C (obsolete function)	134
SetVarE_C (obsolete function)	135
GetVarSN_C (obsolete function)	136
SetVarSN_C (obsolete function)	137
GetVarSC_C	138
SetVarSC_C (obsolete function)	139
ReadErrMsg_C	140
ReadPartProgramMsg_C	141
ReadCurrentErrorMsg_C	142
ReadCurrentEmergMsg_C	143
ReadCurrentAnomalyMsg_C	144
GetGCode_C	145
GetMCode_C	146
SkipPProgBlock_C	147
Ese_C	148
EseEx_C	149
AxesRef_C	151
SetProcVarWord_C	152
GetProcVarWord_C	154
Functions for the "Through Mode"	156
DncInit_C	157
DncData_C	158
DncEof_C	159
DncStop_C	160
Generic functions	161
GetAxesInfo3_C	162
GetCodeNumber_C	163
GetOptions_C	164
GetDateTime_C	165
SetDateTime_C	167
GRead_C	169
GWrite_C	170
SetIpAddress_C	171
GetSerialNumber_C	172
ReadRemapDefinitions_C	173
WriteRemapDefinitions_C	174
GetServoPar_C	175
SetServoPar_C	176
GetCNCRegKey_C	177
File system management functions	178
LogFSTransferFile_C	179
LogFSTransferFileW_C	181
LogFSSetSecurityLevel_C	182
LogFSGetSecurityLevel_C	183
LogFSLongFileNames_C	184
LogFSGetNumDrive_C	185
LogFSGetDriveList_C	186



CndexLink user library documentation

Interface with Cndex Server

LogFSGetHiddenDriveList_C	187
LogFSGetDrivePath_C	188
LogFSAddDrive_C	189
LogFSRemoveDrive_C	190
LogFSReloadDriveList_C	191
LogFSCreateDir_C	192
LogFSCreateFile_C	193
LogFSGetFileSize_C	194
LogFSGetFileAttrib_C	195
LogFSSetFileAttrib_C	198
LogFSChangeFileAttrib_C	199
LogFSFindFirst_C	200
LogFSFindNext_C	202
LogFSFindClose_C	203
LogFSRemoveFile_C	204
LogFSRemoveDir_C	205
LogFSRename_C	206
LogFSCopyFile_C	207
LogFSGetInfo_C	208
Functions for long file name management	209
PPInsertName_C	210
PPDeleteName_C	211
PPGetPLogicalName_C	212
PPGetPhysicalName_C	213
PPGetLogicalDir_C	214
PPUpdate_C	215
Functions for management of CANOPEN HILSCHER	216
CANInit_C	217
CANBoard_C	219
CANSync_C	221
CANNMT_C	223
CANReadSDO_C	225
CANWriteSDO_C	227
CANGetEmergency_C	229
CANConsoleCfg_C	232
Command line tools	234
Overview	234
Sintassi	234
Commands	235
Opzioni	236
Examples	236
Description of the structures and definitions	238
Error Management	247
Error class	248
COM error codes (class 1)	249
Cndex server error codes (class 2)	250
NETBIOS network protocol error codes (class 3)	251
10 Series error codes (class 4)	252
FILESYS_CLASS error codes (class 5)	253
FILESYS_ERRNUM_CLASS error codes (class 6)	254
CNC_BOOT_ERR_CLASS error codes (class 7)	255
SERVER_EXCEPTION_CLASS error codes (class 8)	256



CndexLink user library documentation Interface with Cndex Server

CNDEXLINK_DLL_ERR_CLASS error codes (class 9)	257
DLL_INTERFACE_ERR_CLASS error codes (class 10).....	258
SOAP_INTERFACE_ERR_CLASS error codes (class 11)	259
OPENcontrol error codes (classes from 17 to 62)	261

Last updated: January 20, 2016
WinNBI Version: 4.3.4



CndexLinkUser DLL documentation Interface with Cndex Server

INTRODUCTION

All OSAI systems (CNC, GMC, Operator panels, hand held terminals etc.), connected through network, can communicate with each other and with user applications, using a OSAI server called Cndex and developed using the DCOM (Microsoft®) technology.

The Cndex server can be easily managed using an OSAI DLL referred to as CndexLink. CndexLink handles all functions to manage the DCOM server and implements the communication protocol in a transparent user-friendly manner.

This document explains the utilisation of the CndexLink DLL in user mode (CndexLinkUser).

All the components needed to communicate with the CNC are stored in the PC after the installation of the WinNBI product (Windows graphical interface for OSAI systems – code C07) starting with version 2.4 and on the systems after installed the SW specific to the machine.

Starting from WinNBI version 3.1.1 an interface DLL has been added to the WinNBI suite, for use with the Microsoft® .NET development tools.

In particular, for the operation of the communication mechanism, the system must contain the following files:

Cndex.exe	Server Cndex (can also reside in a different PC, other than the one containing the application).
Cndexps.dll	Proxy library for linking to DCOM server
CndexLink.dll	Library for the use of the Cndex server

The following files are available for the generation (compilation and link) of the applications using the Cndex server:

CndexLink.lib	Library to be linked to the application to be able to start and launch the dynamic functions of CndexLink
CndexLinkUser.h	C language "include file" containing the declarations of the functions and the definitions of the constants required by CndexLink
CndexLinkUser.bas	"Include file" for Visual Basic containing the declarations of the functions and the definitions of the constants required by CndexLink
CndexLinkDotNet.dll	Library to use CndexLink inside application based on Microsoft® .NET.

The CndexLink DLL is certified for applications developed in Microsoft C++ and Microsoft Visual Basic 6.0.

The CndexLinkDotNet DLL is certified for applications developed using Microsoft® .NET Framework 2.0 and .NET Compact Framework 2.0



CndexLinkUser DLL documentation Interface with Cndex Server

CndexLink has been developed to be very similar to OSAI DLL S10WLK32.DLL contained in an obsolete application, E66 "Ethernet Mini-Dnc Communication", previously used for communications with Series 10 CNCs.

WARNING:

The DLL can be used to connect only to CNC equipped with A06 "CndexLink communication" (option). If this option is not present in the CNC, the OpenSession_C function will return an error: class DLL_INTERFACE (value 10) code ERR_OPTION_NOT_ENABLED (value 5)



CndexLinkUser DLL documentation Interface with Cndex Server

The Cndex server

As mentioned in the introduction, all OSAI systems can communicate between them through a DCOM server named Cndex.

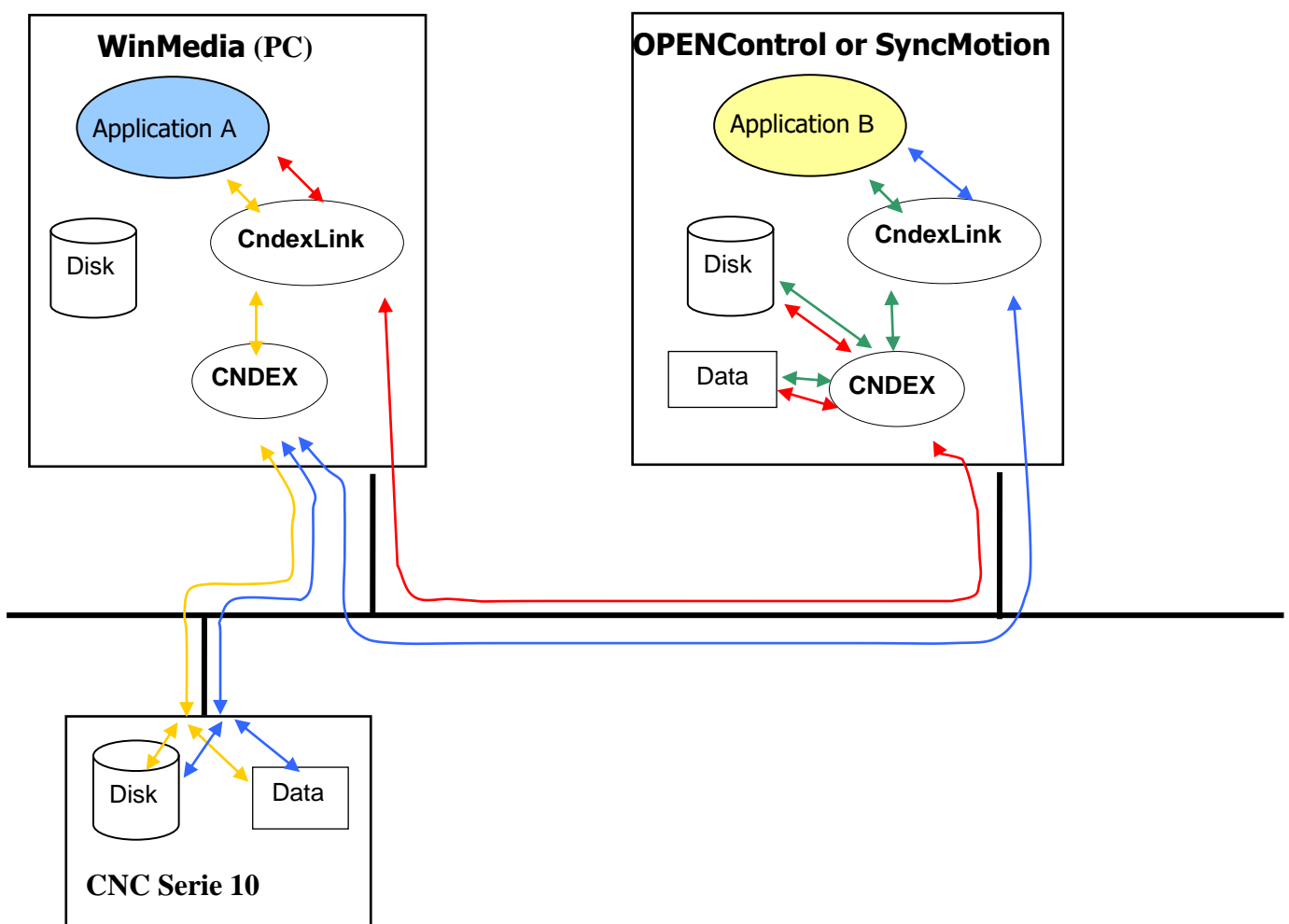
Connecting several systems in a local network, each one of them will be able to communicate with each other.

To do that an application will instantiate a remote object (Cndex) for each system it must connect.

With the exception of 10/Series, the server must be instantiated on the system that must be connected, this operation is done automatically by the OpenSession_C function.

The following picture shown two generic applications (A and B) and their connections used to obtain data from several type of machines. The application A is connected to the local server to communicate with 10/Series CNC (orange arrows). For OPENControl and SyncMotion on the contrary the server is on board of the CNC (connections shown by red arrows).

Application B is connected to local server to collect data from local CNC (green arrows) and connected to the server on board of WinMedia to communicate with the 10/Series CNC (blue arrows).





CndexLinkUser DLL documentation Interface with Cndex Server

General description of the functions

The functions to exchange data with the CNC use the OSAI Cndex server described in the previous paragraphs.

For the .NET applications it is necessary to insert a CndexLinkDotNet interface library reference in the project (click with right mouse button the project name in the VS2005 Solution Explorer, command "add reference", tab "Browse" and selection of the CndexLinkDotNet.dll file supplied with the WinNBI installation). Before you can call the data exchange functions with the CNC it is necessary to link to the Cndex server and open a communication session with the CNC.

For server connection communication session management, see the OpenSession_C and CloseSession_C functions (see also the chapter "CNC bootstrap phase control functions").

Assuming the CNC is already on in RUN mode, the correct sequence for the activation of the data exchange functions is as follows:

C++ and VB6 application

```
OpenSession_C(...)           // Opens a communication session with the CNC
BootPhaseEnquiry_C(...)      // Checks for RUN boot modality
.
.
ReadVarWord_C(...)           // Data exchange functions example
WriteVarDouble_C(...)
.
.
CloseSession_C(...)          // Closes the CNC communication session
```

Applicazioni .NET (C#)

```
using CndexLinkDotNet;       // Permits to use CndexLinkDotNet.
                              // The library must be added to references.

CndexLinkDotNet.Cndex Server; // Reference to Cndex server.
Server = new CndexLinkDotNet.Cndex(); // Create a server instance.

Server.OpenSession_C(...)     // Opens a communication session with the CNC

Server.BootPhaseEnquiry_C(...) // Checks for RUN boot modality
.
.
Server.ReadVarWord_C(...)     // Data exchange functions example
Server.WriteVarDouble_C(...)
.
.
Server.CloseSession_C(...)    // Closes the CNC communication session
```

ConnectServer_C / ReleaseServer_C.



CndexLinkUser DLL documentation Interface with Cndex Server

In previous versions of this library it was mandatory to use the `ConnectServer_C` and `ReleaseServer_C` to create and release the server.

In the current version the two functions are still available for backward compatibility but do not perform any operation.

Error codes

For the management of the errors of all the functions, the CndexLink library uses the return code and two variables (pointed to by `pErrClass` and `pErrNum` in the function prototypes).

The return code is always a "WORD". A value different by zero means that the function has been executed without errors, zero indicates an error during the execution.

When the return code is zero, the variables pointed to by `pErrClass` and `pErrNum` reflect the Class and the code associated with the error that has occurred.

The errors have been organised in different classes to prevent the error codes coming from different levels from overlapping and to be able to identify the type of error at once.

Each class identifies a level in the hierarchy of modules used to communicate with the CNC.

For a detailed description of error Classes and Codes see the paragraphs on "Error Management".

Function arguments

The functions of the CndexLink DLL are described in the following paragraphs, specifying, for each function, the "direction" of transfer of the individual parameters, i.e., who, or what, assigns a parameter.

The parameters can be:

- [in]** parameters compiled by a user and supplied to the function as inputs
- [out]** output parameters from a function (all of which are pointers to a user memory area). The memory is compiled by the function during execution.
- [in,out]** parameters serving both as inputs to and as outputs from a function (all of which are pointers to a user memory area). The memory is compiled by the user before calling the function and by the function during execution.

The functions described in this manual apply to 10 Series MC and Power GP, OPENControl, SyncMotion, Top5 and PC systems.

The compatibility of each function with the systems is specified in the compatibility tables in the following paragraph.

The syntax used to describe the prototype of each function in this manual is the C++ language one.

The Visual Basic 6.0 function prototypes can be found in the `ndexLinkUser.bas` module (see introduction).

The syntax of the .NET functions is available as context help during development of the application.



CndexLinkUser DLL documentation

Interface with Cndex Server

Compatibility tables

The following tables show the compatibility between the functions and the OSAI systems.
The functions marked with • are available for the system reported in the column header.
The gray rows indicate the obsolete (but still available) functions.

Management of the Cndex Server and linking up with the CNC						
	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
ConnectServer_C	•	•	•	•	•	•
ReleaseServer_C	•	•	•	•	•	•
OpenSession_C	•	•	•	•	•	•
CloseSession_C	•	•	•	•	•	•

Functions for the control of the CNC bootstrap phase						
	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
BootPhaseEnquiry_C	•	•	•	•	•	
BootReboot_C	•	•	•			
BootShutDown_C	•	•				
BootMode_C	•	•	•			
BootEnterServer_C	•	•				
BootLeaveServer_C	•	•				
GetHWKey_C	•	•	•	•		

Variable management						
	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
ReadVarWord_C	•	•	•	•	•	
ReadVarDouble_C	•	•	•	•	•	
WriteVarWord_C	•	•	•	•	•	
WriteVarWordBit_C	•	•	•	•	•	
WriteVarDouble_C	•	•	•	•	•	
ReadVarText_C	•	•	•	•	•	
WriteVarText_C	•	•	•	•	•	

PLC dedicated functions						
	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
ReadWarningMsg_C	•	•	•			
GetPLVarWord_C	•	•				
SetPLVarWord_C	•	•				
GetPLVarDouble_C	•	•				
SetPLVarDouble_C	•	•				
GetPLVarAscii_C	•	•				
SetPLVarAscii_C	•	•				
GetPLStreamWord_C	•	•				
SetPLStreamWord_C	•	•				
GetPLStreamDouble_C	•	•				
SetPLStreamDouble_C	•	•				
GetAxisTabRecord_C	•	•				
SetAxisTabRecord_C	•	•				
GetToolTabRecord_C	•	•				
SetToolTabRecord_C	•	•				
GetOffsetTabRecord_C	•	•				



CndexLinkUser DLL documentation Interface with Cndex Server

SetOffsetTabRecord_C	•	•	
GetUserTabRecord_C	•	•	
SetUserTabRecord_C	•	•	
GetTabFieldDouble_C	•	•	
SetTabFieldDouble_C	•	•	
GetTabFieldShort_C	•	•	
SetTabFieldShort_C	•	•	
TabSearchDouble_C	•	•	
TabSearchShort_C	•	•	
SetTabSem_C	•	•	
TabSemInfo_C	•	•	
ResetSingleTableII_C			•
LockTableII_C			•
UnLockTableII_C			•
GetOriginTabRecordII_C			•
SetOriginTabRecordII_C			•
GetToolTabRecordII_C			•
SetToolTabRecordII_C			•
GetOffsetTabRecordII_C			•
SetOffsetTabRecordII_C			•
GetUserTabRecordII_C			•
SetUserTabRecordII_C			•
SaveTables_C			•
SaveSingleTable_C			•
RestoreSingleTable_C			•
SaveBackupMemory_C			•
RestoreBackupMemory_C			•

Process dedicated functions

	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
Cycle_C	•		•			
SyncroCycle_C	•					
Reset_C	•		•			
Hold_C	•		•			
SetFeedManOver_C	•		•			
SetFeedRateOver_C	•		•			
SetFeedRapidOver_C	•		•			
SetSpeedRateOver_C	•		•			
SetManMovDirection_C	•		•			
GetVarJOG_C	•		•			
SetVarJOG_C	•		•			
SetVarUAS_C	•		•			
GetVarRCM_C	•		•			
SetVarRCM_C	•		•			
SetMdiString_C	•		•			
SetProcessMode_C	•		•			
SelectProcess_C	•		•			
GetSelectedProcess_C	•		•			
SelectProcAxis_C	•		•			
SelectPartProgram_C	•		•			
GetActivePartProgram_C	•		•			
GetActivePartProgramFullPath_C			•			
GetPartProgramLines_C	•		•			
GetAxOriginNum_C	•		•			
GetAxesPosition_C	•		•			
GetNcInfo1_C	•		•			
GetNcInfo2_C	•		•			



CndexLinkUser DLL documentation Interface with Cndex Server

GetMarkerInfo_C		•
GetToolNames_C	•	•
GetProcessStatus_C	•	•
GetBlkNum_C	•	•
GetVarE_C	•	•
SetVarE_C	•	•
GetVarSN_C	•	•
SetVarSN_C	•	•
GetVarSC_C	•	•
SetVarSC_C	•	•
ReadErrMsg_C	•	
ReadPartProgramMsg_C	•	•
ReadCurrentErrorMsg_C		•
ReadCurrentEmergMsg_C		•
ReadCurrentAnomalyMsg_C		•
GetGCode_C	•	•
GetMCode_C	•	•
SkipPProgBlock_C	•	•
Ese_C	•	•
Ese_C		•
AxesRef_C	•	•

Functions for the "Through Mode"

	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
DncInit_C	•					
DncData_C	•					
DncEof_C	•					
DncStop_C	•					

Generic functions

	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
GetAxesInfo3_C	•		•			
GetCodeNumber_C	•	•				
GetOptions_C	•	•	•	•		
GetDateTime_C	•	•	•			
SetDateTime_C	•	•	•			
GRead_C	•	•				
GWrite_C	•	•				
SetIpAddress_C			•			
GetSerialNumber_C			•			
ReadRemapDefinitions_C			•			
WriteRemapDefinitions_C			•			
GetServoPar_C			•			
SetServoPar_C			•			

File system management functions

	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
LogFSTransferFile_C	•	•	•	•	•	•
LogFSTransferFileW_C	•	•	•	•	•	•
LogFSSetSecurityLevel_C	•	•	•	•	•	•
LogFSGetSecurityLevel_C	•	•	•	•	•	•
LogFSLongFileNames_C	•	•	•	•	•	•
LogFSGetNumDrive_C	•	•	•	•	•	•
LogFSGetDriveList_C	•	•	•	•	•	•
LogFSGetHiddenDriveList_C	•	•	•	•	•	•
LogFSGetDrivePath_C	•	•	•	•	•	•
LogFSAddDrive_C	•	•	•	•	•	•



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSRemoveDrive_C	•	•	•	•	•	•
LogFSReloadDriveList_C	•	•	•	•	•	•
LogFSCreateDir_C	•	•	•	•	•	•
LogFSCreateFile_C	•	•	•	•	•	•
LogFSGetFileSize_C	•	•	•	•	•	•
LogFSGetFileAttrib_C	•	•	•	•	•	•
LogFSSetFileAttrib_C	•	•	•	•	•	•
LogFSChangeFileAttrib_C	•	•	•	•	•	•
LogFSFindFirst_C	•	•	•	•	•	•
LogFSFindNext_C	•	•	•	•	•	•
LogFSFindClose_C	•	•	•	•	•	•
LogFSRemoveFile_C	•	•	•	•	•	•
LogFSRemoveDir_C	•	•	•	•	•	•
LogFSRename_C	•	•	•	•	•	•
LogFSCopyFile_C	•	•	•	•	•	•
LogFSGetInfo_C	•	•	•	•	•	•

Functions for long file names management

	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
PPInsertName_C	•					
PPDeleteName_C	•					
PPGetPLogicalName_C	•					
PPGetPhysicalName_C	•					
PPGetLogicalDir_C	•					
PPUpdate_C	•					

Functions for management of CANOPEN HILSCHER

	<i>S10 MC</i>	<i>S10 GP</i>	<i>OPENControl</i>	<i>SyncMotion</i>	<i>Top5</i>	<i>PC</i>
CANInit_C	•					
CANBoard_C	•					
CANSync_C	•					
CANNMT_C	•					
CANReadSDO_C	•					
CANWriteSDO_C	•					
CANGetEmergency_C	•					
CANConsoleCfg_C	•					



CndexLinkUser DLL documentation Interface with Cndex Server

Management of the Cndex server and linking up with the CNC

These functions are used to create and release instances of the Cndex server and to open communication sessions with the CNC.



CndexLinkUser DLL documentation Interface with Cndex Server

ConnectServer_C (obsolete function)

Obsolete function. It does nothing. It is present in this library to make it compatible with previous versions. See function *OpenSession_C*.

Creates an instance of the Cndex server either locally, where the application resides, or on a remote PC. To use the server on a remote PC, make sure that the TCP/IP protocol is included among the network protocols used by both PC's (local and remote).

This function must be called only once, when a client application is launched, or when activating a thread.

```
WORD ConnectServer_C (  
    LPSTR      ServerAddress,  
    DWORD      *pErrClass,  
    DWORD      *pErrNum  
);
```

Parameters

ServerAddress

[in] Pointer to a string that contains the network name or the TCP/IP address of the remote PC on which the Cndex server must be instanced. Specify a pointer to an empty string if you want the instance to be created on the local PC.

pErrClass

[out] Pointer to the variable where the class of any error that may occur will be written.

pErrNum

[out] Pointer to the variable where the number of any error that may occur will be written.

Value returned

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

ReleaseServer_C, OpenSession_C, CloseSession_C



CndexLinkUser DLL documentation Interface with Cndex Server

ReleaseServer_C (obsolete function)

Obsolete function. It does nothing. It is present in this library to make it compatible with previous versions. See function CloseSession_C.

Releases the instance of the Cndex server created previously with the ConnectServer_C function. This function must be called only once when the client application is closed, or at the end of a thread, after closing all the communication sessions (see OpnSession_C and Close Session_C).

```
WORD ReleaseServer_C (  
    DWORD *pErrClass,  
    DWORD *pErrNum  
);
```

Parameters

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

ConnectServer_C, OpnSession_C, CloseSession_C



CndexLinkUser DLL documentation Interface with Cndex Server

OpenSession_C

Creates an instance of the Cndex server and opens a communication session with the numerical control. If the connection attempt succeeds, a session identification (UserSession) to be used in all subsequent calls is returned (see notes to use the session in a multi-thread environment).

To access the 10/Series numerical controls, NETBEUI protocol must be present among the network protocols used by (local or remote) PC in which the Cndex server instance is created.

```
WORD OpenSession_C (  
    LPSTR    RemoteName,  
    WORD     *pUserSession,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

RemoteName

[in] String containing the name of the numerical control that you want to connect to. It must be a string with a maximum length of 16 characters. You can use uppercase or lowercase letters indiscriminately.

To address a 10/Series CNC you must specify the CNC network name in RemoteName.

For all other OSAI systems it is mandatory to place the "**IP.**" prefix before the system name or TCP/IP network address of the system to be connected.

I.e. to connect an OPENControl CNC configured with system name "OPENControlCNC" the RemoteName parameter must contain the text "**IP.OPENControlCNC**".

To connect a 10/Series CNC using a PC as a network server, it must be specified the "IP:" prefix followed by the system name or TCP/IP address followed by slash character "/" followed by the 10/Series CNC network name

I.e. to connect a 10/Series CNC configured with network name "CNC_S10" through a PC having network address 192.168.1.36 it must be written in RemoteName parameter the text "**IP.192.168.1.36\CNC_S10**".

The use of the "**IP.**" prefix only indicates the local machine on OPENControl and SyncMotion systems ("Windows CE" applications).

pUserSession

[in] Pointer to the variable in which the identifier of the communication session with the specified numerical control will be returned. The value of the WORD must be used as an input to other functions to refer to the CNC connected.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

Note

The communication sessions opened with this function can be used, normally, only by the thread that called the function. In other words, the identifier of the session opened for instance by the main process of an application, can be used in any point of that process but can not be passed to a "thread".

This is due to the modality to initialise the "DCOM threading model" that is the model set for the management of the thread for the DCOM servers. This model can be set only once in an application and can not be changed.

The Win32 (PC) applications initialise OLE (and consequently DCOM) as single-thread.

I.e. the MFC C++ application initialize DCOM through the OLE initialization function (see function "AfxOleInit" in the class derived from "CwinApp").

The C# application developed for Windows CE initialize the DCOM threading model as multi-thread and the single-thread model is not available.

Basically the `OpenSession_C` function initializes the DCOM threading model as single-thread only in case the model has not been defined yet. In all other cases it uses the model imposed by the caller.

If the application must connect the same CNC in several threads there are two solutions:

- If the application allows it, initialize the "DCOM threading model" as "multi threaded" and uses the communication session opened through the `OpenSession_C` function, in all threads.
- If the application must initialize either OLE or the "DCOM threading model" as "apartment (single) threaded", open several independent communication sessions calling the `OpenSession_C` function in all threads that must connect the CNC.

See also

`CloseSession_C`



CndexLinkUser DLL documentation Interface with Cndex Server

CloseSession_C

Interrupts the connection with a numerical control by closing the session and releases the Cndex server instance created with the OpenSession_C function.

Once you exit this function, the session identifier is no longer valid and must not be reused.

```
WORD CloseSession_C (  
    WORD    UserSession,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

See also

OpenSession_C

Functions for the control of the CNC bootstrap phase

During the acquisition, the CNC goes through various "phases" that can be controlled by means of the **BootPhaseEnquiry_C** function.

The initial phases (phase 1 and phase 2) are executed in a boot environment that activates the OSAI Operating System multitasking and all the tasks making up the CNC. When the Operating System is activated, it destroys the boot system that has been used for its activation.

The boot environment that executes phases 1 and 2 predisposes the CNC for the connection to the network. Thanks to this mechanism, it is possible to open a communication session with the CNC when the boot environment is activated, but the connection fails when the Operating System is activated.

When the O.S. system has been activated, it will be possible again to connect the CNC, now in a definitive manner.

When the boot environment is active, it can be governed in operating modes dedicated to software configuration and update processes.



CndexLinkUser DLL documentation Interface with Cndex Server

A client application must foresee this situation and wait for the RUN mode (phase 4) before executing the request for and transmission of data to the CNC.

The possible phases of activation of the CNC are (the phase number corresponds to the number returned in the pPhase parameter of the **BootPhaseEnquiry** function) :

- 1 = EMERG_SWITCH_PHASE
- 2 = HW_BOOT_PHASE
- 3 = SW_BOOT_PHASE
- 4 = SYSTEM_UP_PHASE
- 5 = SERVER_MODE_PHASE
- 6 = REMOTE_SETUP_PHASE
- 7 = SERVICE_MODE_PHASE
- 8 = AX_PARAM_VERIFY_PHASE
- 15 = PRIMARY_INIT_PHASE
- 16 = NOT_INIT_PHASE
- 17 = SHUTDOWN_PHASE
- 19 = ERROR_PHASE

- Phase 1** The CNC is waiting for a command to guide it in a subsequent status.
This is a phase of transition. If no client has connected after a certain period of time, the CNC carries on the standard BOOT procedure to pass into RUN status.
- Phase 2** The CNC is executing hardware diagnostic and configuration programs. This is a phase of transition.
- Phase 3** The CNC is loading the CNC software to pass to RUN mode (Phase 4). This is a phase of transition.
- Phase 4** The CNC is in RUN status and the machining process can be started. Client applications can call the data exchange functions. This is the stable piece machining phase; to go to a different operating mode, the CNC must perform the bootstrap afresh.
- Phase 5** The CNC is in EMERGENCY status. This is a "stable" phase; to go to a different operating mode (e.g., RUN), the CNC must perform the bootstrap afresh.
- Phase 6** The CNC is in SETUP status. This is a "stable" phase; to go to a different operating mode (e.g., RUN), the CNC must perform the bootstrap afresh.
- Phase 7** The CNC is in SERVICE status. This is a "stable" phase; to go to a different operating mode (e.g., RUN), the CNC must perform the bootstrap afresh.
- Phase 8** The CNC is in PARAMETER CHECKING status. It is not a stable phase. With the prior enable signal by the CNC user, the CNC will go to Phase 4.
- Phase 15** The CNC is booting but has not reached phase 1 yet.
- Phase 16** The CNC is not initialized.
- Phase 17** The CNC is rebooting.
- Phase 19** The CNC boot has been aborted due to an unrecoverable error.



CndexLinkUser DLL documentation Interface with Cndex Server

BootPhaseEnquiry_C

Requires the numerical control activation Phase.

```
WORD BootPhaseEnquiry_C (  
    WORD    UserSession,  
    WORD    *pPhase,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pPhase

[out] Pointer to a variable where the CNC boot phase will be written (See the description at the beginning of the paragraph).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

BootReboot_C

Description at the beginning of the paragraph.



CndexLinkUser DLL documentation Interface with Cndex Server

BootReboot_C

Causes the hardware reset of the numerical control, which is followed by a reboot. The activation of this function corresponds to the process of deactivation and reactivation of the CNC.

This command is available in all activation stages.

The OPENControl CNC execute a software reboot of the CNC section only (that is excluding Windows CE operating system) unless it is in the SETUP boot modality. In this case also this type of CNC performs an hardware reboot.

```
WORD BootReboot_C (  
    WORD    UserSession,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

BootShutDown_C, BootPhaseEnquiry_C

Description at the beginning of the paragraph.



CndexLinkUser DLL documentation Interface with Cndex Server

BootShutDown_C

Requests the CNC to execute the shutdown procedure (CNC closes all files and then switches off).

```
WORD BootShutDown_C (  
    WORD    UserSession,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

BootReboot_C

Description at the beginning of the paragraph.



CndexLinkUser DLL documentation Interface with Cndex Server

BootMode_C

Lets you select the CNC operating mode during the activation phase of the CNC. You can request the EMERGENCY, RUN (normal activation), SETUP and SERVICE operating modes.

For 10/Series CNC this command is available only when the CNC boot phase is EMERG_SWITCH_PHASE (see BootPhaseEnquiry_C). In all the other activation phases, the command is not valid and may cause CNC failures.

For OPENControl CNC the command is always available and must be sent before issuing the CNC reboot command through the BootReboot_C function.

```
WORD BootMode_C (  
    WORD    UserSession,  
    BootMode Mode,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Mode

[in] CNC activation mode requested, which can be:

BOOT_EMERGENCY_MODE: The CNC is requested to continue the activation phase and enter EMERGENCY mode.

BOOT_RUN_MODE: The CNC is requested to continue the activation phase and enter RUN mode (normal operating mode).

BOOT_SETUP_MODE: The CNC is requested to continue the activation phase and enter SETUP mode (software installation mode).

BOOT_SERVICE_MODE: The CNC is requested to continue the activation phase and enter SERVICE mode (OS-Wire drive tuning mode).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

BootReboot_C, BootPhaseEnquiry_C

Description at the beginning of the paragraph.



CndexLinkUser DLL documentation Interface with Cndex Server

BootEnterServer_C (10 Series only)

Warning: this function is for expert users and can only be used under express authorization by Prima Electro. Improper use may cause data loss and corruption of the CNC file system.

This function can only be used with 10 Series systems in SETUP mode. A 10 Series CNC booting in SETUP mode automatically loads the remote connection server after bootstrap end. The remote connection manager handles remote procedure calls from client applications, including Cndex function calls.

BootEnterServer_C forces the CNC to close the remote connection server and open the file server, allowing a remote system to connect to the PRJ, USR and UPP shared directories. The SYS shared directory is not available in SETUP mode. Most Cndex function calls cannot be used while the file server is active.

The remote connection server can be reactivated by calling BootLeaveServer_C. Doing this will close the file server and may cause data loss if done while a remote access to a shared directory is in progress.

```
WORD BootEnterServer_C (  
    WORD    UserSession,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

BootLeaveServer_C



CndexLinkUser DLL documentation Interface with Cndex Server

BootLeaveServer_C (10 Series only)

Warning: this function is for expert users and can only be used under express authorization by Prima Electro. Improper use may cause data loss and corruption of the CNC file system.

This function can only be used with 10 Series systems in SETUP mode. Calling this functions closes the CNC file server and reactivates the remote connection server, allowing the execution of Cndex function calls.

```
WORD BootLeaveServer_C (  
    WORD    UserSession,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

BootEnterServer_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetHWKey_C

Reads the Product Authorization Key (PAK), the security mask and the software release string from the CNC.

```
WORD GetHWKey_C (  
    WORD    UserSession,  
    BYTE    *pHwKey,  
    BYTE    *pSecurLevel,  
    BYTE    *pRelease,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pHwKey

[out] Pointer to a 32 byte long buffer where the PAK will be written.

pSecurLevel

[out] Pointer to a 12 byte long buffer where the security authorization mask for the active security level on the CNC will be written.

pRelease

[out] Pointer to an 11 byte long buffer where the CNC software release string will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

Variable management

The functions belonging to this group allow to read and write most of the CNC variables. These functions can be called only after having opened a communication session (see `OpenSession_C`) and having made sure that the CNC is in RUN mode (see `BootPhaseEnquiry_C`).



CndexLinkUser DLL documentation Interface with Cndex Server

ReadVarWord_C

Reads the system variables that have WORD format (16 bits).

```
WORD ReadVarWord_C (  
    WORD      UserSession,  
    WORD      Code,  
    WORD      Process,  
    WORD      Index,  
    WORD      NumVar,  
    WORD      *pValue  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. The following table shows on the first column (symbol) and second column (value) the available values for the parameters. The other columns shown, for any type of variable, the maximum number of element available for any OSAI system. The (R) tag identifies retained variable types.

<i>symbol</i>	<i>value</i>	<i>10/Series MC</i>	<i>OPENControl</i>	<i>PrimaLogic</i>	<i>SyncMotion</i>
I_CODE	0	512	512 (up to V3.3.1) 4096 (from V3.3.2)	512	0
O_CODE	1	512	512 (up to V3.3.1) 4096 (from V3.3.2)	512	0
MW_CODE	20	10000	20000	20000	0
GW_CODE (R)	21	256	2000	2000	0
SW_CODE	22	20 + 20 * number of processes	20 + 20 * number of processes	20	0
PW_CODE	62	0	6000 (up to V3.3.1) 20000 (from V3.3.2)	0	0
UW_CODE	63	0	0	0	0
SYMO_W_CODE	100	0	20000	0	8192
SYMORET_W_CODE (R)	103	0	2000	0	8192

Process

[in] Identifier (a number from 1 to 20 for 10/Serie MC or from 1 to 24 for OPENControl) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.
For all other type of variable it can be zero.

Index

[in] Index of the first variable to read.

NumVar



CndexLinkUser DLL documentation

Interface with Cndex Server

[in] Number of variables to read. This number added to Index must not exceed the maximum number of variables admitted for the type (Code) of selected variable.

pValue

[out] Pointer to an array of WORD (16 bits) variables that will contain the values of the read variables. The array must consist of a number of elements at least equal to NumVar.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

ReadVarDouble_C

Reads the system variables that have double (floating point 64 bits) format.

```
WORD ReadVarDouble_C (  
    WORD    UserSession,  
    WORD    Code,  
    WORD    Process,  
    WORD    Index,  
    WORD    NumVar,  
    Double  *pValue  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. The possible values are shown in the first column (define) and second column (value) of the following table. The other columns shown, for any type of variable, the maximum number of element available for any OSAI system. . The (R) tag identifies retained variable types.

<i>define</i>	<i>value</i>	<i>10/Series MC</i>	<i>OPENControl</i>	<i>PrimaLogic</i>	<i>SyncMotion</i>
MD_CODE	40	3000	10000	10000	0
GD_CODE (R)	41	64	1000	1000	0
SD_CODE	42	0	1000	20	0
PD_CODE	43	0	6000 (up to V3.3.1) 20000 (from V3.3.2)	0	0
UD_CODE	44	0	0	0	0
L_CODE (R)	145	0	3000	0	0
E_CODE	46	From 0 to 8000 (configured in AMP)	From 0 to 8000 (configured in AMP)	0	0
SN_CODE	47	25	25	0	0
H_CODE	48	100	128	0	0
SYMO_D_CODE	101	0	10000	0	6143
SYMORET_D_CODE (R)	104	0	3000	0	5117

Process

[in] Identifier (a number from 1 to 20 for 10/Serie MC or from 1 to 24 for OPENControl) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.
For all other type of variable it can be zero.

Index

[in] Index of the first variable to read.

NumVar



CndexLinkUser DLL documentation

Interface with Cndex Server

[in] Number of variables to read. This number added to Index must not exceed the maximum number of variables admitted for the type (Code) of selected variable.

pValue

[out] Pointer to an array of double variables that will contain the values of the read variables. The array must consist of a number of elements at least equal to NumVar.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

WriteVarWord_C

Writes the system variables that have WORD format (16 bits).

```
WORD WriteVarWord_C (  
    WORD    UserSession,  
    WORD    Code,  
    WORD    Process,  
    WORD    Index,  
    WORD    NumVar,  
    WORD    *pValue  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. For the possible values see ReadVarWord_C function.

Process

[in] Identifier (a number from 1 to 20 for 10/Serie MC or from 1 to 24 for OPENControl) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.

For all other type of variable it can be zero.

Index

[in] Index of the first variable to write.

NumVar

[in] Number of variables to write. This number added to Index must not exceed the maximum number of variables admitted for the type (Code) of selected variable.

pValue

[in] Pointer to an array of WORD (16 bit) variables. The array must consist of a number of elements at least equal to NumVar and must contain the values to assign to the variables.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

WriteVarWordBit_C

Writes a single bit of a system variables that has WORD format (16 bit).

```
WORD WriteVarWordBit_C (  
    WORD    UserSession,  
    WORD    Code,  
    WORD    Process,  
    WORD    Index,  
    WORD    BitIndex,  
    WORD    BitValue,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. For the possible values see the ReadVarWord_C function.

Process

[in] Identifier (a number from 1 to 20 for 10/Serie MC or from 1 to 24 for OPENControl) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.

For all other type of variable it can be zero.

Index

[in] Index of the variable to write.

BitIndex

[in] Index, inside the variable, of the bit to write.

BitValue

[in] Value to assign to the bit, zero or one.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

WriteVarDouble_C

Writes the system variables that have double (floating point 64 bits) format.

```
WORD WriteVarDouble_C (  
    WORD        UserSession,  
    WORD        Code,  
    WORD        Process,  
    WORD        Index,  
    WORD        NumVar,  
    double *    pValue  
    DWORD *    pErrClass,  
    DWORD *    pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. For the possible values see the ReadVarDouble_C function.

Process

[in] Identifier (a number from 1 to 20) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.
For all other type of variable it can be zero.

Index

[in] Index of the first variable to write.

NumVar

[in] Number of variables to write. This number added to Index must not exceed the maximum number of variables admitted for the type (Code) of selected variable.

pValue

[in] Pointer to an array of double variables. The array must consist of a number of elements at least equal to NumVar and must contain the values to assign to the variables.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

ReadVarText_C

Reads the ASCII (text) variables of the system.

```
WORD ReadVarText_C (  
    WORD        UserSession,  
    WORD        Code,  
    WORD        Process,  
    WORD        Index,  
    WORD        Size,  
    BYTE *      Text  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. The possible values are shown in the first column (define) and second column (value) of the following table. The other columns show, for any type of variable, the maximum number of element available for any OSAI system x the length of the string.

<i>define</i>	<i>value</i>	<i>10/Series MC</i>	<i>OPENControl</i>	<i>PrimaLogic</i>	<i>SyncMotion</i>
A_CODE	45	WinPLUS: 1500 x 40 chr PLUS: not available	1500 x 40 chr	250 x 128 chr	not available
AA_CODE	28	not available	250 x 128 chr	250 x 128 chr	not available
SC_CODE	50	100	100	not available	not available
SYMO_A_CODE	102	not available	64 x 128 chr	not available	64 x 128 chr
LS_CODE	18	not available	250 x 128 chr	250 x 128 chr	not available

Process

[in] Identifier (a number from 1 to 20) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.
For all other type of variable it can be zero.

Index

[in] Index of the first variable to read.

Size

[in] Size of the buffer where the data will be written (see Text parameter).

Text

[out] Pointer to an array of BYTE variables that will contain the text read from variables. The array must consist of a number of elements at least equal to Size and should be able to contain the number of characters (typically 128) expected for the selected variable type.



CndexLinkUser DLL documentation

Interface with Cndex Server

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

WriteVarText_C

Writes the ASCII (text) variables of the system.

```
WORD WriteVarText_C (  
    WORD        UserSession,  
    WORD        Code,  
    WORD        Process,  
    WORD        Index,  
    WORD        Len,  
    BYTE *      Text,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Code

[in] Identifier of the variable. For the possible values see the ReadVarText_C function.

Process

[in] Identifier (a number from 1 to 20) of the process, owner of the variable.

It is taken in account only when the variable specified with Code parameter belong to a process.
For all other type of variable it can be zero.

Index

[in] Index of the first variable to write.

Len

[in] Size of the buffer to write (see Text parameter).

Text

[in] Pointer to an array of BYTE variables that contain the text to write to the variable. The array must consist of a number of elements at least equal to Len.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

PLC oriented functions

These functions are used to exchange data with the PLC and to manage the Axes/Tools/Offset/user tables by synchronising access to them by means of semaphores.

These functions can be called only after creating the Cndex server (see ConnectServer_C), having opened a communication session (see OpenSession_C) and having made sure that the CNC is in RUN mode (see BootPhaseEnquiry_C).

WARNING:

- *It is possible to use the function of this paragraph to read and write the variables but they are obsolete. You can use instead the new series of functions described in the “Variables management” paragraph.*
- *The functions used for the 10/Series tables (Tools, Offset, Origins and User) are different from the ones for OPENControl (see the compatibility tables at the beginning of this document).*



CndexLinkUser DLL documentation Interface with Cndex Server

ReadWarningMsg_C

Reads the message that the machine logic wants to display.

```
WORD ReadWarningMsg_C (  
    WORD    UserSession,  
    LPSTR   pWarningMsg,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pWarningMsg

[out] Pointer to a 130-character string in which the message from the PLC ending in terminator 0 (zero) will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetPLVarWord_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarWord_C function.

Reads the value of the short variables of the PLUS (MW, GW etc.)

```
WORD GetPLVarWord_C (  
    WORD        UserSession,  
    WORD        NumVar,  
    PLVARDESC_C4 *pVarDesc,  
    short       *pValue,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose values have to be read

pVarDesc

[in] Pointer to an array of type PLVARDESC_C4 structures specifying the variables whose values have to be obtained

pValue

[out] Pointer to an array of short type that will contain the values of the variables specified by pVarDesc

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1026 The number of variables NumVar is out of range.

Class 4 – Error 1027 One of the identification codes of the variables specified in the structure is wrong.

Class 4 – Error 1032 One of the indices of the variables specified in the structure is out of range.

Class 4 – Error 1033 One of the masks of the variables specified in the structure is out of range.

See also

SetPLVarWord_C

Paragraph "Description of the Structures" for the definition of PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

SetPLVarWord_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use WriteVarWord_C function.

Writes the value of the word and Boolean variables of the PLUS (MW, GW, M, G etc.)

```
WORD SetPLVarWord_C (  
    WORD        UserSession,  
    WORD        NumVar,  
    PLVARDESC_C4 *pVarDesc,  
    short       *pValue,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be written

pVarDesc

[in] Pointer to an array of structures type PLVARDESC_C4 that specifies the variables to be modified

pValue

[in] Pointer to an array of short type that contains the values to be set in the variables described by pVarDesc

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1026 The number of variables NumVar is out of range.

Class 4 – Error 1027 One of the identification codes of the variables specified in the structure is wrong.

Class 4 – Error 1032 One of the indices of the variables specified in the structure is out of range.

Class 4 – Error 1033 One of the masks of the variables specified in the structure is out of range.

See also

GetPLVarWord_C

Paragraph "Description of the structures" for the definition of PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

GetPLVarDouble_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use *ReadVarDouble_C* function.

Reads the value of the double variables of the PLUS (MD, GD etc.)

```
WORD GetPLVarDouble_C (  
    WORD        UserSession,  
    WORD        NumVar,  
    PLVARDESC_C4 *pVarDesc,  
    double      *pValue,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be read

pVarDesc

[in] Pointer to an array of structures type PLVARDESC_C4 specifying the variables whose values have to be obtained

pValue

[out] Pointer to an array of double type that will contain the values of the variables specified by pVarDesc

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1026 The number of variables NumVar is out of range.

Class 4 – Error 1027 One of the identification codes of the variables specified in the structure is wrong.

Class 4 – Error 1032 One of the indices of the variables specified in the structure is out of range.

Class 4 – Error 1033 One of the masks of the variables specified in the structure is out of range.

See also

SetPLVarDouble_C

Paragraph "Description of the structures" for the definition of PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

SetPLVarDouble_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use WriteVarDouble_C function.

Writes the value of the double variables of the PLUS (MD, GD etc.)

```
WORD SetPLVarDouble_C (  
    WORD        UserSession,  
    WORD        NumVar,  
    PLVARDESC_C4 *pVarDesc,  
    double      *pValue,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be written

pVarDesc

[in] Pointer to an array of structures type PLVARDESC_C4 specifying the variables to be modified

pValue

[in] Pointer to an array of double type containing the values to be set in the variables described in pVarDesc

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1026 The number of variables NumVar is out of range.

Class 4 – Error 1027 One of the identification codes of the variables specified in the structure is wrong.

Class 4 – Error 1032 One of the indices of the variables specified in the structure is out of range.

Class 4 – Error 1033 One of the masks of the variables specified in the structure is out of range.

See also

GetPLVarDouble_C

Paragraph "Description of the structures" for the definition of PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

GetPLVarAscii_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use *ReadVarText_C* function.

Reads the array of ASCII variables of the WinPLUS PLC.

This function is available only if the machine logic has been written with WinPLUS and the variable is definite.

```
WORD GetPLVarAscii_C (  
    WORD    UserSession,  
    WORD    Index,  
    LPSTR    pValue,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Index

[in] Starting index for reading the ASCII variables

pValue

[in] Pointer to a 41-character string in which the text contained in the ASCII variables will be written. The text ends with the string terminator /0 (zero).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

SetPLVarAscii_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetPLVarAscii_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use WriteVarText_C function.

Writes in the array of ASCII variables of the WinPLUS PLC.

This function is available only if the machine logic has been written with WinPLUS and the variable is definite.

```
WORD SetPLVarAscii_C (  
    WORD    UserSession,  
    WORD    Index,  
    LPSTR   pValue,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Index

[in] Starting index for writing the ASCII variables

pValue

[in] Pointer to a 41-character string, ending with /0 (zero), which contains the text to be written in the ASCII variables. The text ends with string terminator /0 (zero).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

GetPLVarAscii_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetPLStreamWord_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarWord_C function.

Reads arrays of short type sequential variables of the PLC (MW, GW etc.)

```
WORD GetPLStreamWord_C (  
    WORD      UserSession,  
    WORD      NumVar,  
    WORD      Code,  
    WORD      StartIndex,  
    short     *pValue,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be read

Code

[in] Type of variables to be read. See Code field of structure PLVARDESC_C4 in paragraph "Definitions of the Structures and Definitions".

StartIndex

[in] Starting index for reading in the array.

pValue

[out] Pointer to an array of short type containing the values of the variables requested. The array must be dimensioned to contain all the variables requested (at least NumVar elements).

pErrClass

[out] Pointer to the variable where the class of an error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of an error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number) see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1027 Code identifying an array of variables either wrong or not available on the system connected.

Class 4 – Error 1032 Index out of range (StartIndex + NumVar).

See also

SetPLVarWord_C



CndexLinkUser DLL documentation Interface with Cndex Server

Paragraph "Description of the Structures" for the definition of the Code field of structure PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

SetPLStreamWord_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use WriteVarWord_C function.

Writes arrays of short type sequential variables of the PLC (MW, GW etc.)

```
WORD SetPLStreamWord_C (  
    WORD      UserSession,  
    WORD      NumVar,  
    WORD      Code,  
    WORD      StartIndex,  
    short     *pValue,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be written

Code

[in] Type of variables to be written. See Code field of structure PLVARDESC_C4 in paragraph "Definitions of the Structures and Definitions".

StartIndex

[in] Starting index for writing in the array.

pValue

[out] Pointer to an array of short type containing the values to be written in the variables. The array must be dimensioned to contain all values to be written (at least NumVar elements).

pErrClass

[out] Pointer to the variable where the class of an error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of an error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number) see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1027 Code identifying an array of variables either wrong or not available on the system connected.

Class 4 – Error 1032 Index out of range (StartIndex + NumVar).

See also

SetPLVarWord_C



CndexLinkUser DLL documentation Interface with Cndex Server

Paragraph "Description of the Structures" for the definition of the Code field of structure PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

GetPLStreamDouble_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarDouble_C function.

Reads arrays of double type sequential variables of the PLC (MD, GD etc.)

```
WORD GetPLStreamWord_C (  
    WORD    UserSession,  
    WORD    NumVar,  
    WORD    Code,  
    WORD    StartIndex,  
    double  *pValue,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be read

Code

[in] Types of variable to be read. See Code field of structure PLVARDESC_C4 in paragraph "Definition of Structures and Definitions".

StartIndex

[in] Starting index for reading in the array.

pValue

[out] Pointer to an array of double type [vv] to contain the values of the variables requested. The array must be dimensioned to contain all the variables requested (at least NumVar elements).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been executed successfully, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1027 Code identifying an array of variables either wrong or not available on the system connected.

Class 4 – Error 1032 Index out of range (StartIndex + NumVar).



CndexLinkUser DLL documentation Interface with Cndex Server

See also

SetPLVarDouble_C

Paragraph "Description of the Structures" for the definition of structure PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

SetPLStreamDouble_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use WriteVarDouble_C function.

Writes arrays of double type sequential variables of the PLC (MD, GD etc.)

```
WORD SetPLStreamWord_C (  
    WORD      UserSession,  
    WORD      NumVar,  
    WORD      Code,  
    WORD      StartIndex,  
    double    *pValue,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

NumVar

[in] Number of variables whose value has to be written

Code

[in] Types of variable to be written. See Code field of structure PLVARDESC_C4 in paragraph "Definition of Structures and Definitions".

StartIndex

[in] Starting index for writing in the array.

pValue

[out] Pointer to an array of double type containing the values to be written in the variables. The array must be dimensioned to contain all values to be written (at least NumVar elements).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been executed successfully, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1027 Code identifying an array of variables either wrong or not available on the system connected.

Class 4 – Error 1032 Index out of range (StartIndex + NumVar).



CndexLinkUser DLL documentation Interface with Cndex Server

See also

SetPLVarDouble_C

Paragraph "Description of the Structures" for the definition of structure PLVARDESC_C4



CndexLinkUser DLL documentation Interface with Cndex Server

GetAxisTabRecord_C

Reads a record in the system axes table (origins).

```
WORD GetAxisTabRecord_C (  
    WORD      UserSession,  
    WORD      RecordNum,  
    AXIS_TABLE_C4 *pRecordBuff ,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to an AXIS_TABLE_C4 type structure where the record requested will be written

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

SetAxisTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetAxisTabRecord_C

Writes a record in the system axes table (origins).

```
WORD SetAxisTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    AXIS_TABLE_C4 *pRecordBuff,  
    DWORD        *pErrClass,  
    DWORD        *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[in] Pointer to an AXIS_TABLE_C4 type structure which contains the data to be written in the record

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

GetAxisTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetToolTabRecord_C

Reads a record in the system tools table.

```
WORD GetToolTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    TOOL_TABLE_C4 *pRecordBuff ,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to a TOOL_TABLE_C4 type structure where the record requested will be written

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

SetToolTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetToolTabRecord_C

Writes a record in the system tools table.

```
WORD SetToolTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    TOOL_TABLE_C4 *pRecordBuff,  
    DWORD        *pErrClass,  
    DWORD        *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[in] Pointer to the TOOL_TABLE_C4 type structure containing the data to be written in the record

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

GetAxisTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetOffsetTabRecord_C

Reads a record in the system offsets table.

```
WORD GetOffsetTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    OFFSET_TABLE_C4 *pRecordBuff,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to an OFFSET_TABLE_C4 type structure where the record requested will be written

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

SetOffsetRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetOffsetTabRecord_C

Writes a record in the system offset table.

```
WORD SetOffsetTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    OFFSET_TABLE_C4 *pRecordBuff,  
    DWORD        *pErrClass,  
    DWORD        *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[in] Pointer to an OFFSET_TABLE_C4 type structure containing the data to be written in the record

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

GetAxisTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetUserTabRecord_C

Reads a record in the system User table.

```
WORD GetUserTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    USER_TABLE_C4 *pRecordBuff,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to a USER_TABLE_C4 type structure where the record requested will be written

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

SetUserTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetUserTabRecord_C

Writes a record in the system User table.

```
WORD SetUserTabRecord_C (  
    WORD        UserSession,  
    WORD        RecordNum,  
    USER_TABLE_C4 *pRecordBuff,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[in] Pointer to a USER_TABLE_C4 type structure containing the data to be written in the record

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

GetUserTabRecord_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetTabFieldDouble_C

Reads a double format field in a system table.

```
WORD GetTabFieldDouble_C (  
    WORD    UserSession,  
    WORD    TableNum,  
    WORD    RecordNum,  
    WORD    FieldNum,  
    double  *pValue,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Table identifier

RecordNum

[in] Record number

FieldNum

[in] Field identifier

pValue

[out] Pointer to the variable where the field value will be written

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 2 The record number RecordNum is out of range.

Class 4 – Error 3 The field number FieldNum is out of range.

Class 4 – Error 4 The format of the field requested is not double type.

See also

SetTabFieldDouble_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetTabFieldDouble_C

Writes a double format field in a system table.

```
WORD SetTabFieldDouble_C (  
    WORD    UserSession,  
    WORD    TableNum,  
    WORD    RecordNum,  
    WORD    FieldNum,  
    double  Value,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Table identifier

RecordNum

[in] Record number

FieldNum

[in] Field identifier

Value

[in] Value to be written in the field

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 2 The record number RecordNum is out of range.

Class 4 – Error 3 The field number FieldNum is out of range.

Class 4 – Error 4 The format of the field specified is not double type.

See also

GetTabFieldDouble_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetTabFieldShort_C

Reads a WORD format field in a system table.

```
WORD GetTabFieldShort_C (  
    WORD    UserSession,  
    WORD    TableNum,  
    WORD    RecordNum,  
    WORD    FieldNum,  
    short *  pValue  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Table identifier

RecordNum

[in] Record number

FieldNum

[in] Field identifier

pValue

[out] Pointer to the variable where the value of the field will be written

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 2 The record number RecordNum is out of range.

Class 4 – Error 3 The field number FieldNum is out of range.

Class 4 – Error 4 The format of the field specified is not short type.

See also

SetTabFieldShort_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetTabFieldShort_C

Writes a WORD format field in a system table.

```
WORD SetTabFieldShort_C (  
    WORD    UserSession,  
    WORD    TableNum,  
    WORD    RecordNum,  
    WORD    FieldNum,  
    short   Value  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Table identifier

RecordNum

[in] Record number

FieldNum

[in] Field identifier

Value

[out] Value to be written in the field

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 2 The record number RecordNum is out of range.

Class 4 – Error 3 The field number FieldNum is out of range.

Class 4 – Error 4 The format of the field specified is not short type.

See also

GetTabFieldShort_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

TabSearchDouble_C

Searches for a double format value in the system tables. To perform the search you must specify the table, the field and the value to be found. The search can be narrowed to a specific set of records.

```
WORD TabSearchDouble_C (  
    WORD    UserSession,  
    WORD    TabNum,  
    WORD    FieldNum,  
    WORD    StartIndex,  
    WORD    StopIndex,  
    double  Value,  
    WORD    *pMatchIndex,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TabNum

[in] Table identifier

FieldNum

[in] Field identifier

StartIndex

[in] Search start index

StopIndex

[in] Search stop index

Value

[in] Field value

pMatchIndex

[out] Index of the first value that coincides

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 2 The field number FieldNum is out of range.

Class 4 – Error 3 StartIndex exceeds the size of the table.



CndexLinkUser DLL documentation Interface with Cndex Server

Class 4 – Error 4 The format of the field requested is not double type.

Class 4 – Error 5 StartIndex is bigger than StopIndex.

Class 4 – Error 6 The search has reached StopIndex without finding the datum.

Class 4 – Error 7 (the search has reached the end of the table without finding the datum)

See also

TabSearchShort_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

TabSearchShort_C

Search for a value in WORD format in the system tables. To perform the search, you have to specify the table, the field and the value to be found. The search can be narrowed to a specific set of records.

```
WORD TabSearchShort_C (  
    WORD    UserSession,  
    WORD    TabNum,  
    WORD    FieldNum,  
    WORD    StartIndex,  
    WORD    StopIndex,  
    WORD    Value,  
    WORD    Mask,  
    WORD    * pMatchIndex,  
    DWORD   * pErrClass,  
    DWORD   * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TabNum

[in] Table identifier

FieldNum

[in] Field identifier

StartIndex

[in] Search start index

StopIndex

[in] Search stop index

Value

[in] Field value

Mask

[in] Mask to narrow the search to some bits of the WORD

pMatchIndex

[out] Index of the first value that coincides

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value



CndexLinkUser DLL documentation Interface with Cndex Server

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 2 The field number FieldNum is out of range.

Class 4 – Error 3 StartIndex is bigger than the table.

Class 4 – Error 4 The format of the field requested is not short type.

Class 4 – Error 5 StartIndex is bigger than StopIndex.

Class 4 – Error 6 The search has reached StopIndex without finding the datum.

Class 4 – Error 7 The search has reached the end of the table without finding the datum.

See also

TabSearchDouble_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetTabSem_C

The SetTabSem function sets the status of the semaphore protecting the tables (AXES, TOOLS, OFFSET or USER).

Every table has a semaphore protecting against simultaneous multiple accesses. The semaphore of each table does not prevent access, but indicates that the resource is already in use.

A table may be in use, for instance, when the NC operator enters or changes data through the TABLE EDITOR application.

Each application must make sure it can access the tables through this function and must not change the data if the tables are already in use.

Access to a table should be according to the following procedure:

1. Set the semaphore on RED to tell the other applications that you are about to access the table. The function gives error if resource is in use; in this case, do not access the table and adopt an appropriate management policy (retry or notify the condition to the user, etc.)
2. Writing the table data
3. Set the semaphore on GREEN to indicate that access to the table is over.

```
WORD SetTabSem_C (  
    WORD    UserSession,  
    WORD    TableNum,  
    WORD    SemStatus,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Table identifier

SemStatus

[in] Semaphore status:

1 = RED

2 = GREEN

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1 SemStatus value is invalid.



CndexLinkUser DLL documentation Interface with Cndex Server

Class 4 – Error 2 The table identifier TableNum is out of range.

Class 4 – Error 3 Semaphore status is already RED.

Class 4 – Error 4 Semaphore status is already GREEN.

See also

TabSemInfo_C



CndexLinkUser DLL documentation Interface with Cndex Server

TabSemInfo_C

The TabSemInfo function gives the status of the semaphores protecting the tables (AXES, TOOLS, OFFSET or USER).

```
WORD TabSemInfo_C (  
    WORD    UserSession,  
    WORD    *pSemStatus,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pSemStatus

[out] Pointer to a variable which is set with the status of the table access semaphores, bit = zero shows that the semaphore is GREEN, bit = 1 tells that the semaphore is RED

Bit 0: Axis table

Bit 1: Tool table

Bit 2: Offset table

Bit 3: User table

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

ResetSingleTableII_C

Resets the content of a table (ORIGINS, TOOLS, OFFSETS or USER).

```
WORD ResetSingleTableII_C (  
    WORD        UserSession,  
    WORD        TableNum,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Identifier of the table to reset

0 = TOOL

1 = OFFSET

2 = ORIGINS

3 = USER

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Valore di ritorno

Different from zero if the function has been executed successfully, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LockTableII_C

Set to red the state of the semaphore to protect the tables (ORIGINS, TOOLS, OFFSET or USER). See SetTabSem_C.

```
WORD LockTableII_C (  
    WORD          UserSession,  
    WORD          TableNum,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Identifier of the table (see ResetSingleTableII_C function).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

VReturn value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 25 – Error 14 The table identifier TableNum is out of range.

Class 25 – Error 17 The state of the semaphore is already RED.

See also

UnLockTable_C, SetTabSem_C.



CndexLinkUser DLL documentation Interface with Cndex Server

UnLockTableII_C

Set to green the state of the semaphore to protect the tables (ORIGINS, TOOLS, OFFSET or USER). See SetTabSem_C.

```
WORD UnLockTableII_C (  
    WORD        UserSession,  
    WORD        TableNum,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

TableNum

[in] Identifier of the table (see ResetSingleTableII_C function).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 25 – Error 14 The table identifier TableNum is out of range.

See also

LockTable_C, SetTabSem_C.



CndexLinkUser DLL documentation Interface with Cndex Server

GetOriginTabRecordII_C

Reads a system origin table record.

```
WORD GetOriginTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    ORIGIN_TABLE_II_C4 * pRecordBuff ,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be retrieved from the origin table.

pRecordBuff

[out] Pointer to a structure of type ORIGIN_TABLE_II_C4 where the requested record will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

SetOriginTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetOriginTabRecordII_C

Writes a record of the system origin table.

```
WORD SetOriginTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    ORIGIN_TABLE_II_C4 * pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be modified in the origin table.

pRecordBuff

[in] Pointer to a structure of type ORIGIN_TABLE_II_C4 that contain the data to be written in the record.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Classe 4 – Errore 4 The record number RecordNum is out of range.

See also

GetOriginTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetToolTabRecordII_C

Reads a record of the system tool table.

```
WORD GetToolTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    TOOL_TABLE_II_C4 *  pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be retrieved from the tool table.

pRecordBuff

[out] Pointer to a structure of type TOOL_TABLE_II_C4 where the requested record will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Classe 4 – Errore 4 The record number RecordNum is out of range.

See also

SetToolTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetToolTabRecordII_C

Writes a record of the system tool table.

```
WORD SetToolTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    TOOL_TABLE_II_C4 *  pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be modified in the tool table.

pRecordBuff

[in] Pointer to a structure of type TOOL_TABLE_II_C4 that contain the data to be written in the record.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Classe 4 – Errore 4 The record number RecordNum is out of range.

See also

GetToolTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetOffsetTabRecordII_C

Reads a record of the system offset table.

```
WORD GetOffsetTabRecordII_C (  
    WORD          UserSession,  
    WORD          RecordNum,  
    OFFSET_TABLE_II_C4 * pRecordBuff,  
    DWORD *       pErrClass,  
    DWORD *       pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be retrieved from the offset table.

pRecordBuff

[out] Pointer to a structure of type OFFSET_TABLE_II_C4 where the requested record will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Classe 4 – Errore 4 The record number RecordNum is out of range.

See also

SetOffsetRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetOffsetTabRecordII_C

Writes a record of the system offset table.

```
WORD SetOffsetTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    OFFSET_TABLE_II_C4 * pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be modified in the offset table.

pRecordBuff

[in] Pointer to a structure of type OFFSET_TABLE_II_C4 that contain the data to be written in the record.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Classe 4 – Errore 4 The record number RecordNum is out of range.

See also

GetOffsetTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetUserTabRecordII_C

Reads a record of the system user table.

```
WORD GetUserTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    USER_TABLE_II_C4 *  pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to a structure of type USER_TABLE_II_C4 where the requested record will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Classe 4 – Errore 4 The record number RecordNum is out of range.

See also

SetUserTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetUserTabRecordII_C

Writes a record of the system user table.

```
WORD SetUserTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    USER_TABLE_II_C4 *  pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be modified in the user table.

pRecordBuff

[in] Pointer to a structure of type USER_TABLE_II_C4 that contain the data to be written in the record.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 4 The record number RecordNum is out of range.

See also

GetUserTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetMagazineTabRecordII_C

Reads a record of the system magazine table.

```
WORD GetMagazineTabRecordII_C (  
    WORD UserSession,  
    WORD RecordNum,  
    MAGAZINE_TABLE_II_C4* pRecordBuff,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to a structure of type MAGAZINE_TABLE_II_C4 where the requested record will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Non-zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetMagazineTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetMagazineTabRecordII_C

Writes a record of the system user table.

```
WORD SetMagazineTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    MAGAZINE_TABLE_II_C4 * pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be modified in the user table.

pRecordBuff

[in] Pointer to a structure of type MAGAZINE_TABLE_II_C4 that contain the data to be written in the record.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Non-zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetMagazineTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetPocketTabRecordII_C

Reads a record of the system magazine table.

```
WORD GetPocketTabRecordII_C (  
    WORD UserSession,  
    WORD RecordNum,  
    POCKET_TABLE_II_C4 * pRecordBuff,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pRecordBuff

[out] Pointer to a structure of type POCKET_TABLE_II_C4 where the requested record will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Non-zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetPocketTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetPocketTabRecordII_C

Writes a record of the system user table.

```
WORD SetPocketTabRecordII_C (  
    WORD                UserSession,  
    WORD                RecordNum,  
    POCKET_TABLE_II_C4 * pRecordBuff,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

RecordNum

[in] 1-based index of the record to be modified in the user table.

pRecordBuff

[in] Pointer to a structure of type POCKET_TABLE_II_C4 that contain the data to be written in the record.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Non-zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetPocketTabRecordII_C, SetTabSem_C



CndexLinkUser DLL documentation Interface with Cndex Server

SaveTables_C

Save all user tables

```
WORD SaveTables_C (  
    WORD UserSession,  
    LPWSTR LocalDir,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LocalDir

[in] Local path where the table files will be saved.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 25 – Error 14	Wrong table type
Class 25 – Error 19	Error backuping table files

See also

SaveSingleTable_C, RestoreSingleTable_C



CndexLinkUser DLL documentation Interface with Cndex Server

SaveSingleTable_C

Saves the specified user table.

```
WORD SaveSingleTable_C (  
    WORD UserSession,  
    LPWSTR LocalDir,  
    TABLE_TYPE_II TableType  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LocalDir

[in] Local path where the table files will be saved.

TableType

[in] Type of the table to save.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 25 – Error 14 Wrong table type

Class 25 – Error 19 Error backuping table files

See also

SaveTables_C, RestoreSingleTable_C



CndexLinkUser DLL documentation Interface with Cndex Server

RestoreSingleTable_C

Restores the specified table.

```
WORD RestoreSingleTables_C (  
    WORD UserSession,  
    LPWSTR LocalDir,  
    TABLE_TYPE_II TableType  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LocalDir

[in] Local path where the table files to be restored have been saved.

TableType

[in] Type of the table to save.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Valore di ritorno

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 25 – Error 14	Wrong table type
Class 25 – Error 18	Error restoring table files

See also

SaveTables_C, SaveSingleTable_C



CndexLinkUser DLL documentation Interface with Cndex Server

SaveBackupMemory_C

Saves the entire dual port ram into a file located in the CNC file system.

```
WORD SaveBackupMemory_C (  
    WORD        UserSession,  
    LPWSTR      FileName,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

FileName

[in] Full path of the file where the contents of the retentive memory will be saved.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the "Error Management" chapter.

Additional error codes:

Class 25 – Error 30 Error saving Retain Memory (file)

Class 25 – Error 31 Retain Memory corrupted

Class 25 – Error 33 Memory allocation error restoring Retain Memory

See also

RestoreBackupMemory_C



CndexLinkUser DLL documentation Interface with Cndex Server

RestoreBackupMemory_C

Restores the entire dual port ram from a file located in the CNC file system.

```
WORD RestoreBackupMemory_C (  
    WORD      UserSession,  
    LPWSTR    FileName,  
    DWORD     RestoreMask,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

FileName

[in] Full path of the file from which the retentive memory will be restored. The file must be located in the CNC file system.

RestoreMask

[in] Bitmask of the dual port data areas to be overwritten. The mask can be a combination of any of the following values:

Symbol	Value	Description
RESET GW	0x00000001	GW memory reset
RESET GD	0x00000002	GD memory reset
RESET L	0x00000004	L memory reset
RESET LS	0x00000008	LS memory reset
RESET PLCM	0x00000010	PLC retain memory reset
RESET PROC	0x00000020	Process memory reset
RESET ALL	0x1000003f	Reset of the whole dual port

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 25 – Error 30	Error saving Retain Memory (file)
Class 25 – Error 31	Retain Memory corrupted
Class 25 – Error 33	Memory allocation error restoring Retain Memory
Class 25 – Error 32	Error restoring Retain Memory (file)

See also

SaveBackupMemory_C



CndexLinkUser DLL documentation Interface with Cndex Server

Process dedicated functions

These functions are used to exchange data with the CNC processes, issue commands and set the processing parameters.

These functions can be called only after creating the Cndex server (see `ConnectServer_C`), having opened a communication session (see `OpenSession_C`) and having made sure that the CNC is in RUN mode (see `BootPhaseEnquiry_C`).



CndexLinkUser DLL documentation Interface with Cndex Server

Cycle_C

Gives the process the command execution start/end command (processing cycle start, manual axis movement, MDI command execution, etc.).

```
WORD Cycle_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    Cmd  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identified (a number from 1 to 24).

Cmd

[in] Value that determines whether the command is Cycle Start or Cycle Stop:

1 : cycle start

0 : cycle stop

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Note

This command is asynchronous, the function doesn't wait neither the end of the command nor that it starts the execution. To wait for the command start please use the SyncroCycle_C function.

See also

SyncroCycle_C, Hold_C, Reset_C



CndexLinkUser DLL documentation Interface with Cndex Server

SyncroCycle_C

Gives the process the command execution start command (processing cycle start, manual axis movement, MDI command execution, etc.).

```
WORD SyncroCycle_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Note

Differently by Cycle_C function that is asynchronous, this function synchronises the start of the command. In other words it waits that the system start the execution of the command before returning the flow to the application.

See also

Cycle_C, Hold_C, Reset_C



CndexLinkUser DLL documentation Interface with Cndex Server

Reset_C

Issues a reset command to the process (the operation underway is halted and predetermined values are restored)

```
WORD Reset_C (  
    WORD    UserSession,  
    WORD    ProcNum  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

Cycle_C, Hold_C



CndexLinkUser DLL documentation Interface with Cndex Server

Hold_C

Issues a hold command to the process (the processing cycle is interrupted and resumed).

```
WORD Hold_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      Cmd  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identified (a number from 1 to 24).

Cmd

[in] Value that determines whether the command is Hold On or Hold Off:

1 : Hold On

0 : Hold Off

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 107 One of the axis is not on the profile.

See also

Cycle_C, Reset_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetFeedManOver_C

Sets the value of Feed Manual Override (percentage variation of manual movement feed) and the direction of manual movements.

```
WORD SetFeedManOver_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    FeedManOver,  
    WORD    Direction,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

FeedManOver

[in] Percentage value to be set. The number must be multiplied by 100 (10000 stands for 100%, 7500 stands for 75% etc.)

Direction

[in] Determines the direction of manual movements

0: "positive" direction

1: "negative" direction

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

SetFeedRateOver_C, SetFeedRapidOver_C, SetSpeedRateOver_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetFeedRateOver_C

Sets the value of Feed Rate Override (percentage feed variation for continuous movements).

```
WORD SetFeedRateOver_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      FeedRateOver,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

FeedRateOver

[in] Percentage value to be set. The number must be multiplied by 100 (10000 stands for 100%, 7500 stands for 75% etc.)

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

SetFeedRapidOver_C, SetFeedManOver_C, SetSpeedRateOver_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetFeedRapidOver_C

Sets the value of Feed Rapid Override (rapid feed percentage variation).
Rapid Override is active only when rapid feed control is enabled.

```
WORD SetFeedRapidOver_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    FeedRapidOver,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

FeedRapidOver

[in] Percentage value to be set. The number must be multiplied by 100 (10000 stands for 100%, 7500 stands for 75% etc.)

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetFeedRateOver_C, SetFeedManOver_C, SetSpeedRateOver_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetSpeedRateOver_C

Sets the value of Speed Override (spindle speed percentage variation).

```
WORD SetSpeedRateOver_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      SpeedRateOver,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

SpeedRateOver

[in] Percentage value to be set. The number must be multiplied by 100 (10000 stands for 100%, 7500 stands for 75% etc.)

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

SetFeedRateOver_C, SetFeedRapidOver_C, SetFeedManOver_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetManMovDirection_C

Sets the direction of manual movements. Setting the direction of movement by means of this function corresponds to the setting entered through the Direction parameter of the SetFeedManOver_C function. Unlike the latter, the SetManMovDirection_C function makes it possible to change the direction of movement without changing the value of Feed Manual Override

```
WORD SetFeedManOver_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      Direction,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Direction

[in] Determines the direction of manual movements

0: "positive" direction

1: "negative" direction

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

SetFeedManOver_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetVarJOG_C

Reads the value of the increment for the Incremental Manual Mode (incremental manual movements of the axes).

```
WORD GetVarJOG_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    double  *Value,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Value

[out] Pointer to the variable where the current JOG value will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

GetNcInfo1_C, SetVarJOG_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetVarJOG_C

Sets the value of the increment for the Incremental Manual Mode (incremental manual movements of the axes).

```
WORD SetVarJOG_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    double  Value,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Value

[in] JOG value to be set

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetVarJOG_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetVarUAS_C

Activate or de-activate the DryRun mode for the part programs test.

```
WORD SetVarUAS_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      Mode,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Mode

[in] Selettore per attivare o disattivare il DryRun. I possibili valori sono:

0 = disattivazione

1 = attivazione

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

DryRun can not be activated when Search in Memory is active on the CNC. In this case the function returns the ERR_UAS_DISABLED_WHEN_RCM_ON error code.

For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetVarRCM_C

Reads the RCM flags that indicates the Memory Search status.

```
WORD GetVarRCM_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    BYTE *    pValue,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pValue

[out] Pointer to the variable where it will be written the current value of RCM flag.
The value is 0 when RCM is inactive, 1 when it is active.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetVarRCM_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetVarRCM_C

Activates or deactivates the RCM flags that is the status of the Memory Search.

```
WORD SetVarRCM_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      Mode,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Mode

[in] Value to set. 0 deactivates the RCM, 1 activates it.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetVarRCM_C



CndexLinkUser DLL documentation Interface with Cndex Server

LoadPTech_C

Copies a set of values into the matrix variable \$PTECH .

```
WORD LoadPTech_C (  
    WORD UserSession,  
    WORD ProcNum,  
    LPSTR PPPathName,  
    PTECH_WRITE_MODE WriteMode,  
    WORD Sheet,  
    WORD Line,  
    WORD Column,  
    WORD NumVar,  
    DOUBLE * pData,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

PPPathName

[in] Pathname of the part program where the values will be loaded. If this field is empty, then values will be loaded into the \$PTECH variable associated with current part program, otherwise data will be loaded into the \$PTECH associated with the partprogram whose name is in PPPathName.

WriteMode

[in] identifies the access modality for the operation. Available modalities are:

PTECH_WRITE_SHEET_C	writes a whole sheet of \$PTECH variable
PTECH_WRITE_LINE_C	writes a line of a sheet of \$PTECH variable
PTECH_WRITE_ELEMENT_C	writes an element of a line of a sheet of \$PTECH variable

Sheet

[in] number of the sheet of \$PTECH variable where data will be loaded.

Line

[in] number of the line where data will be loaded. This parameter is valid when modality is **PTECH_WRITE_LINE_C** or **PTECH_WRITE_ELEMENT_C**.

Column

[in] number of the column where data will be loaded. This parameter is valid when modality is **PTECH_WRITE_ELEMENT_C**.

NumVar

[in] dimension, in byte, of sent data. It is used to check against overflows during data loading.



CndexLinkUser DLL documentation Interface with Cndex Server

pData

[in] pointer to the memory where data are stored .

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), refer to the chapter "Error Management"

See also

GetPTechSizes_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetPTechSizes_C

Returns the three dimensions of \$PTECH variables

```
WORD GetPTechSizes_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD *    Ind1,  
    WORD *    Ind2,  
    WORD *    Ind3,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Ind1

[out] number of sheets of \$PTECH variable

Ind2

[out] number of lines in a sheet of \$PTECH variable

Ind3

[out] number of columns in a line of \$PTECH variable

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), refer to the chapter "Error Management".

See also

LoadPTech_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetMdiString_C

Sets the command to be executed in the event of a cycle start in MDI process mode. The execution of this command is initiated by the Cycle_C function when the MDI mode has been selected through the SetProcessMode_C function.

```
WORD SetMdiString_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    LPSTR    pMdiString,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pMdiString

[in] Pointer to a string containing the command

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

Cycle_C, SyncroCycle_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetProcessMode_C

Sets the operating mode (MDI, Manual, Part Program Execution, etc.) of the process.

```
WORD SetProcessMode_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    Mode,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Mode

[in] Value corresponding to the process mode:

- 1 : MDI
- 2 : AUTO
- 3 : STEP
- 4 : MANUAL JOG
- 5 : INCREMENTAL JOG
- 6 : RETURN TO PROFILE
- 7 : HOME
- 8 : HANDWHEEL

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 120 The value of mode Mode is out of range.

See also

GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SelectProcess_C

Sets the working process, i.e., the process selected, on the CNC.

Changing the process selected on the CNC has no interactions with the functions of this library, the reference process for these functions is always specified as a parameters of the individual functions.

```
WORD SelectProcess_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Identifier of the process to be selected on the CNC (a number from 1 to 24).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetSelectedProcess_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetSelectedProcess_C

Reads the number of the process selected on the CNC.

```
WORD GetSelectedProcess_C (  
    WORD    UserSession,  
    WORD    *pProcNum,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pProcNum

[in] Identifier of the process to be selected on the CNC (a number from 1 to 24).

pProcNum

[out] Pointer to a variable where the number of the process selected on the CNC will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SelectProcess_C



CndexLinkUser DLL documentation Interface with Cndex Server

SelectProcAxis_C

Selection of process axis for manual mode movements.

```
WORD SelectProcAxis_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    BYTE      AxisName,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

AxisName

[in] Axis name

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 121 The axis name AxisName is wrong.

See also

GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SelectPartProgram_C

Selects the part program to be executed for a process. It can also be used to de-select the current part-program.

```
WORD SelectPartProgram_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    LPSTR      pPPName,  
    DWORD      *pErrClass,  
    DWORD      *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pPPName

[in] Pointer to the string containing the part program path-name. The path-name must refer to a physical drive of the CNC. If the string is empty the function de-select the current part-program.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1	Syntax error.
Class 4 – Error 8	Format error.
Class 4 – Error 9	Symbol not defined.
Class 4 – Error 53	Label duplicated.
Class 4 – Error 55	Label too long.
Class 4 – Error 56	Overflow of subroutine table.
Class 4 – Error 57	Overflow of label table.
Class 4 – Error 65	Error during management of part-program file.
Class 4 – Error 66	Part-program file not found.

See also

SelectPartProgramFromDrive_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

SelectPartProgramFromDrive_C

Selects the part program to be executed for a process. It can also be used to de-select the current part-program.

```
WORD SelectPartProgramFromDrive_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    LPSTR       pPPName,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pPPName

[in] Pointer to the string containing the part program path-name. The path-name must refer to a logical drive configured through FileBrowser application. If the string is empty the function de-select the current part-program.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 1	Syntax error.
Class 4 – Error 8	Format error.
Class 4 – Error 9	Symbol not defined.
Class 4 – Error 53	Label duplicated.
Class 4 – Error 55	Label too long.
Class 4 – Error 56	Overflow of subroutine table.
Class 4 – Error 57	Overflow of label table.
Class 4 – Error 65	Error during management of part-program file.
Class 4 – Error 66	Part-program file not found.

See also

SelectPartProgram_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

ManagePartProgram_C

Executes an operation on a part program on a given process.

```
WORD ManagePartProgram_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    WORD        Mode,  
    LPSTR       pPPName,  
    LPSTR       ErrString,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Mode

[in] Requested operation. Available operations are:

REL_SPG	de-select a part-program. This operation corresponds to command SelectPartProgramFromDrive_C or SelectPartProgram_C when executed with an empty pPPName
SPG_EXE	select a part-program. This operation corresponds to command SelectPartProgram_C
SPG_PASSANTE	activate through mode. This operation corresponds to command DncInit_C
SPG_CON_DRIVE	activate a part-program whose name contains a logical drive name. This operation corresponds to command SelectPartProgramFromDrive_C
SPG_NO_ANA	activate a part program, no pre-analysis done
SPG_NO_ANA_DRIVE	activate a part program whose name contains a logical drive name, no pre-analysis done
SPG_SWAP	swap current program and a pre-loaded one. Name of pre-loaded program is stored in pPPName
SPG_SWAP_DRIVE	swap current program and a pre-loaded one. Name of pre-loaded program is stored in pPPName. pPPName contains a logical drive name
PRELOAD	preload a part-program
PRELOAD_DRIVE	preload a part-program whose name contains a logical drive name
UNLOAD	unload a preloaded part program
UNLOAD_DRIVE	unload a preloaded part program whose name contains a logical drive name
UNLOAD_TUTTI	upload all preloaded part programs

pPPName



CndexLinkUser DLL documentation Interface with Cndex Server

[in] Pointer to the string containing the part program path-name. The path-name must refer to a logical drive configured through FileBrowser application. If the string is empty the function de-select the current part-program.

ErrString

[out] If any error occurred this string may contain some info

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 23 – Error 1	Syntax error.
Class 23 – Error 6	Format error.
Class 23 – Error 56	Symbol not defined.
Class 23 – Error 155	Label duplicated.
Class 23 – Error 157	Label too long.
Class 23 – Error 158	Overflow of subroutine table.
Class 23 – Error 159	Overflow of label table.
Class 23 – Error 103	Error during management of part-program file.
Class 23 – Error 100	Part-program file not found.

See also

SelectPartProgram_C, SelectPartProgramFromDrive_C, GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetActivePartProgram_C

Reads the name of the selected part program and subprogram. It also returns the subprogram level reached.

```
WORD GetActivePartProgram_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD *     pLevel,  
    LPSTR      pMain,  
    LPSTR      pSub,  
    DWORD *    pErrClass,  
    DWORD *    pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pLevel

[in] Pointer to the variable where the subprogram nesting level reached by current subprogram will be written. The nesting level value ranges from 1 to 6, where 1 identifies the main program.

pMain

[out] Pointer to a 130 characters long string where the name of the selected main program will be written. The text ends with a null character (ASCII code 0).

pSub

[out] Pointer to a 130 characters long string where the name of the selected subprogram will be written. The text ends with a null character (ASCII code 0).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetActivePartProgramFullPath_C

Reads the full path and name of the selected part program and subprogram. It also returns the subprogram level reached.

```
WORD GetActivePartProgramFullPath_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    WORD *      pLevel,  
    LPSTR       pMain,  
    LPSTR       pSub,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pLevel

[in] Pointer to the variable where the function will write the nesting level reached by current subprogram. The nesting level value ranges from 1 to 6, where 1 identifies the main program.

pMain

[out] Pointer to a 130 characters long string where the function will write the full path and name of the selected main program. The text ends with a null character (ASCII code 0).

pSub

[out] Pointer to a 130 characters long string where the function will write the name of the selected subprogram. The text ends with a null character (ASCII code 0).

pErrClass

[out] Pointer to the variable where the function will write the class of the error, if an error happens.

pErrNum

[out] Pointer to the variable where the function will write the number of the error, if an error happens

Return value

Different from zero if the function successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetPartProgramLines_C

Reads up to 8 lines of the executing program. Line 2 (pPPLine2) is the executing line when the CNC is in RUN state. It is the line to be executed when the CNC is in IDLE state.

```
WORD GetPartProgramLines_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    LPSTR     pPPLine1,  
    LPSTR     pPPLine2,  
    LPSTR     pPPLine3,  
    LPSTR     pPPLine4,  
    LPSTR     pPPLine5,  
    LPSTR     pPPLine6,  
    LPSTR     pPPLine7,  
    LPSTR     pPPLine8,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pPPLine1, PPLine2... PPLine8

[out] Pointers to strings of 130 characters where the program lines will be written. Each line is terminated with zero.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetAxOriginNum_C

Reads the number and type of the origin set for the axes of a process.

```
WORD GetAxOriginNum_C (  
    WORD UserSession,  
    WORD ProcNum,  
    WORD * pAxesNum,  
    AX_ORIG_NUM_C4 * pAxOriginNum,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pAxesNum

[in,out] Pointer to a variable:

[In] in input it must be initialized with the number of the elements of the array pointed by pAxOriginNum.

[Out] in output it will contain the number of read axes, that is the number of valid elements in the array.

pAxOriginNum

[out] Pointer to an array of AX_ORIG_NUM_C4 structures where the data will be written. The number of elements in the array must be at least equal to the number specified with pAxesNum. The maximum number of axes per process is 9. The "OriginFlag" structure field specify the type of origin active for the axis.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

Struttura AX_ORIG_NUM_C4 nel paragrafo "Descrizione delle Strutture"



CndexLinkUser DLL documentation Interface with Cndex Server

GetAxesPosition_C

Returns information on the position of interpolated and auxiliary process axes

```
WORD GetAxesPosition_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    BYTE        AxisName,  
    WORD        Select,  
    WORD        *pNumAxis,  
    GETINTDATA_C4 * pIntPos ,  
    DWORD        *pErrClass,  
    DWORD        *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

AxisName

[in] Axis name.

Must give ASCII name of axis when the position of a single axis is requested. It must be 0 to request the positions of several axes.

Select

[in] Values corresponding to type of data that you want to read:

- 1 : Programmed position (the axis position includes all offsets, origins, etc.). Corresponds to the PROGRAMMED positions displayed by the CNC.
- 2 : Interpolated position (is the output of the calculation, does not include the offsets, origins, etc.). By summing the Position field and the TotalOffset field, we get the WORK positions displayed by the CNC
- 3 : Transducer position (is the position read by the transducer, does not include the offsets, origins, etc.). By summing the Position field and the TotalOffset field, we get the MACHINE positions displayed by the CNC.
- 4 : Following error (this is position error determined as difference between the theoretical and the physical position)
- 5 : Distance to go (is the difference between the current position and the position to be reached)
- 6 : Interpolated position (is the output of the calculation, does not include the offsets, origins, etc.). Corresponds to the ABSOLUTE positions displayed by the CNC.

pNumAxis

[in,out] Pointer to a variable:

[In] initialised with the number of elements in the array.

[Out] containing the number of axes configured, i.e., number of significant elements in the array.



CndexLinkUser DLL documentation Interface with Cndex Server

pIntPos

[out] Pointer to a GETINTDATA_C4 array.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetNcInfo1_C

Returns a set of process parameters.

```
WORD GetNcInfo1_C (  
    WORD UserSession,  
    WORD ProcNum,  
    GETINFO1DATA_C4 *pGetInfo,  
    DWORD *pErrClass,  
    DWORD *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pGetInfo

[out] Pointer to the GETINFO1DATA_C4 type structure that will contain the data

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GETINFO1DATA_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

GetNcInfo2_C

Reads some process-related parameters.

```
WORD GetNcInfo2_C (  
    WORD UserSession,  
    WORD ProcNum,  
    GETINFO2DATA_C4 *pGetInfo,  
    DWORD *pErrClass,  
    DWORD *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pGetInfo

[out] Pointer to the GETINFO2DATA_C4 type structure that will contain the data

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GETINFO2DATA_C4 structure in paragraph "Description of the Structures"

GetNcInfo1_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetMarkerInfo_C

Reads some parameters related to the part program in execution

```
WORD GetMarkerInfo_C (  
    WORD UserSession,  
    WORD ProcNum,  
    BYTE Request,  
    MARKER_INFO_C4 ptMarkerInfo,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identificatore Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Request

[in] flag that specifies if data read refers to nearest point or to the element in execution.

ptMarkerInfo

[out] Pointer to the MARKER_INFO_C4 type structure that will contain the data

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

MARKER_INFO_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

GetToolNames_C

Reads the name of the current and programmed tools.

```
WORD GetNcInfo2_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    LPSTR       pTool,  
    LPSTR       pProgTool,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pTool

[out] Pointer to a string of characters that will contain the name of the current tool.
The string must consist of at least 34 characters.

pProgTool

[out] Pointer to a string of characters that will contain the name of the programmed tool.
The string must consist of at least 34 characters.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetProcessStatus_C

Returns information on process mode and status.

```
WORD GetProcessStatus_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    PROCDATA_C4 *pNewMode,  
    DWORD       *pErrClass,  
    DWORD       *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pNewMode

[out] Pointer to the PROCDATA_C4 type structure that will contain the data

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

PROCDATA_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

GetBlkNum_C

Reads the number of part-programs active (main program and relative subroutines) and the part-program block active (being executed) on the remote CNC. To read valid block numbers the part-programs blocks must be numbered.

```
WORD GetBlkNum_C (  
    WORD UserSession,  
    WORD ProcNum,  
    GETBLKNUMDATA_C4 *lpGetBlkNum,  
    DWORD *pErrClass,  
    DWORD *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

lpGetBlkNum

[out] Pointer to the GETBLKNUMDATA_C4 type structure that will contain the data

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

GETBLKNUMDATA_C4 structure in the paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

GetVarE_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarDouble_C function.

Reads the value of type E process variables

```
WORD GetVarE_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      Index,  
    WORD      *pNum,  
    double    *pValue,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Index

[in] Index of the first variable to be read

pNum

[in,out] Pointer to a variable:

[In] initialised with the number of variables to be read (MAX 22).

[Out] will contain the number of variables actually read.

pValue

[out] Pointer to the array of double variables that will contain the values of the variables read.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetVarE_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetVarE_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use *WriteVarDouble_C* function.

Changes the value of type E process variables.

```
WORD SetVarE_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    Index,  
    WORD    Num,  
    double  *pValue,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Index

[in] Index of the first variable to be changed

Num

[in] Number of variables to be changed (MAX 22)

pValue

[in] Pointer to an array containing the data to be assigned to the variables. The array must contain a number of elements equal to Num

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

GetVarE_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetVarSN_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarDouble_C function.

Reads the value of type SN process variables.

```
WORD GetVarSN_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    Index,  
    WORD    *pNum,  
    double  *pValue,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Index

[in] Index of the first variable to be read

pNum

[in,out] Pointer to a variable:

[In] initialised with the number of variables to be read (MAX 16).

[Out] that will contain the number of variables actually read.

pValue

[out] Pointer to array that will contain the values of the variables read.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetVarSN_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetVarSN_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use *WriteVarDouble_C* function.

Sets the value of SN type process variables.

```
WORD SetVarSN_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      Index,  
    WORD      Num,  
    double    *pValue,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Index

[in] Index of the first variable to be changed.

Num

[in] Number of variables to be changed (MAX 16)

pValue

[in] Pointer to an array containing the data to be assigned to the variables. The array must contain a number of elements equal to Num

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

GetVarSN_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetVarSC_C

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarText_C function.

Reads the value of type SC process variables.

```
WORD GetVarSC_C (
    WORD    UserSession,
    WORD    ProcNum,
    WORD    Index,
    WORD    *pNum,
    BYTE    *pValue,
    DWORD   *pErrClass,
    DWORD   *pErrNum
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Index

[in] Index of the first variable to be read

pNum

[in,out] Pointer to a variable:

[In] initialised with the number of variables to be read (MAX).

[Out] that will contain the number of variables actually read.

pValue

[out] Pointer to the array that will contain the values of the variables read.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetVarSC_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetVarSC_C (obsolete function)

Obsolete function. It is available in this library for backward compatibility. Instead of this function you should use ReadVarDouble_C function.

Changes the value of type SC system variables.

```
WORD SetVarSC_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    Index,  
    WORD    Num,  
    BYTE    *pValue,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Index

[in] Index of the first variable to be read.

Num

[in] Number of variables to be read.

pValue

[in] Pointer to an array containing the data to be assigned to the variables. The array must contain a number of elements equal to Num.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

GetVarSC_C



CndexLinkUser DLL documentation Interface with Cndex Server

ReadErrMsg_C

Reads the process or system error messages. The messages are formulated in the language installed on the CNC.

```
WORD ReadErrMsg_C (  
    WORD UserSession,  
    WORD ProcNum,  
    ERR_MSG_C4 *pSysErrMsg,  
    DWORD *pErrClass,  
    DWORD *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pErrMsg

[out] Pointer to a structure that is compiled with the text describing the process error, if any, or the system error, if any (system errors override process errors).

Structure ERR_MSG_C4 contains 4 arrays with 40 characters each. The CNC returns the text in the arrays as shown on screen, i.e., one text line per array (from 1 to 4).

Unused arrays are reset.

Text strings, if any, do NOT have a terminator, i.e., the text of a line always takes up the 40 characters of the individual array.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

ERR_MSG_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

ReadPartProgramMsg_C

Reads the text relating to the part-program messages (DIS instruction).

```
WORD ReadPartProgramMsg_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    LPSTR      pPartProgramMsg,  
    DWORD      *pErrClass,  
    DWORD      *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pPartProgramMsg

[out] Pointer to a string that will contain the text of a message, ending with /0 (zero). The string must be pre-allocated and with a len of at least 130 characters.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

ReadCurrentErrorMsg_C

Compatibility

OPENcontrol

Reads the current process error message.

```
WORD ReadCurrentErrMsg_C (  
    WORD        UserSession,  
    WORD        ProcNum,  
    MSG_ERROR_C4* pData,  
    DWORD*      pErrClass,  
    DWORD*      pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pData

[out] Pointer to a structure that is compiled with the information describing the process error, if any.

NOTE: the FormatTxt field does not contain translated text, but rather the information required by the OPENcontrol software to generate the translated error message.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Nonzero value if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the "Error Management" chapter.

See also

MSG_ERROR_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

ReadCurrentEmergMsg_C

Compatibility

OPENcontrol

Reads the process current emergency message.

```
WORD ReadCurrentEmergMsg_C (  
    WORD UserSession,  
    WORD ProcNum,  
    MSG_EMERGENCY_C4* pData,  
    DWORD* pErrClass,  
    DWORD* pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pData

[out] Pointer to a structure that is compiled with the information describing the process emergency, if any.

NOTE: the FormatTxt field does not contain translated text, but rather the information required by the OPENcontrol software to generate the translated error message.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Nonzero value if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the "Error Management" chapter.

See also

MSG_EMERGENCY_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

ReadCurrentAnomalyMsg_C

Compatibility

OPENcontrol

Reads the current system anomaly message.

```
WORD ReadCurrentAnomalyMsg_C (  
    WORD UserSession,  
    MSG_ANOMALY_C4* pData,  
    DWORD* pErrClass,  
    DWORD* pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pData

[out] Pointer to a structure that is compiled with the information describing the system anomaly, if any.

NOTE: the FormatTxt field does not contain translated text, but rather the information required by the OPENcontrol software to generate the translated error message.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Nonzero value if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the "Error Management" chapter.

See also

MSG_ANOMALY_C4 structure in paragraph "Description of the Structures"



CndexLinkUser DLL documentation Interface with Cndex Server

GetGCode_C

Returns the activation status of the G functions of the process.

```
WORD GetGCode_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    *pGCode,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pGCode

[out] Pointer to a 14 WORD array that will contain the status of G functions

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetMCode_C

Returns the situation of the active M's of the process.

```
WORD GetMCode_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      *pMCode,  
    DWORD     *pErrClass,  
    DWORD     *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

pMCode

[out] Pointer to a 16 WORD array that will contain the active M's

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetNcInfo2_C



CndexLinkUser DLL documentation Interface with Cndex Server

SkipPProgBlock_C

Move the program cursor to skip execution one or more program lines.
The CNC accepts this command only when a part-program is selected and it is not in execution (CNC not in RUN state). The block-by-block mode must also be selected.

```
WORD SkipPProgBlock_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    WORD      NumOfBlocks,  
    WORD      Direction,  
    WORD *    pMCode,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

NumOfBlocks

[in] Number of blocks to skip to set the new part-program cursor position.

Direction

[in] Program cursor movement direction. 0 forward (end of program direction), 1 backward (begin of program direction).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetNcInfo2_C



CndexLinkUser DLL documentation Interface with Cndex Server

Ese_C

Requests the program execution till the specified line.

```
WORD Ese_C (  
    WORD      UserSession,  
    WORD      ProcNum,  
    DWORD     BlockNum,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

BlockNum

[in] Number of the program block where the program must be stopped. The program must have the sequential numbers in front of the blocks or, at least, the searched block number must be in front of the block to be reached.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Nonzero value if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

EseEx_C

Compatibility
OPENcontrol

Requests the program execution till the specified stop condition becomes true. Extends the Ese_C function by allowing multiple stop conditions.

```
WORD EseEx_C (  
    WORD UserSession,  
    WORD ProcNum,  
    RCM_TO_MODE Mode,  
    DWORD Mask,  
    DWORD BlockNum,  
    DWORD BrkVal,  
    LPWSTR Label,  
    DWORD LineNum,  
    DWORD* pErrClass,  
    DWORD* pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 24).

Mode

[in] part program stop option:

RCM_TO: the part program stops at the specified block, line, or label, whichever is encountered first. Execution resumes from the first valid line after the program stop point (see "Remarks").

RCM_NEXT: the part program stops at the specified block, line, or label. Execution resumes at next block, line, or label (see "Remarks").

Mask

[in] bit mask selecting the stop conditions to be applied. Only the conditions with the corresponding bit set to 1 are applied.

Bit	Condition
0	Block number
1	Line number
2	Label
3	\$BRK value

BlockNum

[in] Number of the program block where the part program execution must stop. The program must have the sequential numbers (Nxx) at the beginning of each block, or at least a line with a sequential number equal to the block number specified by the stop condition.



CndexLinkUser DLL documentation Interface with Cndex Server

Label

[in] Pointer to a C array of wide characters (WCHAR) containing the name of the label where the part program execution must stop. The value of Label cannot be NULL.

LineNum

[in] Number of the program line where the part program execution must stop.

BrkVal

[in] Value of the process variable \$BRK. Part program execution stops as soon as \$BRK is equal to the specified value.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Remarks

In RCM_TO mode, the part program stops as soon as any one of the stop conditions becomes true. Part program execution resumes when the process receives a CYCLE command, and execution starts from the first valid line after the program stop point.

In RCM_NEXT mode, only one stop condition can be set. The part program stops as soon as the stop condition becomes true. Part program execution resumes when the process receives a CYCLE command, and execution starts as follows:

With a block condition, part program execution resumes from the next block, i.e. the next part program line beginning with a sequence number (Nxx).

With a line condition, part program execution resumes from the first valid line after the program stop point.

With a label condition, part program execution resumes from the first label after the program stop point.

With a \$BRK value condition, part program execution resumes from the first line that modifies the value of \$BRK by assigning a value that invalidates the stop condition.

Return value

Nonzero value if the function has been successfully executed, zero if an error has occurred.

For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

AxesRef_C

Forces the condition of referred axis.

WARNING: this function declares the axis referred regardless of its position (even if it is not on the zero micro): if the axis is moved, this entails the risk of moving it to overtravel or against other machine elements, damaging the machine tool.

```
WORD AxesRef_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    WORD    NumAx,  
    LPSTR    pAxisName,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 20).

pAxisName

[in] Pointer to a string containing a list of the names of the axes to be referred, ending in /0 (zero).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

SetProcVarWord_C

Write some of the process variables that have WORD format (16 bits).

Some of these variables enable (or disable) functions and modes of the NC and they cannot be modified directly into the part program. Some of them require that the process is in a precise mode and state before the write operation can occur.

```
WORD SetProcVarWord_C (  
    WORD UserSession,  
    WORD ProcNum,  
    PROC_WORD_VAR_TYPE_C VarType,  
    WORD Value,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identified (a number from 1 to 24).

VarType

[in] Identifier of the process variable we need to write. The available identifiers are:

Identifier	Variable	Description
1	RCM	Enables Search in Memory (RCM)
2	ARM	Arc normalization Mode
3	DPS	Disables part program scroll
4	RAP	Enables Automatic Jog Return
5	TRP	Tapping Return Feedrate Percentage
6	VFF	Percentage value for Velocity Feed Forward
7	UVR	Uses rapid feed
8	DLA	Disables look-ahead
9	MBR	Enables Multi Block Retrace
10	URL	Uses override for rapid feed
11	DSB	Disables barred blocks
12	USO	Enables optional stop for M codes
13	ERR	Enables error management in part program
14	HMP	Enables multi-homing
15	UAS	Enables axes disconnected state
16	MBA	Enables auxiliary functions in Multi Block Retrace
17	REM	Return to profile mode
18	CYCTIME	Enables time estimation mode

Value

[in] Value assigned to the variable.



CndexLinkUser DLL documentation Interface with Cndex Server

pErrClass

[out] Pointer to the variable that in case of error will contain error class.

pErrNum

[out] Pointer to the variable that in case of error will contain error number.

Return Value

Different from zero if the execution was successful executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetProcVarWord_C

Reads some the process variables that have WORD format (16 bits).

```
WORD GetProcVarWord_C (  
    WORD UserSession,  
    WORD ProcNum,  
    PROC_WORD_VAR_TYPE_C VarType,  
    WORD * pValue,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identified (a number from 1 to 24).

VarType

[in] Identifier of the process variable we need to read. The available identifiers are:

Identifier	Variable	Description
1	RCM	Enables Search in Memory (RCM)
2	ARM	Arc normalization Mode
3	DPS	Disables part program scroll
4	RAP	Enables Automatic Jog Return
5	TRP	Tapping Return Feedrate Percentage
6	VFF	Percentage value for Velocity Feed Forward
7	UVR	Uses rapid feed
8	DLA	Disables look-ahead
9	MBR	Enables Multi Block Retrace
10	URL	Uses override for rapid feed
11	DSB	Disables barred blocks
12	USO	Enables optional stop for M codes
13	ERR	Enables error management in part program
14	HMP	Enables multi-homing
15	UAS	Enables axes disconnected state
16	MBA	Enables auxiliary functions in Multi Block Retrace
17	REM	Return to profile mode
18	CYCTIME	Enables time estimation mode

pValue

[out] Pointer to a WORD variables (16 bit) that will contain the value of the read variable.

pErrClass

[out] Pointer to the variable that in case of error will contain error class.

pErrNum



CndexLinkUser DLL documentation Interface with Cndex Server

[out] Pointer to the variable that in case of error will contain error number.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

Functions for the "Through Mode"

These functions are RESERVED

These functions can be called only after creating the Cndex server (see ConnectServer_C), having opened a communication session (see OpenSession_C) and having made sure that the CNC is in RUN mode (see BootPhaseEnquiry_C).

By means of the "Through Mode" it is possible to execute a Part Program transferred run-time from an external application (instead of from a file saved on the NC).

Only the **MAIN** program can be managed with the Through Mode, subroutines **CANNOT** be downloaded run-time. The main program, instead, can recall subroutines residing on the NC in file format.

The maximum size of a "Through" program, i.e., the maximum quantity of data transferred, is 2 Gbytes.

The NC saves the data received from the network in a circular buffer of 65000 characters; this area should be construed as a "window" open on the entire "Through" program. The circular buffer is downloaded from the NC during the execution of the instructions contained in it so as to make it possible to load further data; the data already downloaded (in as much as relating to instructions already executed) are no longer accessible.

During the execution of the Through program, when the end of the active area is reached (i.e., the area has been emptied out) the NC will wait for further data, unless the program "end" has been notified: in the latter case, the NC will end program execution.

Once the execution has been completed (or when the NC is on IDLE), the Through Mode can be closed.

The Reset command does not close the Through mode, rather, it empties out the active area and goes back to program start.



CndexLinkUser DLL documentation Interface with Cndex Server

DncInit_C

Reserved function.

Activates the Through Mode.

```
WORD DncInit_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    LPSTR    pPathName ,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 20).

pPathName

[in] Pointer to the string containing the name to be displayed as active Program

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 160 Command not available in current status.



CndexLinkUser DLL documentation Interface with Cndex Server

DncData_C

Reserved function.

Enters the Part Program lines to be executed in Through Mode.

```
WORD DncData_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    LPSTR    pData,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 20)..

pData

[in] Pointer to the area containing the program lines to be executed (MAX 178 characters). The data must be terminated with character /0 (zero). In a data block it is possible to specify several part program lines, separated by characters CR-LF (Carriage Return – Line Feed).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 11	Function not permitted (mode not active)
Class 4 – Error 10	Memory full
Class 4 – Error 65	Through program too big (2Gbytes)



CndexLinkUser DLL documentation Interface with Cndex Server

DncEof_C

Reserved function.

Informes the NC that all the Part Program lines have been transferred (Part Program end).

```
WORD DncEof_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ProcNum

[in] Process identifier (a number from 1 to 20)..

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

DncStop_C

Reserved function.

Deactivates the Through Mode.

```
WORD DncStop_C (  
    WORD    UserSession,  
    WORD    ProcNum,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 4 – Error 160 Command not available in current status.



CndexLinkUser DLL documentation Interface with Cndex Server

Generic functions

These functions can be called only after creating the Cndex server (see `ConnectServer_C`), having opened a communication session (see `OpenSession_C`) and having made sure that the CNC is in RUN mode (see `BootPhaseEnquiry_C`).



CndexLinkUser DLL documentation Interface with Cndex Server

GetAxesInfo3_C

Reads the process number, axis name, axis type and interface type for all configured axes.

```
WORD GetAxesInfo3_C (  
    WORD        UserSession,  
    WORD        AxisID,  
    WORD        *pAxesNumber,  
    WORD        *pAxOwnerList,  
    BYTE        *pAxNameList,  
    WORD        *pAxType,  
    WORD        *pAxInterface,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

AxisID

[in] selects the axis ID to request data for. Use 0xFFFF to acquire data for all configured axes. In OPENControl this parameter is ignored and data is acquired for all axes.

pAxesNumber

[in] Size of axis data vectors. Must be at least 32 for Series10 and 64 for OPENcontrol.
[out] Number of configured axes.

pAxOwnerList

[out] WORD vector containing the ID of the owner process for all configured axes. The n^{th} element of the vector contains the owner process ID for the axis with ID equal to $n + 1$. The value 0 identifies the machine logic. If the axis is not configured the value is 0xFFFF.

pAxNameList

[out] BYTE vector containing the ASCII code of the axis name for all configured axes. The n^{th} element of the vector contains the axis name for the axis with ID equal to $n + 1$. If the axis is not configured the value is 0.

pAxType

[out] WORD vector containing the bit mask defining the axis type for all configured axes. The n^{th} element of the vector contains the axis type for the axis with ID equal to $n + 1$. If the axis is not configured the value is 0.

pAxInterface

[out] WORD vector containing the interface type for all configured axes. The n^{th} element of the vector contains the interface type for the axis with ID equal to $n + 1$. If the axis is not configured the value is 0.



CndexLinkUser DLL documentation Interface with Cndex Server

GetCodeNumber_C

Read the numerical control code number and the identifier of the installed software version.

```
WORD GetCodeNumber_C (  
    WORD    UserSession,  
    LPSTR    pCodeNumber,  
    LPSTR    pSwVersion,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pWarningMsg

[out] Pointer to a (at least) 20-character string in which the code number from the CNC will be written.

pSwVersion

[out] Pointer to a (at least) 20-character string in which the CNC software version will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetOptions_C

Reserved function.

Reads the flags map of the enabled software options, the flags map of the enabled functions and the identifier of the software version installed on the CNC.

```
WORD GetOptions_C (  
    WORD          UserSession,  
    BYTE *        pOption,  
    BYTE *        pSecurLevel,  
    LPSTR         pSwVersion,  
    DWORD *       pErrClass,  
    DWORD *       pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pOption

[out] Pointer to an array of at least 19 BYTE where the CNC enabled software options flags map will be written. To use this reserved array you need to request the details documentation to OSAI.

pSecurLevel

[out] Pointer to an array of at least 12 BYTE where the CNC enabled functions map will be written. The first two bytes indicate the level selected through Security application. The remaining 10 bytes are the flags map set through Security for the selected level.

pSwVersion

[out] Pointer to a (at least) 20-character string in which the CNC software version will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

GetDateTime_C

Requests date and time set on the CNC.

```
WORD GetDateTime_C (  
    WORD      UserSession,  
    WORD *    pYear,  
    WORD *    pMonth,  
    WORD *    pDay,  
    WORD *    pHour,  
    WORD *    pMinute,  
    WORD *    pSecond  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pYear

[out] Pointer to a variable where the year will be written.

pMonth

[out] Pointer to a variable where the month will be written.

pDay

[out] Pointer to a variable where the day will be written.

pHour

[out] Pointer to a variable where the hours will be written.

pMinute

[out] Pointer to a variable where the minutes will be written.

pSecond

[out] Pointer to a variable where the seconds will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

See also

[SetDateTime_C](#)



CndexLinkUser DLL documentation Interface with Cndex Server

SetDateTime_C

Set the date and time on the CNC.

```
WORD GetDateTime_C (  
    WORD        UserSession,  
    WORD        Year,  
    WORD        Month,  
    WORD        Day,  
    WORD        Hour,  
    WORD        Minute,  
    WORD        Second  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Year

[in] Year that will set on the CNC.

Month

[in] Month that will set on the CNC.

Day

[in] Day that will set on the CNC.

Hour

[in] Hours that will set on the CNC.

Minute

[in] Minutes that will set on the CNC.

Second

[in] Seconds that will set on the CNC.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

See also

[GetDateTime_C](#)



CndexLinkUser DLL documentation Interface with Cndex Server

GRead_C

Reads a shared memory area of the CNC. Shared memory areas of the system are addressed by means of their identifier. To use this function you must know the identifier of the system area that you want to read and its structure. This information is provided by OSAI on request.

```
WORD GRead_C (  
    WORD    UserSession,  
    WORD    MemId,  
    WORD    Offset,  
    WORD    Length,  
    byte    *pBuffer,  
    WORD    *pBufferLength,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

MemId

[in] Identifier of the memory to be read.

Offset

[in] Offset starting from which the memory is read.

Length

[in] Pointer to a variable which determines the length of the data to be read.

pBuffer

[in] Pointer to the data buffer

pBufferLength

[in] Length of the data buffer

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

GWrite_C



CndexLinkUser DLL documentation Interface with Cndex Server

GWrite_C

Writes a shared memory area of the CNC. Shared memory areas of the system are addressed by means of their identifier. To use this function you must know the identifier of the system area that you want to write and its structure. This information is provided by OSAI on request.

```
WORD GWrite_C (  
    WORD    UserSession,  
    WORD    MemId,  
    WORD    Offset,  
    WORD    Length,  
    byte    *pBuffer,  
    WORD    *pBufferLength,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

MemId

[in] Identifier of the memory to be written

Offset

[in] Offset starting from which the memory is written

pLength

[in, out] Pointer to a variable which:

[inp] determines the length of the data to be written.

[out] communicates the number of bytes actually written.

pBuffer

[in] Pointer to the data buffer

BufferLength

[in] Length of the data buffer

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

GRead_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetIpAddress_C

Changes the IP address of the control. The system automatically restarts after the change.

```
WORD SetIpAddress_C (  
    WORD        UserSession,  
    LPTSTR      IpAddress,  
    LPTSTR      SubNetMask,  
    LPTSTR      DefaultGateway,  
    LPTSTR      Dns,  
    LPTSTR      Wins,  
    LPTSTR      Net,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

IpAddress

[in] New Ip Address.

SubNetMask

[in] New subnetmask.

DefaultGateway

[in] New default gateway.

Dns

[in] New Dns.

Wins

[in] New Wins.

Net

[in] Network board id(0:First board - 1:Second board) .

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

Additional error codes:

Class 17 Error 265 Error while changing the IP address



CndexLinkUser DLL documentation Interface with Cndex Server

GetSerialNumber_C

Read the numerical control code number.

```
WORD GetSerialNumber_C (  
    WORD    UserSession,  
    LPSTR    pSerialNumber,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pSerialNumber

[out] Pointer to a (at least) 20-character string in which the code number from the CNC will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

ReadRemapDefinitions_C

Reads from CNC the array of the I/O remap definitions

```
WORD ReadRemapDefinitions_C (  
    WORD                UserSession,  
    int                 ArraySize,  
    REMAP_DEF_C4 *      pArray,  
    int                 Type,  
    int *               pDefinitionCnt,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ArraySize

[in] Size of the pArray array

pArray

[out] Pointer to an array of structures of type REMAP_DEF_C4 that will contain the I/O remap definitions

Type

[in] if 0 the function reads the input remap definitions; if 1 the function reads the output remap definitions

pDefinitionCnt

[out] number of definitions read by the function

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

WriteRemapDefinitions_C



CndexLinkUser DLL documentation Interface with Cndex Server

WriteRemapDefinitions_C

Writes to CNC the array of the I/O remap definitions

```
WORD WriteRemapDefinitions_C (  
    WORD                UserSession,  
    int                 ArraySize,  
    REMAP_DEF_C4 *      pArray,  
    int                 Type,  
    int                 Mode,  
    DWORD *             pErrClass,  
    DWORD *             pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

ArraySize

[in] size of pArray array

pArray

[in] Pointer to an array of structures of type REMAP_DEF_C4 that contain the I/O remap definitions

Type

[in] if 0 the function writes the input remap definitions; if 1 the function writes the output remap definitions

Mode

[in] This parameters may have the following values:

- 0: the function overwrites all the I/O remap definitions
- 1: the function adds one I/O remap definition
- 2: the function removes I/O remap definition

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

ReadRemapDefinitions_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetServoPar_C

Reads the value of an axis parameter

```
WORD GetServoPar_C (  
    WORD      UserSession,  
    WORD      AxisId,  
    WORD      ParId,  
    double *   pValue,  
    DWORD *    pErrClass,  
    DWORD *    pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

AxisId

[in] Axis id

ParId

[in] Parameter id

pValue

[out] Parameter value read by the function

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

SetServoPar_C



CndexLinkUser DLL documentation Interface with Cndex Server

SetServoPar_C

Writes the value of an axis parameter

```
WORD SetServoPar_C (  
    WORD      UserSession,  
    WORD      AxisId,  
    WORD      ParId,  
    double    Value,  
    DWORD *   pErrClass,  
    DWORD *   pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

AxisId

[in] Axis id

ParId

[in] Parameter id

Value

[in] Parameter value written by the function

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

GetServoPar_C



CndexLinkUser DLL documentation Interface with Cndex Server

GetCNCRegKey_C

Reads the value of a CNC registry key.

```
WORD GetCNCRegKey_C (  
    WORD        UserSession,  
    LPWSTR      Key,  
    LPWSTR      SubKey,  
    DWORD *     pValueLen,  
    BYTE *      pValue,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of a communication session with the CNC.

Key

[in] path of the key to be accessed, relative to HKEY_LOCAL_MACHINE\Osai. The full path is computed as follows:

HKEY_LOCAL_MACHINE\Osai\<Key>

Registry keys outside HKEY_LOCAL_MACHINE\Osai cannot be read using this function.

SubKey

[in] name of the subkey to be read.

pValueLen

[in, out] pointer to a 32 bit unsigned integer containing the length of the array that pValue points to. After the function call returns the integer contains the actual length of the registry value. If the array is too small to contain the registry value the function call returns a class 17 number 221 error.

pValue

[in] Pointer to a byte array where the registry value will be copied.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

File system management functions

The functions described in this paragraph allow to manage the file system of the OSAI machines (Serie10, OPENControl, SyncMotion, Top5, PC) using a common layer.

The OSAI machines remote file system bases its structure on **logical drives**. The logical drives are directories identified by a logical name.

In all functions of this paragraph requiring a path-name, a path-name referred to a logical drive must be specified.

As an example it is possible to define a logical drive named "Programs" that identify the directory "[F:\My-Programs](#)" of a defined CNC.

The file system functions using a path-name will refer to "Programs" logical name of that 10/Series CNC and will not use the physical path "[F:\MyPrograms](#)".

Supposing that a sub-directory F:\MyPrograms\dir1 (referred to previous example) containing the file Example.txt exists, the path-name to be used in the function to specify the file will be Programs\dir1\Examples.txt that is the name of the drive followed by sub-directory and file name (please note that it is NOT specified any backslash in front of the logical drive name).

The logical drives are normally configured by OEM through the FileBrowser application belonging to WinNBI human interface (see the logical drive definition in the FileBrowser help).

It is also possible to manage the logical drive configuration using some of the functions described in this paragraph (see the functions LogFSAddDrive_C, LogFSRemoveDrive_C).

On all OSAI target (excluding PC) are also available some predefined logical drives. These drives are not visible in any part of the WinNBI interface (FileBrowser etc.). The drives are defined **Hidden logical drive** and can be used only by software through the functions of this paragraph.

The configuration of the hidden logical drive can not be modified. The defined hidden logical drive are the following:

for OPENControl, SyncMotion, Top5 systems

"DEVICE" that refers to directory "\\SSD"

for 10/Series systems (when they are in **RUN**, **EMERGENCY** or **SERVICE** boot modality)

"**SYS**" that refers to directory "C:"

"**PRJ**" that refers to directory "D:"

"**USR**" that refers to directory "E:"

"**UPP**" that refers to directory "F:"

These functions can be called only after creating the Cndex server (see ConnectServer_C), having opened a communication session (see OpenSession_C) and having made sure that the CNC is in RUN, EMERGENCY or SERVICE mode (see BootPhaseEnquiry_C).

This is not true for two file transfer functions (LogFSTransferFile_C e LogFSTransferFileW_C) that manage by them self the communication session opening.



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSTransferFile_C

Copy a file. The file can be copy from one CNC to another one.

The difference between this function and the function LogFSTransferFileW_C is due to the format of the parameters.

```
WORD LogFSTransferFile_C (  
    LPCSTR          SourceTargetName,  
    LPCSTR          SourceFilePath,  
    LPCSTR          DestTargetName,  
    LPCSTR          DestFilePath,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

SourceTargetName

[in] Network name of the target from where to get the source file (see the parameter RemoteName of function OpenSession_C for the syntax of the name).

SourceFilePath

[in] Full path-name of the source file to read.

DestTargetName

[in] Network name of the target where to copy the destination file (see the parameter RemoteName of function OpenSession_C for the syntax of the name).

DestFilePath

[in] Full path-name of the destination file to write. It must include the destination file name.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSTransferFileW_C, LogFSCopyFile_C



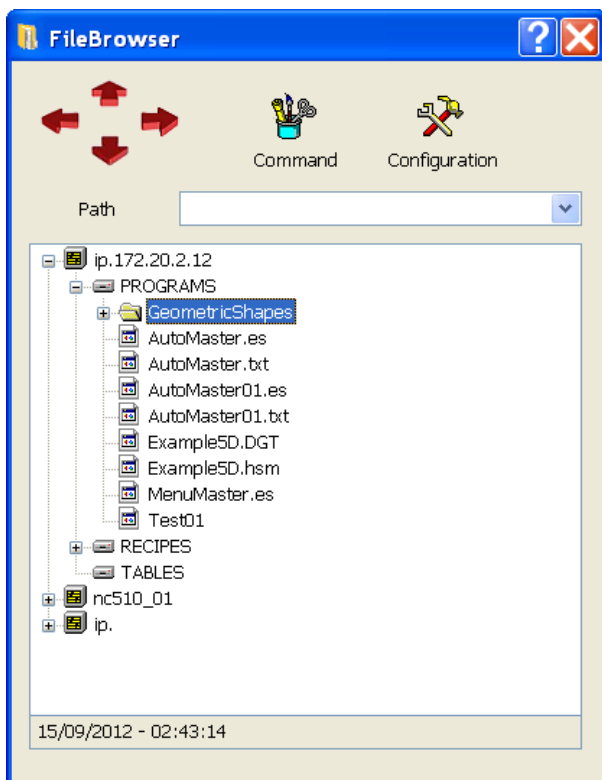
CndexLinkUser DLL documentation Interface with Cndex Server

```
//This C# code sample copies the text file "square.txt" from the local
//directory "C:\Programs" to the remote logical drive "PROGRAMS"
//in the "GeometricShapes" directory

//Creates a new Cndex object
Cndex cndexobj = new Cndex();

//Unlike other functions in the user library, LogFSFileTransfer_C
//and LogFSFileTransferW_C do not require a session number
if (cndexobj.LogFSTransferFileW_C(
    null, //null indicates the local machine
    "C:\\Programs\\Square.txt", //local file system path can be used
    "IP.172.20.2.12", //Remote machine IP address
    //The "IP." prefix identifies an OPENcontrol or PC
    "PROGRAMS\\GeometricShapes\\Square.txt", //The remote path must use
    //the <logical drive>\\<subdirectory> syntax.
    out m_ErrClass, out m_ErrNum) > 0)
    MessageBox.Show("File transfer successful.");
else
{
    String msg = string.Format("File transfer error {0}/{1:X}h",
        m_ErrClass, m_ErrNum);
    MessageBox.Show(msg);
}
```

The logical drive must be configured in either File Browser or Process Controller as shown in the picture. Check the File Browser documentation for further details.





CndexLinkUser DLL documentation Interface with Cndex Server

LogFSTransferFileW_C

Copy a file. The file can be copy from one CNC to another one.

The difference between this function and the function LogFSTransferFile_C is due to the format of the parameters.

```
WORD LogFSTransferFileW_C (  
    LPCWSTR SourceTargetName,  
    LPCWSTR SourceFilePath,  
    LPCWSTR DestTargetName,  
    LPCWSTR DestFilePath,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

SourceTargetName

[in] Network name of the target from where to get the source file (see the parameter RemoteName of function OpenSession_C for the syntax of the name).

SourceFilePath

[in] Full path-name of the source file to read.

DestTargetName

[in] Network name of the target where to copy the destination file (see the parameter RemoteName of function OpenSession_C for the syntax of the name).

DestFilePath

[in] Full path-name of the destination file to write. It must include the destination file name.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSTransferFile_C, LogFSCopyFile_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSSetSecurityLevel_C

Set the security level to use for the file system.

```
WORD LogFSSetSecurityLevel_C (  
    WORD UserSession,  
    LPCWSTR Password,  
    SECURITY_LEVEL_C SecurityLevel,  
    SECURITY_LEVEL_C * pOldSecurityLevel,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

Password

[in] Password (OEM) permitting the modification of the active security level. Currently it is not used. It will be activated in next releases. An application using this function should request to the user the level and password to be inserted in the parameters.

SecurityLevel

[in] Security level to be set. The valid levels are the following:

SECURITY_LEV_ADMIN	= 1
SECURITY_LEV_SERVICE	= 2
SECURITY_LEV_OEM_ADMIN	= 3
SECURITY_LEV_OEM_SERVICE	= 4
SECURITY_LEV_USER_ADMIN	= 5
SECURITY_LEV_USER_SERVICE	= 6

The level 1 and 2 are reserved for OSAI. Others level can be used by OEM/end-user.

With this SW version it is mandatory to require the SECURITY_LEV_OEM_ADMIN level to have the rights to insert or remove logical drives from the target configuration.

pOldSecurityLevel

[out] Active security level before the call to this function.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSGetSecurityLevel_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetSecurityLevel_C

Reads the security level set for the file system.

```
WORD LogFSGetSecurityLevel_C (  
    WORD UserSession,  
    SECURITY_LEVEL_C * pSecurityLevel,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pSecurityLevel

[out] Pointer to the variable where the current security level will be written. See function LogFSSetSecurityLevel_C for more informations.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSSetSecurityLevel_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSLongFileNames_C

This function allow to verify if the CNC operating system manages long files name natively. The OPENControl and SyncMotion manage the long files name natively while 10/Series implements the long file name management through the functions described in the paragraph "Functions for long file names management".

```
WORD LogFSLongFileNames_C (  
    WORD          UserSession,  
    WORD *        pUseLongFileNames,  
    DWORD *       pErrClass,  
    DWORD *       pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pUseLongFileNames

[out] Pointer to the variable where the data will be written. If the value is 0 the system doesn't manage the long files name. In this case the DOS format file names must be used.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetNumDrive_C

Reads the number of user logical drives and hidden drives currently active (see introduction of this paragraph).

```
WORD LogFSGetNumDrive_C (  
    WORD UserSession,  
    WORD * pNumHiddenDrive,  
    WORD * pNumUserDrive,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pNumHiddenDrive

[out] Pointer of the variable where the number of active hidden logical drive will be written.

pNumUserDrive

[out] Pointer of the variable where the number of active user logical drive will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSGetHiddenDriveList_C, LogFSGetDriveList_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetDriveList_C

Reads, one by one, the names of the active user logical drives (see introduction of this paragraph).

```
WORD LogFSGetDriveList_C (  
    WORD        UserSession,  
    LPWSTR      DriveName,  
    WORD        index,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DriveName

[out] Pointer of the variable where the name of the user logical drive will be written.

index

[in] Index of the drive of whom you want to read the name. Must not be equal or greater than the number of user logical drive read with the function LogFSGetNumDrive_C.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSGetHiddenDriveList_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetHiddenDriveList_C

Reads, one by one, the names of the active hidden logical drives (see introduction of this paragraph).

```
WORD LogFSGetHiddenDriveList_C (  
    WORD        UserSession,  
    LPWSTR      DriveName,  
    WORD        index,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DriveName

[out] Pointer of the variable where the name of the hidden logical drive will be written.

index

[in] Index of the drive of whom you want to read the name. Must not be equal or greater than the number of hidden logical drive read with the function LogFSGetNumDrive_C.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSGetDriveList_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetDrivePath_C

Reads the physical path-name linked with a configured logical drive.

```
WORD LogFSGetDrivePath_C (  
    WORD UserSession,  
    LPCWSTR DriveName,  
    LPWSTR DrivePath,  
    SECURITY_LEVEL_C SecurityLevel,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DriveName

[in] Name of the logical drive of whom the linked path-name is required.

DrivePath

[out] Variables where the physical path-name will be written.

SecurityLevel

[in] Security level to whom this request is referred. Currently it is mandatory to write the value SECURITY_LEV_OEM_ADMIN. The same level must be previously set (see function LogFSSetSecurityLevel_C for more details).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSSetSecurityLevel_C.



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSAddDrive_C

Adds a logical drive to the drive configuration.

The new drive will be available only after the activation of the new configuration that must be requested through the function LogFSReloadDriveList_C.

The function return an error if the request is relevant to a PC. The logical drive for a PC can only be configured by the OSAI FileBrowser application (belonging to WinNBI product) running locally on that PC.

```
WORD LogFSAddDrive_C (
    WORD                UserSession,
    LPCWSTR             DriveName,
    LPCWSTR             PathName,
    BOOL                Temporary,
    SECURITY_LEVEL_C     SecurityLevel,
    DWORD *             pErrClass,
    DWORD *             pErrNum
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DriveName

[in] Name of the logical drive to add to the configuration.

PathName

[out] Physical path-name of the directory linked to the logical drive.

Temporary

[in] Parameter not used. Must be set to FALSE (0).

SecurityLevel

[in] Security level to whom this request is referred. Currently it is mandatory to write the value SECURITY_LEV_OEM_ADMIN. The same level must be previously set (see function LogFSSetSecurityLevel_C for more details).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSRemoveDrive_C, LogFSReloadDriveList_C, LogFSSetSecurityLevel



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSRemoveDrive_C

Removes a logical drive from the drive configuration.

The new drive will not be available anymore only after the activation of the new configuration that must be requested through the function `LogFSReloadDriveList_C`.

```
WORD LogFSRemoveDrive_C (  
    WORD                UserSession,  
    LPCWSTR             DriveName,  
    SECURITY_LEVEL_C     SecurityLevel,  
    DWORD *              pErrClass,  
    DWORD *              pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DriveName

[in] Name of the logical drive to remove from configuration.

SecurityLevel

[in] Security level to whom this request is referred. Currently it is mandatory to write the value `SECURITY_LEV_OEM_ADMIN`. The same level must be previously set (see function `LogFSSetSecurityLevel_C` for more details).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

`LogFSAddDrive_C`, `LogFSReloadDriveList_C`, `LogFSSetSecurityLevel`



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSReloadDriveList_C

Activates the configuration modified through the functions LogFSAddDrive_C and LogFSRemoveDrive_C.

```
WORD LogFSReloadDriveList_C (  
    WORD        UserSession,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSAddDrive_C, LogFSRemoveDrive_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSCreateDir_C

Create a new directory.

```
WORD LogFSCreateDir_C (  
    WORD        UserSession,  
    LPCWSTR     PathName,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the directory to be created.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSCreateFile_C

Create a new file.

```
WORD LogFSCreateFile_C (  
    WORD        UserSession,  
    LPCWSTR     PathName,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the file to be created.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetFileSize_C

Reads the size of a file.

```
WORD LogFSGetFileSize_C (  
    WORD        UserSession,  
    LPCWSTR     PathName,  
    UINT *      pSize,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the file

pSize

[out] Pointer to the variable where the size of the file in bytes will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetFileAttrib_C

Reads the attributes of a file

```
WORD LogFSGetFileAttrib_C (  
    WORD UserSession,  
    LPCWSTR PathName,  
    DWORD * pAttrib,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the file

pAttrib

[out] Pointer to the variable where the file attributes will be written.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".

For the meaning of the single bit of the attributes it is here shown a section of the MSDN Microsoft documentation referred to the API function GetFileAttributes.

The attributes can be one or more of the following values.

Attribute	Meaning
FILE_ATTRIBUTE_ARCHIVE (0x00000020)	A file or directory that is an archive file or directory. Applications use this attribute to mark files for backup or removal.
FILE_ATTRIBUTE_COMPRESSED (0x00000800)	A file or directory that is compressed. For a file, all of the data in the file is compressed.



CndexLinkUser DLL documentation Interface with Cndex Server

FILE_ATTRIBUTE_DEVICE (0x00000040)	For a directory, compression is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_DIRECTORY (0x00000010)	Reserved; do not use.
	The handle that identifies a directory.
	A file or directory that is encrypted.
FILE_ATTRIBUTE_ENCRYPTED (0x00004000)	For a file, all data streams in the file are encrypted.
	For a directory, encryption is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_HIDDEN (0x00000002)	The file or directory is hidden. It is not included in an ordinary directory listing.
FILE_ATTRIBUTE_NORMAL (0x00000080)	A file or directory that does not have other attributes set.
	This attribute is valid only when used alone.
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED (0x00002000)	The file is not to be indexed by the content indexing service.
	The data of a file is not available immediately.
FILE_ATTRIBUTE_OFFLINE (0x00001000)	This attribute indicates that the file data is physically moved to offline storage. This attribute is used by Remote Storage, which is the hierarchical storage management software. Applications should not arbitrarily change this attribute.
	A file or directory that is read-only.
FILE_ATTRIBUTE_READONLY (0x00000001)	For a file, applications can read the file, but cannot write to it or delete it.
FILE_ATTRIBUTE_REPARSE_POINT (0x00000400)	For a directory, applications cannot delete it.
FILE_ATTRIBUTE_SPARSE_FILE (0x00000200)	A file or directory that has an associated reparse point.
FILE_ATTRIBUTE_SYSTEM (0x00000004)	A file that is a sparse file.
	A file or directory that the operating system uses a part of, or uses exclusively.
	A file that is being used for temporary storage.
FILE_ATTRIBUTE_TEMPORARY (0x00000100)	File systems avoid writing data back to mass storage if sufficient cache memory is available, because typically, an application deletes a temporary file after the handle is closed. In that scenario, the system can entirely avoid writing the data. Otherwise, the data is written after the handle is closed.
FILE_ATTRIBUTE_VIRTUAL	A file is a virtual file.



CndexLinkUser DLL documentation Interface with Cndex Server

See also

LogFSSetFileAttrib_C, LogFSChangeFileAttrib



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSSetFileAttrib_C

Set the attribute of a file.

```
WORD LogFSSetFileAttrib_C (  
    WORD        UserSession,  
    LPCWSTR     PathName,  
    DWORD       Attrib,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the file.

Attrib

[in] Attributes to set to the file specified by PathName. For the valid values see the function LogFSGetFileAttrib_C.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSGetFileAttrib_C, LogFSChangeFileAttrib



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSChangeFileAttrib_C

Modify the attributes of a file.

```
WORD LogFSChangeFileAttrib_C (  
    WORD        UserSession,  
    LPCWSTR PathName,  
    DWORD        Add,  
    DWORD        Remove,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the file.

Add

[in] Attributes to add to the file specified by PathName. For the valid values see the function LogFSGetFileAttrib_C.

Remove

[in] Attributes to remove from the file specified by PathName. For the valid values see the function LogFSGetFileAttrib_C.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSGetFileAttrib_C, LogFSSetFileAttrib_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSFindFirst_C

This function search, in a directory, a sub-directory or a file with a name corresponding to the specified name.

If the file name contains a wildcard (i.e. "*") the function permits to search for list of files and sub-directory .

If the function has been executed without errors, other than the data relevant to the found file/directory, it returns a search identifier (HANDLE) that can be used to continue the search through the function LogFSFindNext_C.

At the end of the search cycle the handle must be closed using the function LogFSFindClose;

```
WORD LogFSFindFirst_C (
    WORD          UserSession,
    LPCWSTR       pFileName,
    FILE_FIND_DATA_C4 * pFindData,
    HANDLE *       pHFinder,
    DWORD *       pErrClass,
    DWORD *       pErrNum
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pFileName

[in] Full path-name of the first file to search.

pFindData

[out] Pointer to a FILE_FIND_DATA_C4 structure where the information relevant to searched element will be written.

pHFinder

[out] Pointer to the search HANDLE. If the function is executed successfully, the handle can be used in the function LogFSFindNext_C to search next element. If no file has been found, the handle value is 0xFFFFFFFF (INVALID_HANDLE_VALUE).

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. If the search ends without finding any file, no error is returned.

For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

See also

[LogFSFindNext_C](#), [LogFSFindClose_C](#)



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSFindNext_C

This function continues the file search started with a previous call to function LogFSFindFirst_C (see LogFSFindFirst_C description).

```
WORD LogFSFindNext_C (  
    WORD UserSession,  
    FILE_FIND_DATA_C4 * pFindData,  
    HANDLE hFinder,  
    BOOL * pFound,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

pFindData

[out] Pointer to a FILE_FIND_DATA_C4 structure where the information relevant to searched element will be written.

hFinder

[out] Search handle supplied by a previous call to function LogFSFindFirst_C.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSFindFirst_C, LogFSFindClose_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSFindClose_C

Close a file search cycle opened through the function LogFSFindFirst_C (see LogFSFindFirst_C description).

```
WORD LogFSFindClose_C (  
    WORD        UserSession,  
    HANDLE      hFinder,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

hFinder

[out] Search handle supplied by a previous call to function LogFSFindFirst_C. After the call to this functions the HANDLE is not valid anymore and can not be use further.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSFindFirst_C, LogFSFindNext_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSRemoveFile_C

Delete a file.

```
WORD LogFSRemoveFile_C (  
    WORD        UserSession,  
    LPCWSTR     PathName,  
    LPCWSTR     FileName,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the file to be deleted.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSRemoveDir_C

Remove a directory and all its contents.

```
WORD LogFSRemoveDir_C (  
    WORD        UserSession,  
    LPCWSTR     PathName,  
    DWORD *     pErrClass,  
    DWORD *     pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the directory to remove.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.
For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSRename_C

Modify the name of a file or directory.

```
WORD LogFSRename_C (  
    WORD        UserSession,  
    LPCWSTR PathName,  
    LPCWSTR NewPathName,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Path-name completo del file od directory da rinominare. Deve iniziare con il nome di un drive logico.

NewPathName

[in] Nuovo nome da assegnare al file indicato da PathName.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSCopyFile_C

Copies locally a file. This function is faster than the functions LogFSTransferFile_C and LogFSTransferFileW_C because it uses the already available CNC connection but it can not transfer a file from a CNC to another.

```
WORD LogFSCopyFile_C (  
    WORD          UserSession,  
    LPCWSTR PathName,  
    LPCWSTR NewPathName,  
    BOOL          FailIfExists,  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Full path-name of the source file to read.

NewPathName

[in] Full path-name of the destination file to write. It must include the destination file name.

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".

See also

LogFSTransferFile_C, LogFSTransferFileW_C



CndexLinkUser DLL documentation Interface with Cndex Server

LogFSGetInfo_C

Read the total space and the free space of the disk where a logical drive is defined.

```
WORD LogFSGetInfo_C (  
    WORD    UserSession,  
    LPCWSTR PathName,  
    DWORD   Selector,  
    WORD    ItemCounter,  
    DWORD   pBuffer  
    DWORD * pErrClass,  
    DWORD * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PathName

[in] Path-name of the logical drive. The path-name may supply either the logical drive name or a full path-name (directories and file name). The function extracts the logical drive name from the beginning of the path.

Selector

[in] Command selector. The value must be 1.

ItemCounter

[in] Number of elements available in pBuffer array. See also pBuffer parameter.

pBuffer

[in] Array where the data will be written. The array must contain a number of element at least equal ItemCounter. The elements can be either 2 or 4. Declaring 2 parameters the free space will be returned. Declaring 4 parameters the total disk space will be returned too.

The elements of the array will contain the following data.

Buffer[0] = low value of the free disk space

Buffer[1] = high value of the free disk space

Buffer[2] = low value of the total disk space

Buffer[3] = high value of the total disk space

To calculate the free space (or the total space) you can use a dummy `double` variable applying the following formula:

```
double Space = (double)LowValue +  
                ((double)HighValue * (double)0x100000000);
```

pErrClass

[out] Pointer to the variable where the class of the error, if any, will be written.

pErrNum

[out] Pointer to the variable where the number of the error, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred. For error codes (Class and Number), see the chapter on "Error Management".



CndexLinkUser DLL documentation Interface with Cndex Server

Functions for long file name management

These functions can be called only after creating the Cndex server (see `ConnectServer_C`), having opened a communication session (see `OpenSession_C`) and having made sure that the CNC is in RUN mode (see `BootPhaseEnquiry_C`).

On a Series 10 system a file with a "long" name (i.e. a name longer than 8 characters) can be created only through the Part Program interface of the CNC. The system provides four logical directories to store files created with the interface into, named PROGRAM, USER, SYSTEM, and OEM respectively. The CNC uses a proprietary method to manage long file names, making it impossible to transfer a file with long name from a PC using a remote disk connection.

The functions described in this paragraph allow the creation and deletion of files with long names from a remote PC, just as if using the Part Program interface of the CNC.

When creating a file with the CNC interface, the file is actually saved as a DOS file with an automatically generated name. Each file with a long name is actually saved as a DOS file, with an automatically generated name, in the UPP directory of the F drive. Using a cross-reference table the system links the DOS file name to the long file name displayed by the CNC interface.

The `PPInsertName_C` function creates a new entry in the cross-reference table. The `PPDeleteName_C` function deletes a previously created entry from the table.

The `PPGetLogicalName_C` returns the logical (or "long") file name, once the physical name of the DOS file is known. The `PPGetPhysicalName_C` function returns the physical file name when the logical name is known.

The `PPGetLogicalDir_C` function returns the identifier of the logical directory containing the file. For directory identifiers see the "Description of the structures and definitions" paragraph.

The `PPUpdate_C` function updates the file information displayed by the CNC interface. In general the only data that require updating are SIZE, DATE, and TIME.



CndexLinkUser DLL documentation Interface with Cndex Server

PPInsertName_C

Creates an entry in the cross-reference table of long file names. The CNC automatically generates the physical file name and creates the zero-length file in the UPP directory of the F drive.

```
WORD PPInsertName_C (  
    WORD    UserSession,  
    LPSTR    LogicalName,  
    LPSTR    LogicalExt,  
    WORD    LogicalDir,  
    LPSTR    PhysicalName,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LogicalName

[in] Null-terminated string containing the logical file name, without extension. Maximum length 49 characters, including terminator.

LogicalExt

[in] Null-terminated string containing the extension of the logical file name. Maximum length 4 characters, including terminator.

LogicalDir

[in] Identifier of the logical directory to store the file into.

PhysicalName

[out] Null-terminated string containing the physical file name generated by the CNC. The physical file name has no extension. Maximum length 4 characters, including terminator.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

PPDeleteName_C



CndexLinkUser DLL documentation Interface with Cndex Server

PPDeleteName_C

Deletes a previously created entry from the cross-reference table. Requires the logical file name, the extension and the logical directory identifier. The CNC automatically deletes the physical file as well.

```
WORD PPDeleteName_C (  
    WORD    UserSession,  
    LPSTR    LogicalName,  
    LPSTR    LogicalExt,  
    WORD    LogicalDir,  
    WORD    * NumberOfNamesDeleted,  
    DWORD    * pErrClass,  
    DWORD    * pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LogicalName

[in] Null-terminated string containing the logical file name, without extension. Maximum length 49 characters, including terminator.

LogicalExt

[in] Null-terminated string containing the extension of the logical file name. Maximum length 4 characters, including terminator.

LogicalDir

[in] Identifier of the logical directory containing the file.

NumberOfNamesDeleted

[out] number of entries actually deleted. This function does not allow wildcards, therefore this field can have a value of either 0 or 1.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

PPInsertName_C



CndexLinkUser DLL documentation Interface with Cndex Server

PPGetPLogicalName_C

Returns the logical file name and extension. Requires the physical file name.

```
WORD PPGetLogicalName_C (  
    WORD    UserSession,  
    LPSTR    PhysicalName,  
    LPSTR    LogicalName,  
    LPSTR    LogicalExt,  
    WORD    LogicalDir,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

PhysicalName

[in] Null-terminated string containing the physical file name generated by the CNC. The physical file name has no extension. Maximum length 4 characters, including terminator.

LogicalName

[out] Null-terminated string containing the logical file name, without extension. Maximum length 49 characters, including terminator.

LogicalExt

[out] Null-terminated string containing the extension of the logical file name. Maximum length 4 characters, including terminator.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

PPGetPhysicalName_C, PPGetLogicalDir_C



CndexLinkUser DLL documentation Interface with Cndex Server

PPGetPhysicalName_C

Returns the physical file name. Requires the logical file name, extension, and the logical directory identifier.

```
WORD PPGetPhysicalName_C (  
    WORD    UserSession,  
    LPSTR    LogicalName,  
    LPSTR    LogicalExt,  
    WORD     LogicalDir,  
    LPSTR    PhysicalName,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LogicalName

[in] Null-terminated string containing the logical file name, without extension. Maximum length 49 characters, including terminator.

LogicalExt

[in] Null-terminated string containing the extension of the logical file name. Maximum length 4 characters, including terminator.

LogicalDir

[in] Identifier of the logical directory containing the file.

PhysicalName

[out] Null-terminated string containing the physical file name generated by the CNC. The physical file name has no extension. Maximum length 4 characters, including terminator.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

PPGetLogicalName_C, PPGetLogicalDir_C



CndexLinkUser DLL documentation Interface with Cndex Server

PPGetLogicalDir_C

Returns the identifier of the logical directory containing the file. Requires the logical file name and extension.

```
WORD PPGetLogicalDir_C (  
    WORD    UserSession,  
    LPSTR   LogicalName,  
    LPSTR   LogicalExt,  
    WORD    *LogicalDir,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LogicalName

[in] Null-terminated string containing the logical file name, without extension. Maximum length 49 characters, including terminator.

LogicalExt

[in] Null-terminated string containing the extension of the logical file name. Maximum length 4 characters, including terminator.

LogicalDir

[out] Pointer to the identifier of the logical directory containing the file.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

PPGetLogicalName_C, PPGetPhysicalName_C



CndexLinkUser DLL documentation Interface with Cndex Server

PPUpdate_C

Updates the file information displayed by the CNC interface. An update is usually required after the file has been modified from outside the Part Program interface.

```
WORD PPUpdate_C (  
    WORD    UserSession,  
    LPSTR    LogicalName,  
    LPSTR    LogicalExt,  
    WORD     LogicalDir,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

LogicalName

[in] Null-terminated string containing the logical file name, without extension. Maximum length 49 characters, including terminator.

LogicalExt

[in] Null-terminated string containing the extension of the logical file name. Maximum length 4 characters, including terminator.

LogicalDir

[in] Identifier of the logical directory containing the file.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

PPInsertName_C, PPDeleteName_C



CndexLinkUser DLL documentation Interface with Cndex Server

Functions for management of CANOPEN HILSCHER

This paragraph contains the functions to manage the CANOpen BUS devices connected with Hilsher master card. For more information please see the WinPlus Library manual, chapter 10.



CndexLinkUser DLL documentation Interface with Cndex Server

CANInit_C

This function is used to set the refresh time of the data available to the logic, i.e., the data transmitted and received through PDO (Process Data Object).

Moreover, it enables you to inform the board as to the memory mapping of reception and transmission data, as well as the diagnostic data about the master board and the nodes.

This function must be called before any other operation to be performed on BUS CANopen nodes. **The system starts refreshing the Input/Output data only after the CAN_INIT function is called.**

```
WORD CANInit_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    WORD    TickNr,  
    WORD    Inp1,  
    WORD    Inp2,  
    WORD    Out1,  
    WORD    Out2,  
    WORD    OfftDiagn,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

TickNr

[in] Indicates the data refresh time. The time must be set as a multiple of 10ms. If the value set is not a multiple of 10ms, the system rounds it up to the nearest higher multiple (E.g., the value set by the logic is 13ms · the system sets a value of 20ms).

Inp1

Inp2

[in] These indicate the first and last byte relating to the input data, respectively (range: 0-1023). If you want the input offsets to be remapped, set both parameters to **zero**; map information is stored in the [INPUT] section of the *.RMP file.

Out1

Out2

[in] These indicate the first and last byte relating to the output data, respectively (range: 0-1023). If you want the output offsets to be remapped, set both parameters on **zero**; map information is stored in the [OUTPUT] section of the *.RMP file.

OffsDiagn

[in] Indicates the first byte of the diagnostics buffer. This buffer is allocated by the system inside the input buffer. The status of the master board and the status of the nodes is stored starting from the byte specified and over a length of 48 bytes. For a description of the data stored in the diagnostics buffer, see the description given in the WinPLUS application manual.



CndexLinkUser DLL documentation Interface with Cndex Server

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function completed successfully
307	Initialisation already performed
308	The input range, or the diagnostics range, overlaps the input range of another CANopen master board present in the system
309	The input range, or the diagnostics range, overlaps the diagnostics range of another CANopen master board present in the system
310	The output range overlaps the output range of another CANopen master board present in the system
314	Remapping file not found
400	Invalid master board number
405	Negative offset value
406	Inputs offset out of range
407	First input byte greater than last input byte
408	Outputs offset out of range
409	First output byte greater than last output byte
410	Diagnostics buffer offset out of range
411	Diagnostics buffer overlaps the input range

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANBoard_C, CANNMT_C, CANSync_C, CANReadSDO_C, CANWriteSDO_C, CANGetEmergency_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANBoard_C

The function sends a command to the master board identified by the **DevNum** parameter.

```
WORD CANBoard_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    WORD    Cmd,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

Cmd

[in] Indicates type of command to be sent. Possible values include:

Values	Description
1	Master board RESET
2	CANopen communication START
3	CANopen communication STOP

RESET command initialises the DualPort of the master board with the parameters stored in the FLASH memory of the device. A RESET operation may take up to 10 seconds and it stops all CANopen communications, and hence the remote nodes may change to WatchDog status.

To reactivate the communications at the end of the RESET command, you must always give the **START** command.

The START command restores the CANopen communications and executes the entire "Node Bootup" sequence set by means of the SyCon configurator.

The **STOP** command stops all CANopen activities on the BUS.

In STOP status and during a RESET, remote node status can change to WatchDog.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.



CndexLinkUser DLL documentation Interface with Cndex Server

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function completed successfully
104	Invalid command type
105	Board reset underway
400	Invalid master board number
Altro	See Appendix B of WinPLUS application manual

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANInit_C, CANNMT_C, CANSync_C, CANReadSDO_C, CANWriteSDO_C, CANGetEmergency_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANSync_C

This function sends SYNC commands to the entire network. This command is used to actualise data to/from analog modules. **This function can only be executed in WAIT mode.**

```
WORD CANSync_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    WORD    Timeout,  
    BYTE    Sync,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

Timeout

[in] Timeout in ms (0 = no timeout).

Sync

[in] Function execution mode (0 = NOWAIT, 1 = WAIT).

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function completed successfully
105	Master board reset underway
106	CAN chip of master board in OFFLINE status
107	CAN chip of master board in STOP status
108	CAN chip of master board in CLEAR status
306	Channel not initialised yet
400	Invalid master board number
404	NO WAIT mode not possible

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.



CndexLinkUser DLL documentation Interface with Cndex Server

See also

CANInit_C, CANBoard_C, CANNMT_C, CANReadSDO_C, CANWriteSDO_C, CANGetEmergency_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANNMT_C

This function is used for the execution of network management services (NMT: Network management). Through these services, the nodes can be initialised, started, stopped and reset. This function is uniquely identified by the node identifier (idNode 1-127). For the execution of the service on all the nodes present on the BUS, set idNode = 0.

```
WORD CANNMT_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    BYTE    idNode,  
    WORD    Cmd,  
    WORD    Timeout,  
    BYTE    Sync,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

idNode

[in] Node identifier(0 = Broadcasting).

Cmd

[in] indicates the type of service to be transmitted. Possible values include:

Valori	Description
1	Remote node start
2	Remote node stop
128	Set pre-operational status
129	Remote node start reset
130	Communication reset

Timeout

[in] Timeout in ms (0 = no timeout).

Sync

[in] Function execution mode (0 = NOWAIT, 1 = WAIT).

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:



CndexLinkUser DLL documentation Interface with Cndex Server

Value	Description
0	Function completed successfully
100	Command rejected (Internal commands list full) (only in NO WAIT mode)
101	Command accepted waiting to be transmitted (only in NO WAIT mode)
102	Command transmitted to node and waiting for reply (only in NO WAIT mode)
103	Timeout error
105	Master board reset underway
106	CAN chip of master board in OFFLINE status
107	CAN chip of master board in STOP status
108	CAN chip of master board in CLEAR status
306	Channel not yet initialised
400	Invalid master board number
401	Invalid node identifier
403	Wrong NMT command

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANInit_C, CANBoard_C, CANSync_C, CANReadSDO_C, CANWriteSDO_C, CANGetEmergency_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANReadSDO_C

This function allows reading the information contained in an object of a node. The object is specified through the **Index** and **SubIndex** parameters.

```
WORD CANReadSDO_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    WORD    idNode,  
    WORD    Index,  
    BYTE    SubIndex,  
    WORD    Timeout,  
    BYTE    Sync,  
    BYTE    *Data,  
    WORD    *DataLen,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

idNode

[in] Node identifier(0 = Broadcasting).

Index

[in] Object index.

SubIndex

[in] Object subindex.

Timeout

[in] Timeout in ms (0 = no timeout).

Sync

[in] Function execution mode (0 = NOWAIT, 1 = WAIT).

DataLen

[out] pointer to a word containing the length in bytes of the data block returned by the function.

Data

[out] Pointer to the buffer where the data block returned by the function will be copied.

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum



CndexLinkUser DLL documentation Interface with Cndex Server

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function completed successfully
100	Command rejected (Internal commands list full) (only in WAIT mode)
101	Command accepted by driver (only in WAIT mode)
102	Command transmitted to node (only in WAIT mode)
103	Timeout error
105	Master board reset underway
106	CAN chip of master board in OFFLINE status
107	CAN chip of master board in STOP status
108	CAN chip of master board in CLEAR status
306	Channel not yet initialised
400	Invalid master board number
401	Invalid node identifier
1003	Service has been rejected by node with Abort SDO. The cause might be Index and subIndex parameters not valid, or you don't have the node access rights. Function output data contain the Abort SDO code.
1017	No reply from node or node not present
1019	Node is not in operational status and access is denied. SDO channel is being used by master board for node configuration at start-up stage. Try requesting service again.
1051	Reception buffer limits exceeded.
1053	Reception buffer limits exceeded by fragmented protocol data.
1054	Previous service is still active and has not been confirmed at application.
1057	Sequence error in protocol fragmentation. Request is aborted.
1200	Master board not configured.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANInit_C, CANBoard_C, CANSync_C, CANNMT_C, CANWriteSDO_C, CANGetEmergency_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANWriteSDO_C

This function allows writing the information contained in the **Data** parameter in an object of a node. The object is identified by the **Index** and **SubIndex** parameters.
If the write command is rejected by the node with an **Abort SDO** code, the **Data** parameter contains the Abort SDO error code.

```
WORD CANWriteSDO_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    WORD    idNode,  
    WORD    Index,  
    BYTE    SubIndex,  
    WORD    Timeout,  
    BYTE    Sync,  
    BYTE    *Data,  
    WORD    *DataLen,  
    DWORD    *pErrClass,  
    DWORD    *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

idNode

[in] Node identifier(0 = Broadcasting).

Index

[in] Object index.

SubIndex

[in] Object subindex.

Timeout

[in] Timeout in ms (0 = no timeout).

Sync

[in] Function execution mode (0 = NOWAIT, 1 = WAIT).

DataLen

[in,out] Pointer to a word containing the length in bytes of the data block to be written, or the Abort SDO error code.

Data

[in,out] Pointer to aPuntatore al buffer contenente il blocco dati da scaricare nel nodo, oppure il codice di Abort SDO.



CndexLinkUser DLL documentation Interface with Cndex Server

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function completed successfully
100	Command rejected (Internal commands list full) (only in WAIT mode)
101	Command accepted by driver (only in WAIT mode)
102	Command transmitted to node (only in WAIT mode)
103	Timeout error
105	Master board reset underway
106	CAN chip of master board in OFFLINE status
107	CAN chip of master board in STOP status
108	CAN chip of master board in CLEAR status
306	Channel not initialised yet
400	Invalid master board number
401	Invalid node identifier
1003	Service has been rejected by node with Abort SDO. The cause might be Index and subIndex parameters not valid, or you don't have the node access rights. Function output data contain the Abort SDO code.
1017	No reply from node or node not present
1019	Node not in operational status. SDO channel is being used by master board for node configuration at start-up stage. Try requesting service again.
1051	Reception buffer limits exceeded.
1053	Reception buffer limits exceeded by fragmented protocol data.
1054	Previous service is still active and has not been confirmed at application.
1057	Sequence error in protocol fragmentation. Request is aborted.
1200	Master board not configured.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANInit_C, CANBoard_C, CANSync_C, CANNMT_C, CANReadSDO_C, CANGetEmergency_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANGetEmergency_C

This function gets the emergency data relating to a node from the master board internal buffer. The function can be called after checking for emergency nodes. The check is performed on the diagnostic buffer starting from the OFFSETDIAGN+32 byte over a length of 16 bytes.

```
WORD CANGetEmergency_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    WORD    idNode,  
    WORD    Timeout,  
    BYTE    Sync,  
    WORD    *NodeStatus,  
    WORD    *AddInfo,  
    WORD    *ProfNum,  
    short   *NodeStateNG,  
    short   *ActualErr,  
    short   *EmcyLen,  
    EMCY_TYPE *EmcyData  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

idNode

[in] Node identifier. A zero value (Broadcasting) is not allowed.

Timeout

[in] Timeout in ms (0 = no timeout).

Sync

[in] Function execution mode (0 = NOWAIT, 1 = WAIT).

NodeStatus

[out] Node status mask. Flags are mapped as follows:

Bit	Description
0	Node does not respond.
1	Buffer limits exceeded by number of emergencies.
2	Difference between master board and node configuration data.
3	WatchDog active on node.
4-7	Reserved



CndexLinkUser DLL documentation Interface with Cndex Server

AddInfo

[out] Information contained in the 1000H object.

ProfNum

[out] Profile number of the 1000H node.

NodeStateNG

[out] Node status. The values are listed in the table below:

Value	Description
1	Node not connected
2	Node being connected
3	Node in preparation
4	Node prepared
5	Node in operational status
127	Node in pre-operational status

ActualErr

[out] Node error code. The values it may assume are listed in the table below, a possible solution is given in brackets:

Value	Description
30	WatchDog error (Make sure node is connected)
31	Node status has changed and node is no longer operational (node reset)
32	Wrong sequence in WatchDog management (node reset)
33	No reply to a PDO (make sure node is enabled for PDOs).
34	No reply from node during its configuration (make sure node is connected and operational).
35	Profile number specified in the configuration not the same as node profile number (check profile number).
36	Device type specified in the configuration not the same as node device type (check services supported by the node).
37	Reply has been received from unknown SDO (node not compatible with CiA specifications).
38	Length of message received by an SDO is not 8 (node not compatible with CiA specifications).
39	Node is not managed by master board, node in STOP state.

EmcyLen

[out] Number of valid messages contained in parameter *EmcyData*.

EmcyData

[out] Buffer of the emergencies [0..4].



CndexLinkUser DLL documentation Interface with Cndex Server

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function completed successfully
306	Channel not initialised yet
400	Invalid master board number
401	Invalid node identifier
100	Command rejected (Internal commands list full) (only in WAIT mode)
101	Command accepted by driver (only in WAIT mode)
102	Command transmitted to node (only in WAIT mode)
103	Timeout error
105	Master board reset underway
106	CAN chip of master board in OFFLINE status
107	CAN chip of master board in STOP status
108	CAN chip of master board in CLEAR status
1161	Invalid node identifier

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANInit_C, CANBoard_C, CANNMT_C, CANSync_C, CANReadSDO_C, CANWriteSDO_C, CANConsoleCfg_C.



CndexLinkUser DLL documentation Interface with Cndex Server

CANConsoleCfg_C

This function allows the configuration of a WinMedia console connected to an intelligent CANopen master board.

```
WORD CANConsoleCfg_C (  
    WORD    UserSession,  
    WORD    DevNum,  
    BYTE    idNode,  
    WORD    InpOffs,  
    WORD    OutOffs,  
    WORD    Timeout,  
    BYTE    Sync,  
    DWORD   *pErrClass,  
    DWORD   *pErrNum  
);
```

Parameters

UserSession

[in] Identifier of the communication session with the numerical control.

DevNum

[in] CANopen master board number.

idNode

[in] Console identifier.

InpOffs

[in] Input offset in bytes.

OutOffs

[in] Output offset in bytes.

Timeout

[in] Timeout in ms (0 = no timeout).

Sync

[in] Function execution mode (0 = NOWAIT, 1 = WAIT).

pErrClass

[out] Pointer to the variable where the error class, if any, will be written.

pErrNum

[out] Pointer to the variable where the error number, if any, will be written. Class 4 errors are coded as follows:

Value	Description
0	Function executed correctly
2	Console already configured
100	Command rejected (internal command list full)



CndexLinkUser DLL documentation Interface with Cndex Server

Value	Description
101	Command accepted by the driver (NO WAIT mode only)
102	Command sent to the node (NO WAIT mode only)
103	Timeout error
105	Master board reset in progress
106	Master board CAN chip in OFFLINE status
107	Master board CAN chip in STOP status
108	Master board CAN chip in CLEAR status
306	Channel not yet initialised
400	Invalid master board number
401	Invalid node identifier
1000+X o 2000+X	Led configuration has not been executed correctly. Run the function again.

Return value

Different from zero if the function has been successfully executed, zero if an error has occurred.

See also

CANInit_C, CANBoard_C, CANNMT_C, CANSync_C, CANReadSDO_C, CANWriteSDO_C, CANGetEmergency_C.



CndexLinkUser DLL documentation Interface with Cndex Server

Command line tools

Overview

WinNbiCmd.exe is a console application which implements a subset of the functions of BootController, ODM and Security. The executable is deployed into the WinNBI installation directory when the product is installed. Each operation is identified by a command keyword which is passed to the executable as a command line argument.

Sintassi

Usage syntax is as follows:

```
<executable name> <command> <options>
```

The command keyword identifies the operation, while the type and number of options depends on the requested operation.

Options are passed to the executable from the command line using the following syntax:

```
<option identifier> [[<value>] [<value>] ...]
```

Option identifiers use the format -<letter>. Identifiers are case sensitive.

For example, the CNC name can be inserted with the following syntax:

```
-c <CNC name or IP address>
```

Like this:

```
-c 172.20.2.14
```

The same option name can have different meanings when used with different commands.



CndexLinkUser DLL documentation Interface with Cndex Server

Commands

As of today, the following commands are available:

<code>backupcnc</code>	Creates a backup of one or more data areas of the CNC. The CNC must be in SETUP mode.
<code>restorecnc</code>	Restores one or more data areas of the CNC from a previously created backup. The CNC must be in SETUP mode.
<code>installcnc</code>	Installs a new software release on the CNC. The CNC must be in SETUP mode.
<code>rebootcnc</code>	Reboots the CNC. The boot mode is required as a command line argument. CNC reboot may be either synchronous or asynchronous. In synchronous mode WinNbiCmd.exe keeps running until the CNC reaches the required mode, or until the timeout expires if specified. To activate synchronous mode use the <code>-w</code> option. In asynchronous mode (default) the executable terminates as soon as the reboot command is sent to the CNC. <i>Note: in OPENcontrol versions prior to 3.4.3, when the CNC is in SETUP mode the reboot command may generate an error. Rebooting with BootController is recommended in this case.</i>
<code>activateamp</code>	Makes an AMP configuration active. The active configuration is loaded at next CNC reboot. No check is performed to ensure that the specified AMP slot actually contains an AMP database, and that the AMP is compiled.
<code>importamp</code>	Imports an AMP database from a .zpo file to the specified AMP slot in a CNC, or to the offline library.
<code>exportamp</code>	Imports an AMP database from the specified AMP slot in a CNC, or from the offline library, to a .zpo file.
<code>buildamp</code>	Compiles an AMP configuration. If the specified AMP slot does not contain an AMP database the command returns an error.

List of commands and related options:

```
backupcnc      -o <output file path>
               -c <cncname>
               [-a <SYSTEM OEM USER DUALPORT ALL default=ALL>]
restorecnc     -i <input file path>
               -c <cncname>
               [-a <SYSTEM OEM USER DUAL PORT ALL default=ALL>]
installcnc     -i <input file path>
               -c <cncname>
rebootcnc      -m <RUNNING|EMERGENCY|SETUP|SERVICE>
               -c <cncname>
               [-w]
               [-t <timeout>]
activateamp    -s <0-9>
               -c <cncname>
importamp      -i <input file path>
               [-n <offline amp name> | -s <0-9> -c <cncname>]
               [-u]
exportamp      -o <output file path>
               [-n <offline amp name> | -s <0-9> -c <cncname>]
               -s <0-9>
               -c <cncname>
               [-u]
```



CndexLinkUser DLL documentation Interface with Cndex Server

Opzioni

- `-c <cncname>`
network name or IP address of the CNC.
- `-i <input file path>`
full path of the input file.
- `-o <output file path>`
full path of the output file.
- `-a <SYSTEM OEM USER DP ALL default=ALL>`
list of the CNC permanent data storage areas, separated by spaces.
'ALL' indicates all areas.
- `-m <RUNNING|EMERGENCY|SETUP|SERVICE>`
CNC boot mode.
- `-w`
wait until requested command completes. Only available for commands that require operations to be executed by the CNC, such as CNC reboot.
- `-t <timeout>`
command execution timeout in milliseconds. When the timeout expires the command is aborted. if the timeout value is 0 or the option is not present the command execution lasts until completion.
- `-s <0-9>`
AMP slot number in the CNC. Must be used in conjunction with the `-c` option.
- `-n <offline amp name>`
Name of the AMP configuration in the ODM offline library.
- `-u`
update AMP configuration to the latest version if necessary.

Examples

The following paragraph lists some examples of use of WinNbiCmd.exe.

Backup of a CNC (system and user areas)

```
winnbicmd backupcnc -c 172.20.2.14 -a SYSTEM USER -o c:\backup\mybackup
```

Restore of a CNC (system and oem areas)

```
winnbicmd restorecnc -c 172.20.2.14 -a SYSTEM OEM -i c:\backup\172.20.2.14
```

Installation or update of CNC software:

```
winnbicmd installcnc -c 172.20.2.14 -i c:\backup\172.20.2.14
```

Reboot of the CNC in Emergency mode:

```
winnbicmd rebootcnc -c 172.20.2.14 -m EMERGENCY
```

Reboot of the CNC in Emergency mode with synchronous boot option:

```
winnbicmd rebootcnc -c 172.20.2.14 -m EMERGENCY -w
```

Import of an AMP database into the offline library with name "MyAmpConfig":

```
winnbicmd importamp -n MyAmpConfig -i c:\amp\ampcfg.zpo
```

Import of an AMP database into AMP slot #2 of a CNC:

```
winnbicmd importamp -c 172.20.2.14 -s 2 -i c:\amp\ampcfg.zpo
```

Export of an AMP database named "MyAmpConfig" from the offline library to a .zpo file:

```
winnbicmd exportamp -n MyAmpConfig -o c:\amp\ampcfg.zpo
```

Export of an AMP database from slot #0 of a CNC to a .zpo file:



CndexLinkUser DLL documentation Interface with Cndex Server

```
winnbicmd exportamp -c 172.20.2.14 -s 0 -o c:\amp\ampcfg.zpo
```

Building of an AMP configuration contained in slot #0 of a CNC:

```
winnbicmd buildamp -c 172.20.2.14 -s 0
```

Activation of an AMP configuration contained in slot #0 of a CNC:

```
winnbicmd activateamp -c 172.20.2.14 -s 0
```



CndexLinkUser DLL documentation Interface with Cndex Server

Description of the structures and definitions

The following structures and definitions are given in the include file of the DLL (CndexLinkUser.h and CndexLink.bas).

```
#define ON 1
#define OFF 0

// Process variable classes
#define E_VAR_CLASS 1 // E variable identification code
#define SN_VAR_CLASS 2 // SN variable identification code
#define SC_VAR_CLASS 255 // SC variable identification code

// Process Mode selected
#define MDI 1 // MDI Mode (Manual Data Input)
#define AUTO 2 // Automatic Mode
#define SEMI 3 // Semi-Automatic (Block-Block) Mode
#define MANJOG 4 // Continuous Manual Mode
#define INCJOG 5 // Incremental Manual Mode
#define PROFILE 6 // Return to Profile Mode
#define HOME 7 // Axis Reference Mode

// Process status
#define IDLE 1
#define CYCLE 2
#define HOLDA 3
#define RUNH 4
#define HRUN 5
#define ERRO 6
#define WAIT 7
#define RESET 8
#define EMERG 9
#define INPUT 10

// Process Sub-Status
#define MAS 6
#define MBR 4

// Maximum number of axes in a Process
#define NUM_ELEM_SEL_AXI 9

// Axis Position type selector
#define PROGPOS 1 // Programmed position
#define INTPOS 2 // Interpolated position
#define TRANSDPOS 3 // Transducer position
#define ERRPOS 4 // Following error
// 5 // Distance To Go
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
#define MAIN_PROGR_NAME_LEN    55

// Logic variables identifier
// Boolean variables
#define I_CODE    0    // I (Input) variables identifier code
#define O_CODE    1    // O (Output) variables identifier code
#define M_CODE    2    // M Boolean variables identifier code
#define G_CODE    3    // G Boolean variables identifier code
#define S_CODE    4    // S Boolean variables identifier code
#define P_CODE    7    // P Boolean variables identifier code
#define U_CODE    8    // U Boolean variables identifier code
// WORD variables
#define MW_CODE    20   // MW word variables identifier code
#define GW_CODE    21   // GW word variables identifier code
#define SW_CODE    22   // SW word variables identifier code
#define PW_CODE    62   // PW word variables identifier code
#define UW_CODE    63   // UW word variables identifier code
// DOUBLE variables
#define MD_CODE    40   // MD double variables identifier code
#define GD_CODE    41   // GD double variables identifier code
#define PD_CODE    43   // PD double variables identifier code
#define UD_CODE    44   // UD double variables identifier code

// Tables identifiers
#define AXIS_TABLE_ID    1 // Axis Table identifier
#define TOOL_TABLE_ID    2 // Tool Table identifier
#define OFFSET_TABLE_ID  3 // Offset Table identifier
#define USER_TABLE_ID    4 // User Table identifier

//Logical directory identifiers
#define PPDIR_PROGRAM    0 //PROGRAM Directory
#define PPDIR_USER        1 //USER Directory
#define PPDIR_SYSTEM      2 //SYSTEM Directory
#define PPDIR_OEM          3 //OEM Directory

#pragma pack(push, VB_requests_pack_4, 4)

struct GETINTDATA_C4
{
    BYTE    AxisName;    // ASCII axis name
    BYTE    mode;        //
    double  position;    // Current position
    double  TotalOffset; // Total Offset
};
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
struct PROCDATA_C4
{
    WORD    Mode;           // Selected Mode
    WORD    Status;         // Process Status
    WORD    SubStatus;      // Process Sub-Status
};

struct GETINFO1DATA_C4
{
    BYTE    status;         // Process Status
    BYTE    substatus;      // Process Sub-Status
    BYTE    mode_select;    // Selected Mode
    BYTE    jog_dir;        // JOG direction (1 = negative, 0 = positive)
    WORD    speed_ov;       // Spindle Speed Override (multiplied by 100)
    WORD    feed_ov;        // Feed Rate Override (multiplied by 100)
    WORD    rap_feed_ov;    // Rapid Feed Override (multiplied by 100)
    WORD    man_feed_ov;    // Manual Feed Override (multiplied by 100)
    double   jog_incr;      // Value of increment for
// Incremental Manual Mode
    BYTE    num_ax_sel;     // Number of axes selected
    BYTE    ax_sel[NUM_ELEM_SEL_AXI]; // Array of axes selected
    double   actual_tool;    // Current tool loaded
    double   progr_tool;     // Current tool programmed
    WORD     actual_tool_off; // Current offset number loaded
    WORD     progr_tool_off;  // Current offset number programmed
    double   real_speed;     // Real spindle speed
    double   progr_speed;    // Programmed spindle speed
    double   progr_feed;     // Programmed feed for continuous movement
    double   rapid_feed;     // Rapid feed
    double   real_feed;      // Real feed
    WORD     feed_mis_unit;   // Feed measuring unit:
                                //0 = inch without G95
                                //1 = mm without G95
                                //2 = inch with G95
                                //3 = mm with G95
    BYTE     main_progr_name[MAIN_PROGR_NAME_LEN]; // Main Part Program active
    BYTE     dry_run;        // 1 = dry run mode active
    BYTE     rapid_override; // 1 = Limited rapid feeds active
    BYTE     disable_slashed_blk; // 1 = Slashed Blocks deactivation active
    BYTE     optional_stop;  // 1 = Forced stop on M00 active
    BYTE     force_rapid_feed; // 1 = Forced rapid feed active
    BYTE     auto_jog_ret;   // 1 = Automatic return to profile and
                                // Automatic Incremental Movements active
    BYTE     block_retrace;  // 1 = multi block retrace active
    WORD     last_nc_error;   // Las Process Error. 0 = no Error
    BYTE     free[22];       // reserved fields
};
```




CndexLinkUser DLL documentation Interface with Cndex Server

```
struct GETINFO2DATA_C4
{
    WORD        StatusWORD; //status word for changed fields
    double       Urp;        //plane rotation angle value
    double       ActTool;    //actual tool number
    WORD        ActOffset;   //actual tool offset number
    double       ProgTool;   //programmed tool number
    WORD        ProgOffset;  //programmed tool offset number
    WORD        M_Status;    //status for highlighted M codes
    WORD        M_Value[16]; //value of the 16 M codes
    BYTE        Ax1Name;     //axis 1 name
    double       Ax1Offset;  //axis 1 tool offset value
    BYTE        Ax2Name;     //axis 2 name
    double       Ax2Offset;  //axis 1 tool offset value
    double       Radius;     //radius of the active tool
};

struct MARKER_INFO_C4
{
    short        Number; // Number of active part-programs
    unsigned long LineNum[MAX_LEVEL]; // Line number of active part-programs
    unsigned long NumBlks[MAX_LEVEL]; // Block number of active part-programs
    char         MarkerName[MAX_LEVEL][LABEL_LEN+1]; // Marker of active
                                                    // part-programs
    unsigned char PPname[MAX_LEVEL][MAX_LEN_BLK+5]; // Name of active
                                                    // part-programs
    unsigned long breakVal; // Value of break-point variable ($BRK)
    unsigned long free[8]; // RESERVED
};

struct GETBLKNUMDATA_C4
{
    WORD        ppActNum; // Number of active programs
    unsigned long MainActBlk; // Block number of main part-program (if active).
    unsigned long Sbr1ActBlk; // Block number of subroutine 1 (if active).
    unsigned long Sbr2ActBlk; // Block number of subroutine 2 (if active).
    unsigned long Sbr3ActBlk; // Block number of subroutine 3 (if active).
    unsigned long Sbr4ActBlk; // Block number of subroutine 4 (if active).
};

// Series10 axis table descriptor
struct AXIS_TABLE_C4
{
    WORD        ax_owner; // Code of axis owner environment
    WORD        ax_name;  // ASCII axis name
    double       ax_orig;  // value of origin in use
    double       free1;    // reserved field
    double       ax_ofg92; // offset G92
    double       ax_toff;  // Current tool offset
    double       free2;    // reserved field
    double       ax_offset; // total offset value
    double       orig1;    // origin 1
    double       orig2;    // origin 2
    double       orig3;    // origin 3
    double       orig4;    // origin 4
}
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
double    orig5;        // origin 5
double    orig6;        // origin 6
double    orig7;        // origin 7
double    orig8;        // origin 8
double    orig9;        // origin 9
double    orig10;       // origin 10
WORD      reserved[2]; // reserved field
};

// Series10 tool table descriptor
struct TOOL_TABLE_C4
{
    double    tcode;        // tool number
    WORD      pocket;       // position in tool magazine
    WORD      tfamcol;      // tool family code
    WORD      tclass;       // tool class
    WORD      tstatus;      // status word
    WORD      tcntrl;       // control word
    double    maxtime;      // total tool life
    double    remtime;      // residual tool life
    double    tuser1;       // user parameter 1
    double    tuser2;       // user parameter 2
    double    tuser3;       // user parameter 3
    double    tuser4;       // user parameter 4
    WORD      tolnfr;       // offset number associated with tool
};

// Series10 offset table descriptor
struct OFFSET_TABLE_C4
{
    double    tactl1;       // current tool length 1
    double    tcmaxl1;      // maximum variation in tool length 1
    double    tcactl1;      // current variation in tool length 1
    double    tactl2;       // current tool length 2
    double    tcmaxl2;      // maximum variation in tool length 2
    double    tcactl2;      // current variation in tool length 2
    double    tdiameter;    // diameter
    double    tcacdiam;     // current diameter variation
    WORD      torient;      // tool tip orientation
};

// Series10 user table descriptor
struct USER_TABLE_C4
{
    double    user1;        // user variable 1
    double    user2;        // user variable 2
    double    user3;        // user variable 3
    double    user4;        // user variable 4
};

// OPENcontrol tool table descriptor
struct TOOL_TABLE_II_C4
{
    char      ToolName [TOOL_ASCII_LEN]; // Tool name
    WORD      Status;           // Tool status
    WORD      LifeType;         // Type of life management
};
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
double    MaxLife;                // Starting tool life
double    RemLife;                // Remaining tool life
WORD      OffsNum;                // Number of default offsets
WORD      ExternType;             // Type of external management
                                   // (..tool magazine)
WORD      ExternInd;              // Index for external management (..)
double    Duser [NUM_DUSER];      // User double parameters
short     Suser [NUM_SUSER];      // User short parameters
WCHAR     Descr [DESCR_LEN];      // Null terminated comment
};

struct OFFSET_VAL_II_C4
{
    double    ValOrig;             // Offset initial value
    double    MaxChangeVal;        // Maximum allowed difference
    double    ActChangeVal;        // Current difference
};

// OPENcontrol offset table descriptor
struct OFFSET_TABLE_II_C4
{
    struct OFFSET_VAL_II_C4 LenVal [OFFS_AX];    // Length
    struct OFFSET_VAL_II_C4 DiaVal [DIAM_AX];    // Diametres / Toroidal radius
    WORD      Orient;               // Orientation
    WORD      ExternType;           // Type of external management (..)
    WORD      ExternInd;           // Index for external management (..)
    double    Duser [NUM_DUSER];    // User double parameters
    short     Suser [NUM_SUSER];    // User short parameters
    WCHAR     Descr [DESCR_LEN];    // Null terminated comment
};

// OPENcontrol origin table descriptor
struct ORIGIN_TABLE_II_C4
{
    double    AxisVal [MAX_NUM_AX_XTEND];    // Origin value for each axis
    WORD      ExternType; // Type of external management (..)
    WORD      ExternInd; // Index for external management (..)
    WCHAR     Descr [DESCR_LEN]; // Null terminated comment
};

// OPENcontrol user table descriptor
struct USER_TABLE_II_C4
{
    double    UserVal [4];          // user variable (1 - 4)
};

struct MAGAZINE_TABLE_II_C4
{
    WORD      Type;                 // Tipo del magazzino
    WORD      NumPockets;           // Numero di pockets del magazzino
    WORD      RowPockets;           // Numero di pocket per riga (planari)
    WORD      ExternType;           // Tipo gestione esterna (..)
    WORD      ExternInd;            // Indice per gestione esterna (..)
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
double    Duser [10];          // Parametri User double
WORD      Suser [10];          // Parametri user short
WCHAR     Descr [DESCR_LEN];  // Commento con /0
};

struct POCKET_TABLE_II_C4
{
    WORD     Status;            // Pocket free or occupied
    WORD     isRandom;          // assigned to a fixed tool or not
    WORD     Type;              // Pocket type (master, bulk ... )
    WORD     Class;             // Pocket class
    WORD     ToolInd;           // Indice del tool caricato in pocket
    WCHAR     Descr [DESCR_LEN]; // Commento con /0
};

struct ERR_MSG_C4
{
    unsigned char Msg1[40];     // first line of message
    unsigned char Msg2[40];     // second line of message
    unsigned char Msg3[40];     // third line of message
    unsigned char Msg4[40];     // fourth line of message
};

struct PLVARDESC_C4
{
    unsigned short Code;        // Variable identifier code
                                // See "Logic variable identifier"
                                // in this paragraph
    unsigned short Index;       // Variable index
    unsigned short Bit;         // Variable bit index (0 - 15).
                                // This field is used in write mode
                                // only and when Code identifies a Boolean variable.
};

struct MSG_ERROR_C4
{
    DWORD BootID;
    SYSTEMTIME_CNDEX_C4 SystemTime; // - Event time
    DWORD UnIdSeq;                // - Sequence number (since last boot)
    DWORD Code_Err;              // - Error code
    short Process;               // - Process
    char Comando;                // - Command that generated the error
    char SubCom;                 // - Subcommand that generated the error
    BYTE FormatTxt[148];         // - Additional error data
};

struct MSG_EMERGENCY_C4
{
    DWORD BootID;
    SYSTEMTIME_CNDEX_C4 SystemTime; // - Event time
    DWORD UnIdSeq;                // - Sequence number (since last boot)
    DWORD Code_Err;              // - Emergency code
    short Process;               // - Process
    short Proc_Err;              // - Involved processes
};
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
    BYTE    FormatTxt[148];    // - Additional emergency data
};

struct MSG_LOGS_C4
{
    DWORD BootID;
    SYSTEMTIME_CNDEX_C4 SystemTime; // - Event time
    DWORD UnIdSeq;                  // - Sequence number (since last boot)
    DWORD Code_Log;                 // - Log code
    DWORD Err_info;                 // - Log entry
    BYTE    FormatTxt[148];         // - Additional log data
};

struct MSG_ANOMALY_C4
{
    DWORD BootID;
    SYSTEMTIME_CNDEX_C4 SystemTime; // - Event time
    DWORD UnIdSeq;                  // - Sequence number (since last boot)
    DWORD Code_Err;                 // - Error code
    DWORD Linea;                    // - Line
    BYTE    FormatTxt[148];         // - Additional error data
};
```



CndexLinkUser DLL documentation Interface with Cndex Server

```
struct REMAP_DEF_C4
{
    unsigned char VarType;    // 0: bool; 1: byte; 2: word; 3: dword
    unsigned short PhysAddr;  // physical address
    unsigned char PhysBit;    // physical bit
    unsigned short LogicAddr; // logical address
    unsigned char LogicBit;   // logical bit
    unsigned char Mode;       // 0: ignore; 1: copy; 2: set to true;
                             // 3: set to false; 4: negate; 5: set to value

    unsigned long RiseTime;   // rise time (us)
    unsigned long FallTime;   // fall time (us)
    unsigned long Value;      // value
    unsigned long IOMsgId;    // message id
    unsigned char IOClass;    // class

    unsigned long CurrPhysValue; // current physical value
    unsigned long CurrLogicValue; // current logical value
};

struct AX_ORIG_NUM_C4
{
    unsigned char AxisName;    //Axis name
    unsigned char OriginNumber; //Number of active origin
    unsigned short OriginFlag; //Bit mask of origin type:
                             // bit 0 - absolute origin (activated with UAO)
                             // bit 1 - temporary origin (activated with UTO)
                             // bit 2 - incremental origin (activated with UIO)
};
```



CndexLinkUser DLL documentation Interface with Cndex Server

Error Management

In all functions, the last two parameters give the codes of the errors, if any have occurred.

The first (pErrClass) identifies the class to which the error belongs, the second (pErrNum) gives the number of the actual error code.



CndexLinkUser DLL documentation Interface with Cndex Server

Error class

Errors have been grouped into the following classes:

COM error class (class 1)

The COM interface used to communicate with the Cndex server causes the errors belonging to this class.

SERVER error class (class 2)

The errors belonging to this class may be caused by a preliminary series of checks that the Cndex server performs on the parameters of the functions called, or by internal server malfunctioning.

NETBIOS error class (class 3)

The errors belonging to this class are caused by anomalies or errors in the exchange of data between the Cndex server and numerical controls with which communication sessions are active on the local network.

CNC error class (class 4)

The errors belonging to this class are caused by the processing of data by the numerical controls connected to the Cndex server.

FILESYS error class (class 5)

Contains errors generated by logical file system (LogFS) functions.

FILESYS_ERR error class (class 6)

Contains errors generated by the operating system where Cndex is running when a logical file system (LogFS) function encounters an error while accessing the local file system. Please check the System Error Codes in MSDN for a detailed description of each error code.

CNC_BOOT (class 7)

This class contains the errors occurred during the boot phase of the numerical control.

CNDEXLINK_DLL error class (class 9)

These errors are generated by CndexLink.dll.

DLL_INTERFACE error class (class 10)

These errors are generated by the CndexLink user library API, i.e. the functions described in this document.

OPENcontrol error classes (classes from 17 to 62)

Errors in this class range are generated by OPENcontrol software. For a detailed description please check the OPENcontrol user documentation.



CndexLinkUser DLL documentation Interface with Cndex Server

COM error codes (class 1)

This is a list of the most frequent COM errors. For error codes not included in this list, please check the MSDN documentation.

CO_E_SERVER_EXEC_FAILURE (0x80080005)	On Windows 95/98 the server cannot be launched automatically, and has to be executed manually, keeping in mind that when all applications are no longer active, the server is automatically terminated.
RPC_S_SERVER_UNAVAILABLE (0x800706ba)	Verify that the CNC is up and running, and connected to the local area network. Make sure that the value of parameter <enableremoteconnect> in the registry is Y. Add or activate the TCP/IP protocol in the network configuration.
RPC_S_CALL_FAILED (0x800706be)	Verify that the CNC is up and running, and connected to the local area network. Verify that the CNC is reachable by connecting with BootController.
E_ACCESSDENIED (0x80070005)	Verify that DCOM is enabled on the system and that the application initialises the DCOM security with the correct options. For more details please refer to DCOM_WindowsXP_Configuration_English.pdf .
REGDB_E_CLASSNOTREG (0x80040154)	The Cndex server has not been recorded in the Windows registry. Record it with this command, to be executed in a DOS shell: <code>cndex.exe /regserver</code> .
CO_S_NOTALLINTERFACES (0x00080012)	The proxy DLL of the Cndex server has not been registered in the Windows registry. Please register the proxy DLL with this command, to be executed in a DOS shell: <code>regsvr32 cndexps.dll</code> .
RPC_E_DISCONNECTED (0x80010108)	The Cndex server has disconnected from the client. Disconnections are usually due to timeout or network failure.
RPC_E_SERVERFAULT (0x80010105)	The server threw an exception. Please contact PrimaElectro technical support.
RPC_X_NULL_REF_POINTER (0x800706F4)	A null reference pointer was passed to the stub. Verify that all pointer type arguments passed to the function are not null.
RPC_E_CANTCALLOUT_INEXTERNALCALL (0x80010005)	A WM_PAINT message handler contains a call to a Cndex function or to a generic COM function. Remove the function call that generates the error from the message handler.
OR_INVALID_OXID (0x80070776)	This error may occur when the TCP/IP settings of one or more network cards are changed and Windows CE has not yet been rebooted. Reboot Windows CE. If the problem persists contact PrimaElectro technical support.



CndexLinkUser DLL documentation Interface with Cndex Server

Cndex server error codes (class 2)

Memory for dynamic allocations insufficient	1	
Impossible to create synchronisation events	2	
Session aborted and no longer usable	3	
Session not open	4	
Impossible to allocate a channel	5	
Function of a non-existing process requested	6	
Broadcasting command aborted	7	
Buffer supplied by function in which the data supplied by the NC are copied is too small	8	
Session already open	9	
Broadcasting list invalid	10	
Realtime command aborted	11	
Function already active	12	
Function not yet active	13	
Reception thread ended	14	
No reply to command received within allotted time	15	
NC release to which you are connected is not compatible with communications with server	16	
Cookie does not identify any communication session	17	
Realtime thread cannot be created	18	
No more sessions available	19	
Error in symbol acquisition	20	
Internal object instance cannot be created	23	
Broadcasting thread cannot be created	24	
Function cannot be executed in the numerical control boot phase	25	
Parameter wrong	26	
Invalid buffer	27	
Session identifier (UserSession) is invalid (session closed or never opened)	29	
Session identifier is invalid (value out of range) or session has been closed automatically by Cndex server	31	
Error writing registry	32	
Out of memory	33	
Function not available (boot phase is not correct or an error has occurred while loading the minimal SW configuration for running the requested command)	34	
Time-out while waiting for a generic event (used during reboot)	35	
Error when registering current thread as an OPENControl thread	36	
Error when launching or running an external process or application	37	
Function not implemented	38	
DLL not found	39	
A function entry point in a DLL was not found	40	
Error while declaring a shared memory area for data exchange with an application	41	
Unknown error	42	
The index of the active AMP in BootAmp.txt is not valid	43	
Error while reading from the registry	44	
Registry key not present	45	
Error while opening a file	46	
Error while reading data from file	47	
Error while writing data to file	48	
CNC name too long	49	



CndexLinkUser DLL documentation Interface with Cndex Server

NETBIOS network protocol error codes (class 3)

Illegal buffer length	0x0001
Illegal command	0x0003
Command timed out	0x0005
Message incomplete, issue another command	0x0006
Illegal buffer address	0x0007
Session number out of range	0x0008
No resource available	0x0009
Session closed	0x000a
Command canceled	0x000b
Duplicate name	0x000d
Name table full	0x000e
No deletions, name has active sessions	0x000f
Local session table full	0x0011
Remote session table full	0x0012
Illegal name number	0x0013
No callname	0x0014
Cannot put * in NCB_NAME	0x0015
Name in use on remote adapter	0x0016
Name deleted	0x0017
Session ended abnormally	0x0018
Name conflict detected	0x0019
Interface busy, IRET before retrying	0x0021
Too many commands outstanding, retry later	0x0022
Ncb_lana_num field invalid	0x0023
Command completed while cancel occurring	0x0024
Command not valid to cancel	0x0026
Name defined by another local process	0x0030
Environment undefined. RESET required	0x0034
Required OS resources exhausted	0x0035
Max number of applications exceeded	0x0036
No saps available for netbios	0x0037
Requested resources are not available	0x0038
Invalid ncb address or length > segment	0x0039
Invalid NCB DDID	0x003B
Lock of user area failed	0x003C
NETBIOS not loaded	0x003f
System error	0x0040

Errors in bold are those that occur most frequently.

Error " No call name" occurs when the connection with a numerical control is impossible.

Error " Session ended abnormally" occurs when the communication session is lost due to a network connection problem or because the numerical control is turned off or rebootstrapped.



CndexLinkUser DLL documentation Interface with Cndex Server

10 Series error codes (class 4)

Error codes for Real Time functions

Command unknown	0x0101
No channel available	0x0102
Tick requested not multiple of system tick	0x0103
Id channel wrong	0x0104
Data acquisition still underway	0x0105
Channel not configured	0x0106
Error on stop trigger	0x0107
Channel already configured for other types of data	0x0108

Error codes for Dry Run functions

Process not configured	0x0200
Axis not present in process	0x0201
Dry run not configured	0x0202
Dry run already being executed	0x0203
Dry run already in stop status	0x0204

Error codes for Process functions

See Series 10 Programming Manual.



CndexLinkUser DLL documentation Interface with Cndex Server

FILESYS_CLASS error codes (class 5)

ERROR_OPENING_TRANSACTION 1 (0x1)	An error has occurred while opening a transaction with the target for file transfer.
ERROR_CLOSING_TRANSACTION 2 (0x2)	An error has occurred while closing a transaction with the target for file transfer.
WRONG_FILEID 3 (0x3)	Invalid file ID.
WRONG_TRANSACTION 4 (0x4)	The number of transaction is not valid.
ERROR_OPENING_FILE 5 (0x5)	Error opening file.
ERROR_CLOSING_FILE 6 (0x6)	Error closing file.
WILDCARD_NOTALLOWED 7 (0x7)	A file path containing wildcards was passed to a function that does not support wildcards.
ERROR_READING_DATA 8 (0x8)	Error reading data from file.
ERROR_WRITING_DATA 9 (0x9)	Error writing data to file.
DRIVE_NOT_FOUND 10 (0xA)	The logical drive was not found on the target.
WRONG_PATH_NAME 12 (0xC)	The specified file path was not valid.
DRIVE_NAME_ALREADY_EXISTS 13 (0xD)	An attempt was made to create a logical drive on the target with the same name as an existing logical drive.
WRONG_SECURITY_LEVEL 14 (0xE)	The invoked function requires a different security level on the target.
NO_ADD_DRIVE_ON_PC 15 (0xF)	A logical drive cannot be created on a remote PC.
INSUFFICIENT_DISK_SPACE 16 (0x10)	Insufficient disk space to complete the requested operation.
FILE_NOT_FOUND 17 (0x11)	The requested file was not found.



CndexLinkUser DLL documentation Interface with Cndex Server

FILESYS_ERRNUM_CLASS error codes (class 6)

When the physical file system generates an error, Cndex returns a class 6 error. Only logical file system functions return class 6 errors. The error number matches the exact error code returned by the target operating system, whether it is Windows CE or a desktop Windows operating system. Please refer to the [MSDN documentation on system error codes](#).



CndexLinkUser DLL documentation Interface with Cndex Server

CNC_BOOT_ERR_CLASS error codes (class 7)

ERROR_LOADING_IPC_DLL 1 (0x1)	Error while loading the IPC module.
ERROR_LOADING_TABLE_DLL 2 (0x2)	Error while loading tables and system history management module.
ERROR_LOADING_LOADER_DLL 3 (0x3)	Error during boot phase of the numerical control.
ERROR_CNC_SHUT_DOWN 4 (0x4)	
ERROR_CNC_INIT 16 (0x10)	Error loading boot routines of the numerical control.
ERROR_CNC_HALT_BOOT 19 (0x13)	Error connecting to numerical control. The loader module of the numerical control is in error state.



CndexLinkUser DLL documentation Interface with Cndex Server

SERVER_EXCEPTION_CLASS error codes (class 8)

Errors in this class indicate that an exception has occurred during execution of the remote procedure in the CNC. The error number contains the actual error code returned by the Windows CE API GetLastError(). For example, a class 8 number 0xC0000005 error indicates that an access violation exception has been raised during execution of the invoked function in the CNC.



CndexLinkUser DLL documentation Interface with Cndex Server

CNDEXLINK_DLL_ERR_CLASS error codes (class 9)

ERR_INTERFACE_NOT_AVAILABLE 1 (0x1)	The invoked function is not supported by the target system.
ERR_CE_USED_AS_A_NETBIOS_ROUTER 2 (0x2)	An attempt was made to connect to a Series10 target using a Windows CE machine as Cndex server/router. Only system that support NetBEUI can be used for such purpose.
ERR_OPENING_LOCAL_FILE 3 (0x3)	The system encountered an error while opening a local file in a logical file system (LogFS) function call.
ERR_READING_LOCAL_FILE 4 (0x4)	The system encountered an error while reading a local file in a logical file system (LogFS) function call.
ERR_WRITING_LOCAL_FILE 5 (0x5)	The system encountered an error while writing data into a local file in a logical file system (LogFS) function call.
ERR_CLOSING_LOCAL_FILE 6 (0x6)	The system encountered an error while closing a local file in a logical file system (LogFS) function call.
ERR_NO_MEMORY_AVAILABLE 7 (0x7)	The system could not allocate a memory buffer.
ERR_READING_LOCAL_FILE_SIZE 8 (0x8)	The system encountered an error while reading the size of a local file in a logical file system (LogFS) function call.
ERR_INSUFFICIENT_DISK_SPACE 9 (0x9)	The system encountered a disk full error while copying a file in a logical file system (LogFS) function call.
ERR_SYNCHRONIZATION 10 (0xA)	An error occurred while waiting for a worker thread to complete.
ERR_THREAD_CREATION 11 (0xB)	An error occurred while trying to create a worker thread.
ERR_NO_MEMORY 12 (0xC)	An error occurred while trying to create a shared data area for a worker thread
ERR_CNDEX_SERVER_NOT_CREATED 13 (0xD)	The Cndex server associated to the current session ID was not created.
ERR_DURING_SERVER_LOCK 14 (0xE)	An error occurred while trying to lock the Cndex server.
ERR_NETBIOS_NAME_TOO_LONG 15 (0xF)	An attempt was made to connect to a Series10 CNC using a CNC name longer than 15 characters.



CndexLinkUser DLL documentation Interface with Cndex Server

DLL_INTERFACE_ERR_CLASS error codes (class 10)

ERR_SERVER_ALREADY_CREATED 1 (0x1)	The server has been created more than once
ERR_CREATING_SERVER_OBJECT 2 (0x2)	An error has occurred during the creation of the Cndex server
ERR_SERVER_NOT_CREATED 3 (0x3)	A function has been called without having created the Cndex server
ERR_INVALID_PARAMETER 4 (0x4)	One or more function input parameters are not valid
ERR_OPTION_NOT_ENABLED 5 (0x5)	Option A06 "CndexLink communication" for network communications with external applications is not enabled on the CNC you are trying to connect to.
ERR_UAS_DISABLED_WHEN_RCM_ON 6 (0x6)	A Dry Run request cannot be accepted while a Memory Search is active.
ERR_OUT_OF_MEMORY 7 (0x7)	A memory buffer allocation failed.
ERR_BUFFER_TOO_SMALL 8 (0x8)	The data buffer passed to the function is too small to contain the requested data.



CndexLinkUser DLL documentation Interface with Cndex Server

SOAP_INTERFACE_ERR_CLASS error codes (class 11)

Error class 11 contains SOAP communication errors detected on the client side. The source of the error can be on either the client or the server. For example, a SOAP_TCP_ERROR returned by a SOAP function call on the client may indicate that the CNC has rebooted, was switched off, or has encountered an unrecoverable error.

Code	Symbol	Description
0	SOAP_OK	No error
1	SOAP_CLI_FAULT	The service returned a client fault (SOAP 1.2 Sender fault)
2	SOAP_SVR_FAULT	The service returned a server fault (SOAP 1.2 Receiver fault)
3	SOAP_TAG_MISMATCH	An XML element didn't correspond to anything expected
4	SOAP_TYPE	An XML Schema type mismatch
5	SOAP_SYNTAX_ERROR	An XML syntax error occurred on the input
6	SOAP_NO_TAG	Begin of an element expected, but not found
7	SOAP_IOB	Array index out of bounds
8	SOAP_MUSTUNDERSTAND	An element needs to be ignored that need to be understood
9	SOAP_NAMESPACE	Namespace name mismatch (validation error)
10	SOAP_USER_ERROR	User error (reserved for soap.user usage)
11	SOAP_FATAL_ERROR	Internal error
12	SOAP_FAULT	An exception raised by the service
13	SOAP_NO_METHOD	The dispatcher did not find a matching operation for a request
14	SOAP_NO_DATA	No data in HTTP message
15	SOAP_GET_METHOD	HTTP GET operation not handled, see Section 19.10
20	SOAP_EOM	Out of memory
21	SOAP_MOE	Memory overflow/corruption error (DEBUG mode)
23	SOAP_NULL	An element was null, while it is not supposed to be null
24	SOAP_DUPLICATE_ID	Element's ID duplicated (multi-ref encoding)
25	SOAP_MISSING_ID	Element ID missing for an href/ref (multi-ref encoding)
26	SOAP_HREF	Reference to object is incompatible with the object referred to
27	SOAP_UDP_ERROR	Message too large to store in UDP packet
28	SOAP_TCP_ERROR	A connection error occurred
29	SOAP_HTTP_ERROR	An HTTP error occurred
30	SOAP_SSL_ERROR	An SSL error occurred
31	SOAP_ZLIB_ERROR	A Zlib error occurred
32	SOAP_DIME_ERROR	DIME formatting error or DIME size exceeds SOAP_MAXDIMESIZE
33	SOAP_DIME_HREF	DIME attachment has no href from SOAP body (and no DIME callbacks were defined to save the attachment)
34	SOAP_DIME_MISMATCH	DIME version/transmission error
35	SOAP_DIME_END	End of DIME attachments protocol error
36	SOAP_MIME_ERROR	MIME parsing error
37	SOAP_MIME_HREF	MIME attachment has no href from SOAP body error
38	SOAP_MIME_END	End of MIME attachments protocol error
39	SOAP_VERSIONMISMATCH	SOAP version mismatch or no SOAP message
40	SOAP_PLUGIN_ERROR	Failed to register plugin
41	SOAP_DATAENCODINGUNKNOWN	SOAP 1.2 DataEncodingUnknown fault



CndexLinkUser DLL documentation Interface with Cndex Server

Code	Symbol	Description
42	SOAP_REQUIRED	Attributed required validation error
43	SOAP_PROHIBITED	Attributed prohibited validation error
44	SOAP_OCCURS	Element minOccurs/maxOccurs validation error or SOAP_MAXOCCURS exceeded
45	SOAP_LENGTH	Element length validation error or SOAP_MAXLENGTH exceeded
46	SOAP_FD_EXCEEDED	Too many open sockets (for non-win32 systems not supporting poll())
47	SOAP_UTF_ERROR	An UTF-encoded message decoding error occurred
48	SOAP_NTLM_ERROR	An NTLM authentication handshake error occurred
#N/D	SOAP_LEVEL	XML nesting depth level exceeds SOAP_MAXLEVEL
EOF (-1)	SOAP_EOF	Unexpected end of file, no input, or timeout receiving data
EOF (-1)	SOAP_ERR	Error (for internal use)



CndexLinkUser DLL documentation Interface with Cndex Server

OPENcontrol error codes (classes from 17 to 62)

Errors in this class range are generated by OPENcontrol software. For a more detailed descriptions please check the OPENcontrol user documentation.

Classe	ID	Description
IPCEclass	17 (0x11)	IPC error class
CNSEclass	18 (0x12)	Consoles error class
EMGEclass	19 (0x13)	Emergency error class
IOEclass	20 (0x14)	IO error class
PLCEclass	21 (0x15)	PLC error class
GMCEclass	22 (0x16)	GMC error class
CNCEclass	23 (0x17)	CNC error class
OSWEclass	24 (0x18)	OSWire error class
TBLEclass	25 (0x19)	Table error class
GEOEclass	26 (0x1A)	Geometry error class
SPLclass	27 (0x1B)	Spline error class
LODEclass	31 (0x1F)	LOADER error class
SEREclass	32 (0x20)	SERVO error class
SEROSWcls	33 (0x21)	SERVO OSWIRE error class
SERDSlcls	34 (0x22)	SERVO SERCOS error class
SERCANcls	35 (0x23)	SERVO CANOPEN error class
SERSENScls	36 (0x24)	SERVO SENSORE error class
SERCATcls	37 (0x25)	SERVO EtherCAT error class
SERMECcls	38 (0x26)	SERVO Mechatrolink error class
MONEClass	48 (0x30)	MONITORING error class
SLIEClass	49 (0x31)	Serial Line error class
SECEClass	50 (0x32)	Security error class
SERCCClass	51 (0x33)	DLL SERCOS error class
FLDBClass	52 (0x34)	DLL FIELDBUS error class
CANClass	53 (0x35)	DLL CANBUS error class
ECATClass	54 (0x36)	DLL EtherCAT error class
PROFIEClass	55 (0x37)	DLL PROFIBUS error class
XMLEClass	56 (0x38)	DLL XML error class
SOCKEClass	57 (0x39)	DLL Socket error class
CTRLClass	58 (0x3A)	Controls error class
_3DEclass	59 (0x3B)	3D error class
RETAEclass	60 (0x3C)	Retain DB error class
MECHClass	61 (0x3D)	Mechatrolink error class
SOAPClass	62 (0x3E)	SOAP Server error class