Volume

4

AxesBrain

Servizi DCOM



Servizi DCOM

AxesBrain

Servizi DCOM

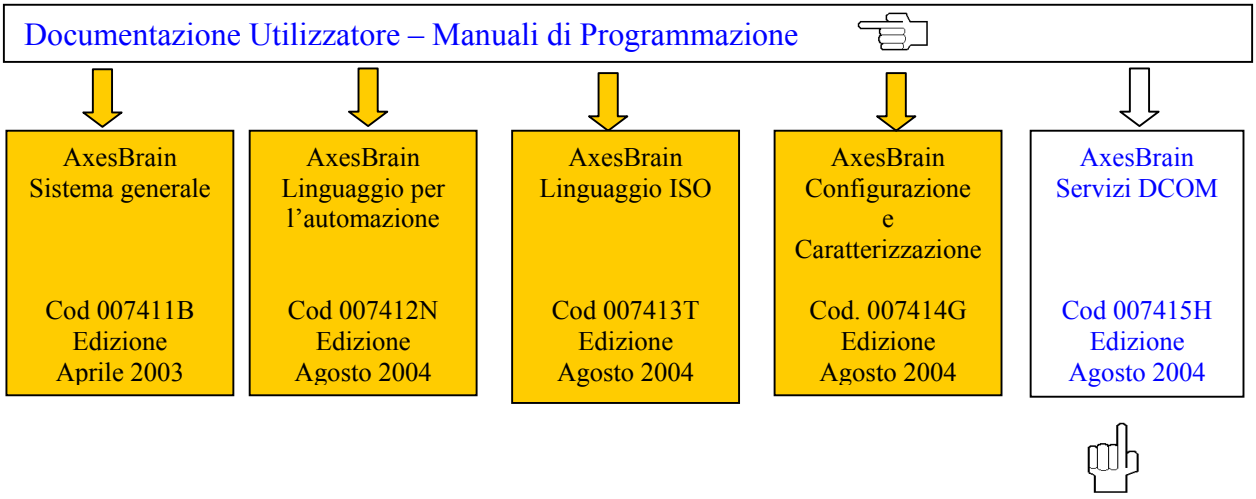
AB&T Tecnologie Informatiche™
Via dell'About, 2/A • 10015 Ivrea
Fax 0125 234397
www.bausano.net
info@bausano.net

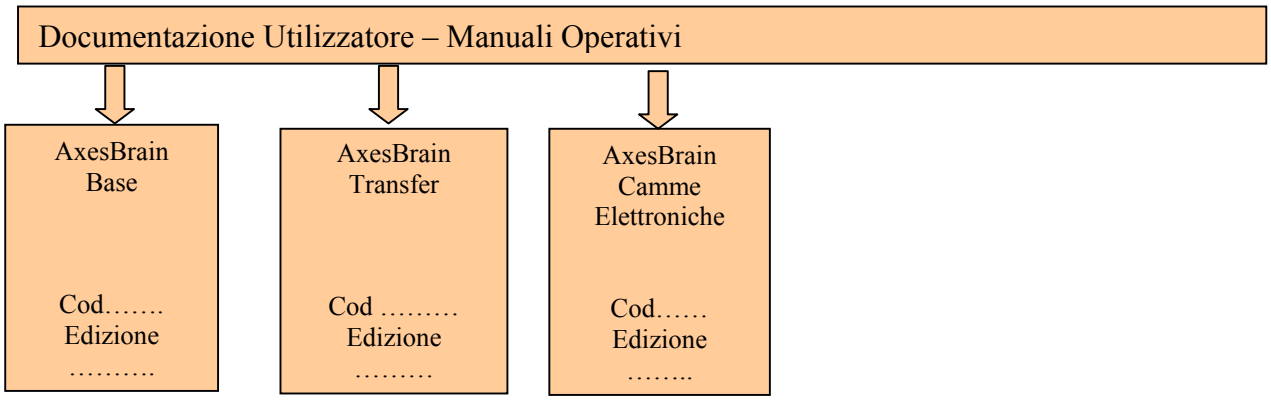
Informazioni legali

Le informazioni con questo documento, incluse URL e gli altri riferimenti sul sito Internet, possono cambiare senza alcun avvertimento.
A meno di specifica annotazione, i riferimenti a compagnie, organizzazioni, prodotti, persone ed eventi sono fittizi e non associate con reali compagnie, organizzazioni, prodotti, persone ed eventi.
L'AB&T Tecnologie Informatiche™ può registrare, licenziare, richiedere il copyright o marchi e rivendicare la proprietà intellettuale a tutti gli argomenti trattatati in questo documento.
Senza limitare i diritti sotto copyright, nessuna parte di questo documento può essere riprodotta, o modificata o trasmessa sotto ogni forma o mezzo (elettronico, meccanico,per fotocopiare, per registrare, od altro), senza l'espressa autorizzazione dell' AB&T Tecnologie Informatiche™.
Eccetto che per accordi scritti con la AB&T Tecnologie Informatiche™ la fornitura di questo documento non autorizza nessuno alla registrazione, a dare licenze, a richiedere il copyright o marchi e a rivendicare la proprietà intellettuale agli argomenti trattati in questo documento.

AxesBrain™, VisAlgo™, ScadaMERCURIO™ sono marchi registrati
©AxesBrain, ©VisAlgo sono tutelati da copyright
ActiveX, DirectX, JScript, Microsoft, Microsoft Press, MS-DOS, Visual Basic, Visual C++, Win32, Win32s, Windows, WDM, Windows NT, Windows 2000 e Windows Me sono o prodotti o marchi registrati dalla Microsoft Corporation negli Stati Uniti e/o negli altri Paesi.
I nomi di compagnie e prodotti citati in questo documento possono essere marchi registrati dai loro proprietari.

Panoramica della Documentazione Utilizzatore Servizi DCOM





Sommario

1. Architettura 1

2. Sistema base 4
- GetInformations 5
- GetSystemParameters 7
- GetSystemStatus 8
- System Command 9

3. Assi e mandrini 10
- ReadAxisRegister 11
- WriteAxisRegister 14
- StartAxisBuffer 17
- ReadAxisBuffer 19
- StopAxisBuffer 21
- SendCommandAxis 22
- ChangeOnFlyDinamicParameterAxis 24
- OpenHandleAxes 26
- CommandMotionAxes 27
- CloseHandleAxes 31
- ChangeDinamicParameterSystem 32
- ChangeDinamicParameterSpeedSystem 34

4. Input/Output 36
- ReadIO 37
- WriteIO 39
- WaitIO 41
- ReadFieldIO 44
- WriteFieldDescription 46

5. Sottosistemi – AxesBrainL, AxesBrainISO, AxesBrainAWL e OEM Capability 49
- GetSubSystemStatus 50
- SubSystemCommand 52
- GetSubSystemRegistry 54
- SetSubSystemRegistry 57
- GetSubSystemArray 60
- SetSubSystemArray 63
- ExecSubSystemProcess 66
- ExecSubSystemSequence 69
- KillSubSystemProcess 72
- GetSubSystemRunningProcesses 74
- GetSubSystemProcessRegistry 76
- SetSubSystemProcessRegistry 79
- GetSubSystemProcessParameters 82
- SetSubSystemProcessParameters 85
- ExecSubSystemInstruction 88
- KillSubSystemInstruction 91
- SendOrClearSubSystemEventSignal 93
- WriteServiceParameterAndContinue 96
- WriteServiceElementAndContinue 99
- SetBreakPoint 103

- GetSubSystemBreakPointsProcesses....106
- GetValueDebug....109
- GetSubSystemProcessRecord....110

6. Segnalazione d'impianto....114
- GetEventsLogFile....115
- DeleteEventsLOGFILE....117

7. Configurazione del sistema....118
- ReadSystemConfiguration....119
- EditSystemConfiguration....121
- GetNumberAndNameKindElement....123
- ReadElementConfiguration....125
- EditElementConfiguration....129

8. Eventi dal server....131
- Evento....132
- OnSystemEvent....135
- OnCompleteCommand....137
- OnCompleteElement....139
- OnRequestServiceEvent....141
- OnRequestServiceEventElement....144
- EventError....146

9. Collegamento a Visual Basic....148
- Funzionalità per l'utilizzo del Server con Visual Basic....148
- Esempio di uso del server in Visual Basic:....149

10. Elenco Errori....151

1.Architettura

Il prodotto "AxesBrain" è stato sviluppato per offrire dei servizi di movimentazione agli applicativi utente, utilizzando la tecnologia "DCOM", sarà così possibile avere l'accesso a questi servizi non solo dallo stesso PC, ma anche da PC integrati in rete locale o collegati tramite Internet.

Per utilizzare i servizi di movimentazione possono essere utilizzate essenzialmente due strade o utilizzando le funzioni dirette alle risorse assi, mandrini e i segnali di ingesso uscita oppure avvalendosi di cicli programmati. Tramite i linguaggi di programmazione vengono definite delle procedure o cicli di percorsi degli assi e di operazioni di manipolazione e lavorazione, nel nostro caso abbiamo tre linguaggi disponibili: AxesBrainL, AxesBrainISO e AxesBrainAWL che l'utente può utilizzare a secondo del tipo di applicazione che si viene a presentare, è inoltre possibile integrare come libreria , delle funzionalità o addirittura dei processi per la gestione di cicli di lavorazione personalizzati ad esigenze specifiche, completamente sviluppate dall' utente.

I servizi di movimentazione sono:

- FUNZIONI dirette
    - comandi agli assi e ai segnali d'ingrasso uscita
    - acquisizioni dei valori dal campo , lettura assi, I/O, trasduttori, ecc.
    - impostazioni delle modalità per le traiettorie, le acquisizioni , ecc.
- CICLI di lavorazione e manipolazione con programmazione tramite
    - AxesBrainL - Linguaggio proprietario adatto per descrivere i cicli di manipolazione per l'automazione in genere
    - AxesBrainISO - Linguaggio standard ISO adatto per descrivere i cicli di lavorazione tipici della fresatura, foratura e tornitura.
    - AxesBrainAWL - Linguaggio AWL per descrivere i cicli di PLC
- FUNZIONI speciali "OEM"
- CICLI di lavorazione e manipolazione con la programmazione di procedure speciali "OEM "

Il "AxesBrain" è un prodotto articolato in moduli funzionali, in modo da poter avere una ampia flessibilità sia a livello di gestione dei dispositivi che in termini interfaccia utente che di prestazioni.

Il "AxesBrain" è così suddiviso:

- "AxesBrain OpenEnviroment" modulo per il pilotaggio dei dispositivi d' interfaccia, oltre che gestire alcune gamme di dispositivi commerciali è stato sviluppato in modo da poter ampliare i dispositivi a seconda delle esigenze degli utenti .

- "AxesBrainServer" modulo base, fornitore dei servizi del sistema in tecnologia "DCOM" .

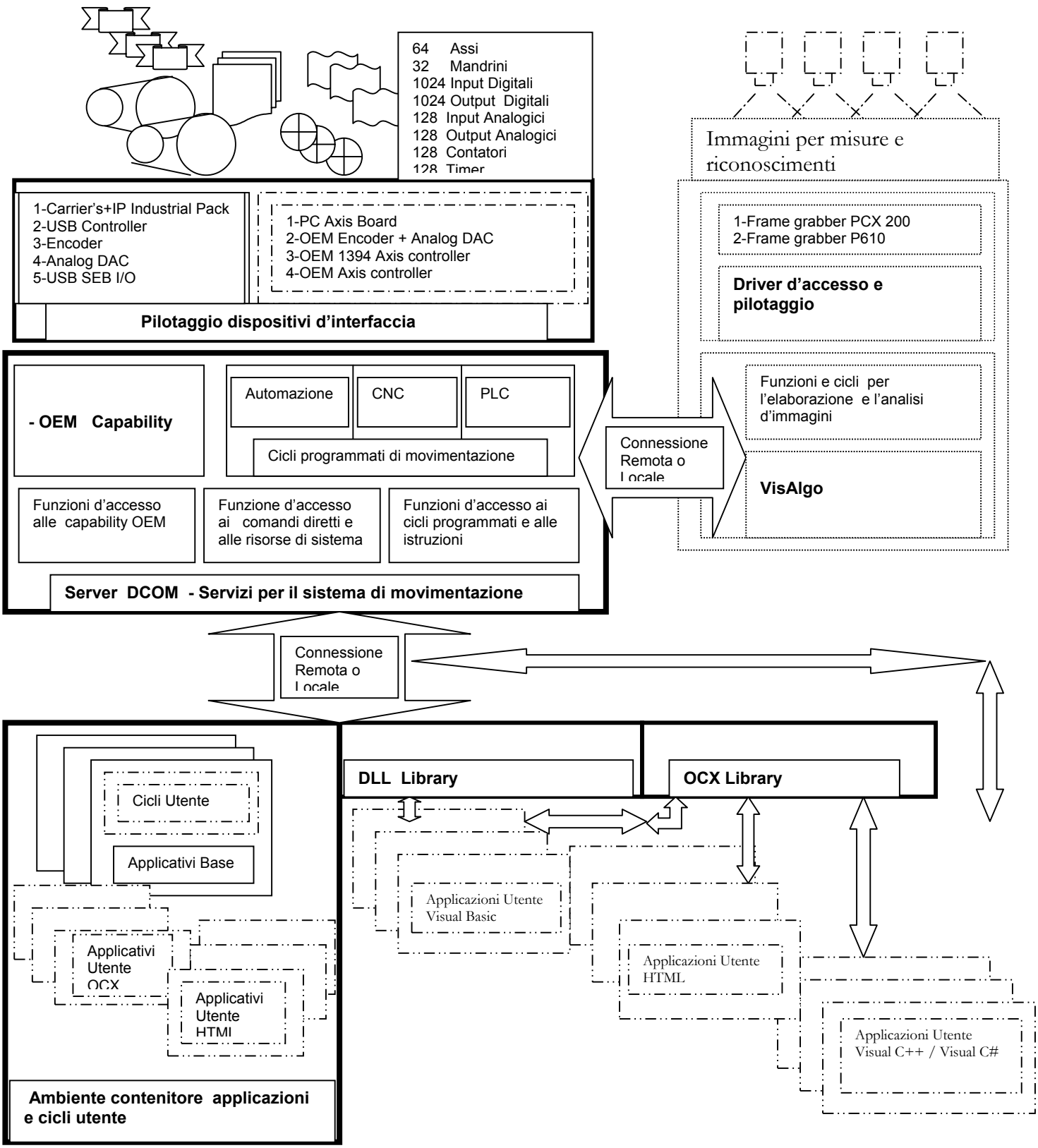
- "AxesBrainOpenTemplate" modulo di classi base per sviluppare nuove prestazioni direttamente dal l'utente, sia in termini di funzionalità che di procedure.

- "AxesBrainBase" modulo d'intefaccia per la connessione con i servizi per applicativi in Visual Basic o Visual C++

- "AxesBrainCom" modulo d'intefaccia per la connessione con i servizi per le pagine utente HTML o per applicativi scritti nei vari linguaggi di programmazione.

- "AxesBrainStudio" ambiente d'interfaccia utente, modulo che permette la realizzazione d'interfacce utente con alto grado di personalizzazione.

Schema dell'architettura AxesBrain



Capitolo 1

1

2.Sistema base

Per poter gestire le risorse di sistema nella loro globalità con i cicli di lavorazione è necessario avere delle funzioni per ottenere lo stato, dare o togliere potenza agli assi, cancellare i cicli di lavorazione n esecuzione sugli applicativi utente o programmati con i rispettivi linguaggi

1-GetInformations	Legge le informazioni del sistema
2-GetSystemStatus	Legge lo stato del sistema
3-SystemCommand	Invia un comando per accendere la potenza agli assi o toglierla o azzera le richieste di funzionamento sia per i cicli d'automazione che per i cicli di CNC che per i comandi assi in corso

○ GetInformations

Legge le informazioni del sistema

STDMETHODIMP GetInformations (

int *	**piStatus,**
int *	**piNumberSampleError,**
double *	**pdTimeSample,**
double *	**pdTimeUsed**
)	

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Driver di accesso alle risorse non trovato

Parametri

int *piStatus puntatore ad una variabile int dove viene caricato lo stato
I bit seguenti indicano lo stato del sistema:

1	ServoOn	0x1
2	System wait	0x2
3	System running	0x4
4	Anticollision	0x8

int *piNumberSampleError puntatore ad una variabile int dove viene caricato il numero di volte che la campionatura è andata in FrameError

double *piSampleTimer puntatore ad una variabile double dove viene caricato il tempo di campionatura in secondi.

double *piTimeUsed puntatore ad una variabile double dove viene caricato il tempo usato per la gestione del sistema AxesBrain.

Esempio

```
int             iStatus;
int             iClockCPU;
double          dTimeSample;
double          dTimeUsed;
HRESULT         hr;

 hr =  m_GMSSrv->GetGmsInformations( &iStatus , &iClockCPU , &dTimeSample, &dTimeUsed );
 if ( hr != S_OK )
{
        AfxMessageBox( "System not ready" );
}
else
{
        char sStringa[128];

        if ( iStatus & 1 )
        {
                AfxMessageBox( "Servo On" );
        }
        if ( iStatus & 2 )
        {
                AfxMessageBox( "System wait" );
        }
        if ( iStatus & 4 )
        {
                AfxMessageBox( "System running" );
        }

        sprintf(sStringa,"NumberSampleError=%ld  Time Sample=%lf Time Used=%lf",
                                        iClockCPU,
                                        dTimeSample,
                                        dTimeUsed );
        AfxMessageBox( sStringa );
}
```

o GetSystemParameters

Legge i parametri del sistema

STDMETHODIMP GetSystemParameters (
int ikindParameter,
ELEMENTS * pElements,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Driver di accesso alle risorse non trovato

Parametri

int	iKindParameters	Tipo di parametro richiesto 1 Tuttigli stati degli eventi o semafori I 128 eventi suddivisi i pacchetti di 32 sono caricati nelle prime quattro variabili di pElements -1 –128 Ritorna lo stato del numero di evento (negativo) nella prima variabile di pElements
ELEMENTS *	pElements	Struttura pElementi
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Numero degli eventi errato

○ GetSystemStatus

Legge lo stato del sistema

STDMETHODIMP GetSystemStatus (int *piStatus)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Driver di accesso alle risorse non trovato

Parametri

int *pStatus puntatore ad una variabile int dove viene caricato lo stato
I bit seguenti indicano lo stato del sistema:

4	ServoOn	0x8
5	Hold su AxesBrain_Automation	0x10
6	CycleStart su AxesBrain _Automation	0x20
7	Hold su AxesBrain _CNC	0x40
8	CycleStart su AxesBrain _CNC	0x80

Esempio

```
int             iStatus;
HRESULT         hr;

 hr =  m_GMSSrv->GetSystemStatus( &iStatus );
 if ( hr != S_OK )
{
        AfxMessageBox( "System not ready" );
}
else
if ( iStatus & 8 )
{
        AfxMessageBox( "Servo On" );
}
if ( iStatus & 0x10 )
{
        AfxMessageBox( "Automation cycle in Hold status" );
}
if ( iStatus & 0x20 )
{
        AfxMessageBox( "Automation cycle in Cycle Start status" );
}
if ( iStatus & 0x40)
{
        AfxMessageBox( "CNC cycle in Hold status" );
}
if ( iStatus & 0x80 )
{
        AfxMessageBox( "CNC cycle in Cycle Start status" );
}
```

○ System Command

Invia un comando per accendere la potenza agli assi o toglierla o azzera le richieste di funzionamento sia per i cicli d'automazione che per i cicli di CNC che per i comandi assi in corso

STDMETHODIMP SystemCommand (int iCommand)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iCommand	1	Emergency	Toglie la potenza agli assi
		2	Servo On	Attiva la potenza agli assi
		3	System Reset	Azzera le richieste attive
		4	Unlock	Sblocca i partprogram in LOCK

Esempio

```
HRESULT         hr;

 hr =  m_GMSSrv->SystemCommand( 2 ); // Servo On
 if ( hr != S_OK )
{
        AfxMessageBox( "System not ready" );
}
else
{
        AfxMessageBox( "Servo On Command OK" );
}
```

Capitolo

2

3.Assi e mandrini

Per poter gestire le risorse di assi e mandrini abbiamo funzioni che lavorano direttamente sulle risorse stesse riferite tramite il loro nome. Per poter lanciare dei movimenti coordinati con sistemi di riferimento tra loro omogenei e con traiettorie continue o "SPLINE" è necessario aprire degli HANDLE di gestione.

1-ReadAxisRegister	Legge il valore di un registro dell'asse o mandrino
2-WriteAxisRegister	Scrive un valore di un registro su un asse o mandrino
3-StartAxisBuffer	Inizia la memorizzazione di fino a 4 registri di un asse
4-ReadAxisBuffer	Legge i dati memorizzazione dell' asse
5-StopAxisBuffer	Ferma la memorizzazione di un asse
6-SendCommandAxis	Invia un comando ad un asse
7-ChangeOnFlyDinamicParameterAxis	Cambia al volo i parametri dinamici di un asse
8-OpenHandleAxes	Apertura di un HANDLE di gestione assi
9-CommandMotionAxes	Comando di movimento sugli assi
10-CloseHandleAxes	Chiusura di un HANDLE di gestione assi
11-ChangeDinamicParameterSystem	Cambia i parametri dinamici del sistema
12-ChangeDinamicParameterSpeedSystem	Cambia i parametri dinamici dei mandrini del sistema

○ ReadAxisRegister

Legge il valore di un registro dell'asse o mandrino

STDMETHODIMP ReadAxisRegister (

int iKindName,
BSTR sAxisName,
int iNumberAxis
int iKindRegister,
double * pdValue,
int * piError)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sAxisName	Nome dell'asse o mandrino
int	iNumberAxis	Numero asse o mandrino nel sistema

int iKindRead	Tipo di registro 0 Posizione teorica o Velocità teorica mandrino 1 Posizione reale o Velocità reale mandrino 2 Errore 3 Posizione reale assoluta 4 Posizione teorica assoluta 5 Offset relativo 6 Volt 7 KP 8 KI 9 KD 10 Velocity Feed Forward 11 Massimo errore integrativo 12 Informazioni tipo asse 13 KD2 14 Acceleration Feed Forward 15 Volt offset 16 Volt assoluti 17 Distanza dopo origine 18 Feed o Speed 19 Posizione di tastatura 20 Posizione di tastatura assoluta 21 Over travel positivo 22 Over travel negativo 23 Over travel positivo assoluto 24 Over travel negativo assoluto 25 Massima velocità 26Massima accelerazione 27Massima decelerazione 28In movimento 31Distanza reale 32Distanza teorica
double * pdValue	Ritorna il valore del registro
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato del asse o mandrino 3 Registro errato

Esempio

```
int             iStatus;
int             iKindRegister;
int             iError;
double          dValue;
HRESULT         hr;

iKindRegister = 0;    // Teorical Position
 hr = m_GMSSrv->ReadAxisRegister(       1,
                                        CComBSTR( "A1")  ,
                                        0,
                                        iKindRegister,
                                        &dValue,
                                        &iError
                                    );
                );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa,"Error ReadAxisRegister =%ld", iError);
        AfxMessageBox(sStringa);
}
else
{
        char sString[128];

        sprintf(sStringa,"Value=%lf", dValue);
        AfxMessageBox(sStringa);
}
```

○ WriteAxisRegister

Scrive il valore di un registro dell'asse o mandrino

```
STDMETHODIMP WriteAxesRegister (
                                    int     iKindName,
                                    BSTR    sAxisName,
                                    int     iNumberAxis
                                    int     iKindRegister,
                                    int     iMode,
                                    double  dValue,
                                    int *   piError
                                )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sAxisName	Nome dell'asse o mandrino
int	iNumberAxis	Numero asse o mandrino nel sistema

int	iKindRegister	Tipo di registro 0 Posizione teorica o Velocità teorica mandrino 1 Posizione reale o Velocità reale mandrino 2 Errore 3 Posizione reale assoluta 4 Posizione teorica assoluta 5 Offset relativo 6 Volt 7 KP 8 KI 9 KD 10 Velocity Feed Forward 11 Massimo errore integrativo 12 Informazioni tipo asse 13 KD2 14 Acceleration Feed Forward 15 Volt offset 16 Volt assoluti 17 Distanza dopo origine 18 Feed o Speed 19 Posizione di tastatura 20 Posizione di tastatura assoluta 21 Over travel positivo 22 Over travel negativo 23 Over travel positivo assoluto 24 Over travel negativo assoluto 25 Massima velocità 26 Massima accelerazione 27 Massima decelerazione
int	iMode	Modalità di registrazione 0 non persistente 1 persistente (rimane anche dopo il comando di reset)
double	dValue	Valore da scrivere sul registro
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato del asse o mandrino 3 Registro errato

Esempio

```
int             iStatus;
int             iKindRegister;
int             iError;
double          dValue;
HRESULT         hr;

dValue     = 100.
iKindRegister = 9;    // KD
 hr = m_GMSSrv->WriteAxisRegister(      1
                                        CComBSTR( "A1")  ,
                                        0,
                                        iKindRegister,
                                        1,        // persistente anche dopo il comando di reset
                                        dValue,
                                        &iError
                                    );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error WriteAxisRegister =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "WriteAxesRegister OK" );
}
```

○ StartAxisBuffer

Inizia la memorizzazione di fino a 4 registri di un asse su un'area interna a scorrimento circolare.

STDMETHODIMP StartAxisBuffer (

int	**iKindName,**
BSTR	**sAxisName,**
int	**iNumberAxis**
int	**iNumberRegister,**
int	**iKindRegister1,**
int	**iKindRegister2,**
int	**iKindRegister3,**
int	**iKindRegister4,**
int *	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sAxisName	Nome dell'asse o mandrino
int	iNumberAxis	Numero asse o mandrino nel sistema
int	iNumberRegister	Numero di registri da registrare
int	iKindRegister1	Tipo di registro 1
int	iKindRegister2	Tipo di registro 2
int	iKindRegister3	Tipo di registro 3
int	iKindRegister4	Tipo di registro 4 0 Posizione teorica o Velocità teorica mandrino 1 Posizione reale o Velocità reale mandrino 2 Errore 3 Posizione reale assoluta 4 Posizione teorica assoluta 5 Offset relativo 6 Volt
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato del asse o mandrino 3 Registro errato 4 Bufferizzazione già aperta 5 iNumberRegister errato

Esempio

```
HRESULT          hr;
int              iKindRegister;
int              iError;
HRESULT          hr;

iKindRegister = 2;    // Error position

 hr =  m_GMSSrv->StartAxisBuffer(  1
                                    CComBSTR( "A1") ,
                                   0,
                                   1,
                                   iKindRegister,
                                   0,
                                   0,
                                   0,
                                   &iError
                                  );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa,"Error StartAxisBuffer = %ld", iError);
        AfxMessageBox(sStringa);
}
else
{
        AfxMessageBox( "StartAxesBuffer OK" );
}
```

○ ReadAxisBuffer

Legge i dati memorizzati dell'asse

```
STDMETHODIMP ReadAxisBuffer (   VARIANT * pVar,
                                int *     piNumberElement,
                                int *     piError
                            )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

VARIANT * pVar	VARIANT di double degli elementi memorizzati
int * piNumberElement	Ritorna il numero di elementi del VARIANT * pVar
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Bufferrizzazione non aperta 3 Error nella creazione del VARIANT

Esempio

```
VARIANT                 Var;
SAFEARRAY FAR*          psa;
HRESULT                 hr;
int                     iError;
int                     iNumberElements;
long                    lNumElem;
long                    lLBound;
long                    lUBound;
double   *              pValDouble;

VariantInit(&Var);
Var.vt        = VT_ARRAY | VT_R8 ;
Var.parray  = NULL;
hr =  m_GMSSrv->ReadAxisBuffer( &Var , &iNumberElements ,&iError  );
if ( hr == S_OK )
{
        if ( Var.vt == ( VT_ARRAY | VT_R8 ) )
        {
                psa = Var.parray;
                if ( psa )
                {
                        long ilog;

                        hr = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElem   = lUBound - lLBound + 1;
                        pValDouble = new double[lNumElem];
                        if ( !FAILED ( hr ))
                        {
                                for ( ilog=0 ; ilog< lNumElem; ilog++ )
                                {
                                        SafeArrayGetElement( psa, &ilog , & pValDouble [ilog]);
                                }
                        }
                        SafeArrayDestroyData( psa );
                        SafeArrayDestroyDescriptor( psa );

                        …….
                        …….
                        delete pValDouble;
                }
        }
}
else
{
        AfxMessageBox( "StopAxisBuffer non OK" );
}
```

○ StopAxisBuffer

Ferma la memorizzazione di un asse

STDMETHODIMP StopAxisBuffer (int * piError)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Bufferrizzazione non aperta

Esempio

```
HRESULT         hr;
int             iError;

 hr =  m_GMSSrv->StopAxisBuffer( &iError  );
 if ( hr != S_OK )
{
        AfxMessageBox( "StopAxisBuffer non OK" );
}
else
{
        AfxMessageBox( "StopAxisBuffer  OK" );
}
```

○ SendCommandAxis

Invia un comando ad un asse

```
STDMETHODIMP SendCommandAxes (    int     iKindName,
                                  BSTR    sAxisName,
                                  int     iNumberAxis
                                  int     iKindCommand,
                                  double  dPosition,
                                  int     iMode,
                                  double  dVelocity,
                                  double  dAcceleration,
                                  double  dDecelaration,
                                  int *   piError
                              )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sAxisName	Nome dell'asse o mandrino
int	iNumberAxis	Numero asse o mandrino nel sistema
int	iKindCommand	Tipo di comando 1 Start movimento asse 2 Stop movimento asse 3 Posizionamento asse 4 Origine
double	dPosition	Valore di posizione
int	iMode	Modalità Assoluta o Incrementale nella posizione 0 = Posizione Incrementale 1 = Posizione Assoluta
double	dVelocity	Valore di velocità percentuale nello Start movimento e esplicita nel posizionamento
double	dAcceleration	Valore di accelerazione
double	dDeceleration	Valore di decelerazione
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato del asse o mandrino 3 Comando errato 4 Parametro errato

Esempio

```
int             iMode;
int             iKindCommand;
int             iError;
HRESULT         hr;

iKindCommand = 1;    // Start motion
iMode = 2 // Absolute Value
 hr = m_GMSSrv->SendCommandAxis(     1,
                                     CComBSTR( "A1")  ,
                                     0,
                                     iKindCommand,
                                     99999,
                                     iMode,
                                     2000,
                                     1000,
                                     1000,
                                     &iError
                                );
                );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error SendCommandAxis =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "SendCommandAxis OK" );
}
```

○ ChangeOnFlyDinamicParameterAxis

Cambia al volo i parametri dinamici di un asse

STDMETHODIMP ChangeOnFlyDinamicParameterAxis (

int	**iKindName,**
BSTR	**sAxisName,**
int	**iNumberAxis**
int	**iMode,**
double	**dVelocity,**
double	**dAcceleration,**
double	**dDecelaration,**
int *	**piError**

)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sAxisName	Nome dell'asse o mandrino
int	iNumberAxis	Numero asse o mandrino nel sistema
int	iMode	Modalità 1 = Valori percentuali 2 = Valori assoluti
double	dVelocity	Valore di velocità
double	dAcceleration	Valore di accelerazione
double	dDeceleration	Valore di decelerazione
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato del asse o mandrino 3 Parametri errati

Esempio

```
int             iMode;
int             iError;
HRESULT         hr;

iMode = 2 // Absolute Value
hr = m_GMSSrv->ChangeOnFlyDinamicParameterAxis(     1,
                                                    CComBSTR( "A1")  ,
                                                    0,
                                                    iMode,
                                                    2000,
                                                    1000,
                                                    1000,
                                                    &iError
                                                );

 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error ChangeOnFlyDinamicParameterAxis =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "ChangeOnFlyDinamicParameterAxis OK" );
}
```

○ OpenHandleAxes

Apertura di un HANDLE di gestione assi

STDMETHODIMP OpenHandleAxes (
int * piHandle,
int * piError)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int * piHandle	Ritorna un HANDLE per la gestione assi coordinati
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato

Esempio

```
HRESULT         hr;
int             iError;
int             iHandle;

 hr =  m_GMSSrv->OpenHandleAxes( &iHandle,
                                 &iError    );
 if ( hr != S_OK )
{
        AfxMessageBox( "OpenHandleAxes non OK" );
}
else
{
        AfxMessageBox( "OpenHandleAxes OK" );
}
```

○ CommandMotionAxes

Comando di movimento sugli assi

STDMETHODIMP CommandMotionAxes (

int	**iHandle,**
BSTR	**sCommand,**
BSTR	**sParameters,**
int	**iNotifyRequest,**
int *	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iHandle HANDLE per la gestione assi coordinati da chiudere

BSTR sCommand	Comando movimento sugli assi:
HOM	Origine di un asse (HOMing)
MOV	Movimento interpolato linearmente di un gruppo di assi (MOVe)
MOC	Movimento interpolato linearmente di un gruppo di assi con condizione di start (MOve Conditional)
CIR	Movimento interpolato circolare o ellittico in senso orario di un gruppo di assi (CIrcular Right)
CIL	Movimento interpolato circolare o ellittico in senso antiorario di un gruppo di assi (CIrcular Left)
CRR	Movimento interpolato circolare o ellittico in senso orario di un gruppo di assi con raggio noto (Circular Radius Right)
CRL	Movimento interpolato circolare o ellittico in senso antiorario di un gruppo di assi con raggio noto (Circular Radius Left)
STC	Inizio di una movimentazione in continuo con definizione di percorso (STart Continuous)
HLC	Attesa del completamento della movimentazione in continuo (HaLt Continuous)
ABC	Cancellazione del movimento in continuo (Abort Continuous)
CAP	Cambia i parametri asse (Change Axis Parameter)
HMS	Gestione Master Slave (Handling Master Slave)
HEC	Gestione delle camme (Handling Electronic Cam)
CFR	Cambia i parametri dinamici di un asse (Change Feed Rate)
CPL	Cambia il loop di posizione (Change Position Loop)
PRD	Legge le posizioni dell'asse (Position ReaD)
RAV	Legge i parametri dell'asse (Read Axis Value)
RSV	Legge la velocità di un mandrino (Read Speed Value)
SFP	Imposta la velocità del profilo di movimentazione (Set Feed Profile)
SPD	Imposta la velocità di rotazione di un mandrino (SPeeD)
TCH	Movimento con tostatura (TouCH)
TMT	Movimento con ricerca valore di segnale analogico (Test Movement Trasducer)
TMS	Movimento con ricerca valore di sensore digitale (Test Movement Sensor)
TPE	Abilita il tastatore (Touch Probe Enable)
SZP	Definisci la posizione di un set zeri macchina (Set Zero Point)
LZP	Attiva un set di zeri macchina (Load Zero Point)
PIN	Flag di incrementale su un asse (Position INcremental)
PAB	Flag di assoluto su un asse (Position ABsolute)
ANT	Gestione dell' anticollisione(ANTicollision)

BSTR sParameters	Parametri del comando (Vedi Manuale Riferimento AxesBrainL per i parametri dei comandi)
int iNotifyRequest	Richiesta di notifica al completamento del comando 0 = Nessuna notifica # = Chiave di notifica
(OnCompleteCommand	Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta)) BSTR "CommandMotionAxis", int iHandle, int iNotifyRequest, int iNumberElement, VARIANT * pVariant array[double] dei risultati a seconda del comando dato.
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 HANDLE non corretto 3 Comando non corretto 4 Assi in movimento 5 Parametri non corretti

Esempio

```
HRESULT          hr;
int              iError;
int              iHandle;

hr =  m_GMSSrv->OpenHandleAxes( &iHandle,
                                &iError    );
 if ( hr != S_OK )
{
        AfxMessageBox( "OpenHandleAxes non OK" );
}
else
{
        hr =  m_GMSSrv->CommandMotionAxes(   iHandle,
                                             CComBSTR( "MOV") ,
                                             CComBSTR( "A1,100,B1,200"),
                                             0,
                                             &iError    );
        if ( hr != S_OK )
        {
                AfxMessageBox( "CommandMotionAxes non OK" );
        }
        else
        {
                AfxMessageBox( "CommandMotionAxes OK" );
        }
        ……..
        ……..
        hr =  m_GMSSrv->CloseHandleAxes( iHandle,
                                         &iError    );
        if ( hr != S_OK )
        {
                AfxMessageBox( "CloseHandleAxes non OK" );
        }
        else
        {
                AfxMessageBox( "CloseHandleAxes OK" );
        }
}
```

○ CloseHandleAxes

Chiusura di un HANDLE di gestione assi

STDMETHODIMP CloseHandleAxes (
int iHandle,
int * piError)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iHandle	HANDLE per la gestione assi coordinati da chiudere
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 HANDLE non corretto

Esempio

```
HRESULT         hr;
int             iError;
int             iHandle;

hr =  m_GMSSrv->OpenHandleAxes( &iHandle,
                                &iError    );
 if ( hr != S_OK )
{
        AfxMessageBox( "OpenHandleAxes non OK" );
}
else
{
        ……..
        ……..
        hr =  m_GMSSrv->CloseHandleAxes( iHandle,
                                         &iError    );
        if ( hr != S_OK )
        {
                AfxMessageBox( " CloseHandleAxes non OK" );
        }
        else
        {
                AfxMessageBox( " CloseHandleAxes OK" );
        }
}
```

○ ChangeDinamicParameterSystem

Cambia i parametri dinamici del sistema

```
STDMETHODIMP ChangeDinamicParameterSystem (
                                        double   dVelocity,
                                        double   dAcceleration,
                                        double   dDecelaration,
                                        double   dVelocityPerc,
                                        double   dAccelerationPerc,
                                        double   dDecelarationPerc,
                                        int *    piError
                                                        )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

double	dVelocity	Valore di velocità
double	dAcceleration	Valore di accelerazione
double	dDeceleration	Valore di decelerazione
double	dVelocityPerc	Valore di velocità percentuale
double	dAccelerationPerc	Valore di accelerazione percentuale
double	dDecelerationPerc	Valore di decelerazione percentuale

int * piError — Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:

1. Driver di accesso alle risorse non trovato
2. Parametri errati

Esempio

```
int             iError;
HRESULT         hr;

hr = m_GMSSrv->ChangeDinamicParameteSystem(
                                                        2000,
                                                        1000,
                                                        1000,
                                                        80,
                                                        80,
                                                        80,
                                                        &iError
                                                );

 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error ChangeDinamicParameteSystem =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "ChangeDinamicParameteSystem OK" );
}
```

o ChangeDinamicParameterSpeedSystem

Cambia i parametri dinamici dei mandrini del sistema

STDMETHODIMP ChangeDinamicParameterSpeedSystem (
double dVelocityPerc,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

double	dVelocity	Valore di velocità
int * piError		Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Parametri errati

Esempio

```
int             iError;
HRESULT         hr;

hr = m_GMSSrv->ChangeDinamicParameterSpeedSystem(
                                                        50,
                                                        &iError
                                                );

 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, ”Error ChangeDinamicParameterSpeedSystem =%ld”, iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( “ChangeDinamicParameterSpeedSystem OK” );
}
```

Capitolo

3

4. Input/Output

Per poter gestire le risorse di Input e di Output digitali o analogiche abbiamo delle funzione che fanno riferimento agli I/O tramite il loro nome.

1-ReadIO	Legge il valore del I/O/Globali
2-WriteIO	Scrive il valore di un I/O/Globali
3-WaitIO	Attendi il valore di un I/O/Globali
4-ReadFielIO	Legge il campo stringa di un I/O/Globali
5-WriteFieldDescription	Riscrive le descrizioni degli I/O/Globali

○ ReadIO

Legge il valore del I/O

STDMETHODIMP ReadIO (

int	**iKindName**
BSTR	**sIOName,**
int	**iNumberIO,**
int	**iKindIO,**
int	**iKindValue,**
double *	**pdValue,**
int *	**piError,**

)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iKindName — Tipo di identificazione asse per numero o nome
- 1 Nome
- 2 Numero

BSTR sIOName — Nome dell'Input Output

int iNumberIO — Numero dell'I/O

int iKindIO — Tipo di IO
1. Input digitale
2. Output digitale
3. Input analogico
4. Output analogico
5. Globale

int iKindValue — Tipo di valore
- 1 Bit
- 2 Byte
- 3 WORD (16 bit)

double * pdValue — Ritorna il valore del Input Output

int * piError — Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:
1. Driver di accesso alle risorse non trovato
2. Nome errato dell'I/O

Esempio

```
int             iKindValue;
int             iError;
double          dValue;
HRESULT         hr;

iKindValue = 1;    // Bit
hr = m_GMSSrv->ReadIO(          2.
                                CComBSTR( "") ,
                                10,                         // 10 IO
                                iKindValue,
                                &dValue,
                                &iError
                            );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error ReadIO =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "ReadIO OK" );
}
```

○ WriteIO

Scrive i valore del I/O

STDMETHODIMP WriteIO (

int	**iKindName**
BSTR	**sIOName,**
int	**iNumberIO,**
int	**iKindIO,**
int	**iKindValue,**
double	**dValue,**
int *	**piError,**

)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR sIOName	Nome dell'Input Output
int iNumberIO	Numero dell'I/O
int iKindIO	Tipo di IO 1 Input digitale 2 Output digitale 3 Input analogico 4 Output analogico 5 Globale
nt iKindValue	Tipo di valore 1 Bit 2 Byte 3 WORD (16 bit)
double dValue	Valore del Input Output
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato dell'I/O

Esempio

```
int             iKindValue;
int             iError;
HRESULT         hr;

iKindValue = 1;    // Bit
hr = m_GMSSrv->WriteIO(         2.
                                CComBSTR( "") ,
                                10,                         // 10 IO
                                iKindValue,
                                1,
                                &iError
                        );
                );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error WriteIO =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "WriteIO OK" );
}
```

○ WaitIO

Attendi il valore di un I/O entri in un ambito di valori

STDMETHODIMP WaitIO (

int	**iKindName**
BSTR	**sIOName,**
int	**iNumberIO,**
int	**iKindIO,**
int	**iKindValue,**
double	**dValueLow,**
double	**dValueHight,**
double	**dTimeSec,**
int	**iNotifyRequest,**
int *	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sIOName	Nome dell'Input Output
int	iNumberIO	Numero dell'I/O
int	iKindIO	Tipo di IO 1 Input digitale 2 Output digitale 3 Input analogico 4 Output analogico
int	iKindValue	Tipo di valore 1 Bit 2 Byte 3 WORD (16 bit)
double	dValueLow	il valore del Input Output maggiore o uguale
double	dValueHight	il valore del Input Output minore o uguale
double	dTimeSec	Tempo in secondi di timeout, se 0 il tempo di timeout e' infinito.

int iNotifyRequest Richiesta di notifica al completamento del comando
= Chiave di notifica
(OnCompleteCommand Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta))
Valore nel range :
BSTR "WaitIO", 0 , int iNotifyRequest, 1 , VARIANT * pVariant [double] valore I/O.

Se il tempo di timeout è scaduto :
BSTR "WaitIO", 1 , int iNotifyRequest, 1 , VARIANT * pVariant [double] valore I/O.

int * piError Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:
1 Driver di accesso alle risorse non trovato
2 Nome errato dell'I/O

Esempio

```
int             iKindValue;
int             iError;
HRESULT         hr;

iKindValue = 1;    // Bit
hr = m_GMSSrv->WaitIO(          2.
                                CComBSTR( "") ,
                                10,                         // 10 IO
                                iKindValue,
                                1,
                                1,
                                &iError
                        );
                );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error WaitIO =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( " WaitIO OK" );
}
```

○ ReadFieldIO

Legge il valore di un campo stringa del I/O

```
STDMETHODIMP ReadFieldIO (
                              int       iKindName
                              BSTR      sIOName,
                              int       iNumberIO,
                              int       iKindIO,
                              int       iKindValue,
                              BSTR *    sValue,
                              int *     piError,
                            )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR	sIOName	Nome dell'Input Output
int	iNumberIO	Numero dell'I/O
int	iKindIO	Tipo di IO 1 Input digitale 2 Output digitale 3 Input analogico 4 Output analogico 5 Globale 6 Asse fisico 7 Asse virtuale ascisse 8 Asse virtuale ordinate
int	iKindValue	Tipo di stringa 1 Nome 2 Descrizione 3 Esplicit Description
BSTR *	sValue	Ritorna il valore del campo stringa del Input Output
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Driver di accesso alle risorse non trovato 2 Nome errato dell'I/O

Esempio

```
                    USES_CONVERSION;
int                 iKindValue;
int                 iError;
HRESULT             hr;
BSTR                sTmp;
CString             szText

iKindValue = 2;    // Descrizione
hr = m_GMSSrv-> ReadFieldIO (   2.
                                CComBSTR( "") ,
                                10,                         // numero 10 IO
                                iKindValue,
                                & sTmp,
                                &iError
                            );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, "Error ReadIO =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        szText  = OLE2A(sTmp);
        AfxMessageBox( "ReadIO OK" );
}
```

- **WriteFieldDescription**

Riscrive le descrizioni degli I/O/Globali

STDMETHODIMP WriteFieldDescription (

int	**iKindName**
BSTR	**sIOName,**
int	**iNumberIO,**
int	**iKindIO,**
int	**iKindValue,**
BSTR	**sValue,**
int *	**piError,**

)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iKindName	Tipo di identificazione asse per numero o nome 1 Nome 2 Numero
BSTR sIOName	Nome dell'Input Output
int iNumberIO	Numero dell'I/O
int iKindIO	Tipo di IO 1 Input digitale 2 Output digitale 3 Input analogico 4 Output analogico 5 Globale 6 Asse fisico 7 Asse virtuale ascisse 8 Asse virtuale ordinate
int iKindValue	Tipo di stringa 3 Esplicit Description
BSTR sValue	Campo descrizione
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 9 Driver di accesso alle risorse non trovato 10 Nome errato dell'I/O

Esempio

```
USES_CONVERSION;
int iError, iNumIO;

CString sTmp;

sTmp = m_GridInp.GetItemText(dispinfo->item.row, dispinfo->item.col);

iNumIO = atoi(m_GridInp.GetItemText(dispinfo->item.row, 0));

m_pDG->m_pServer->WriteFieldDescription(2, CComBSTR(""), iNumIO, 1, 3, CComBSTR(sTmp), &iError);
```

Capitolo

4

5.Sottosistemi – AxesBrainL, AxesBrainISO, AxesBrainAWL e OEM Capability

Funzioni per poter gestire le risorse di sistema nella loro globalità con i cicli di lavorazione e manipolazione programmati con i linguaggi AxesBrainL, AxesBrainISO, AxesBrainAWL e OEM Cabability , che permette la contemporaneità di più attività o cicli, adatto per la sua flessibilità all'automazione in senso lato.

1-GetSubSystemStatus	Legge lo stato di un sotto sistema
2-SubSystemCommand	Invia un comando al sotto sistema
3-GetSubSystemRegistry	Legge i registri delle sotto sistema
4-SetSubSystemRegistry	Scrive i registri di un sotto sistema
5-GetSubSystemArray	Legge le tabelle del sotto sistema
6-SetSubSystemArray	Scrive le tabelle del sotto sistema
7-ExecSubSystemProcess	Richiede l'esecuzione di un processo
8-ExecSubSystemSequence	Richiede l'esecuzione di un processo
9-KillSubSystemProcess	Sospende e cancella l'esecuzione di un processo
10-GetSubSystemRunningProcesses	Legge i processi in esecuzione nel sotto sistema
11-GetSubSystemProcessStatusRegistry	Legge i registri di stato di un processo
12-SetSubSystemProcessStatusRegistry	Scrive i registri di stato di un processo
13-GetSubSystemProcessParameters	Legge i parametri di un processo
14-SetSubSystemProcessParameters	Scrive i parametri di un processo
15-ExecSubSystemInstruction	Richiede l'esecuzione di una istruzione
16-KillSubSystemInstruction	Sospende e cancella l'esecuzione di una istruzione
17-SendOrClearSubSystemEventSignal	Usata come sincronizzazione semplice con il sottosistema segnalando un evento, se il processo del sottosistema era in attesa dell'evento viene riattivato, altrimenti l'evento viene memorizzato.
18-WriteServiceParameterAndContinue	Il processo del sotto sistema si pone in attesa inviando una richiesta si servizio, sarà appunto il l'esecutore del servizio a chiamare il metodo: "WriteServiceParameterAndContinue"
19-SetBreakPoint	Seleziona o rimuove delle linee di breakpoint
20-GetSubSystemBreakPointsProcesses	Richiede i break points attivati
21-GetSubSystemProcessStatusRecord	Legge i parametri di un processo
22-ExecIsoProgram	Esegue i processo ISO esterno
23-SendIsoInstruction	Acquisisce un' istruzione ISO dal'esterno
24-CloseIsoInstruction	Chiude un processo ISO esterno
25-KillIsoProgram	Cancella un processo ISO esterno
26-GetIsoRegister	Legge i parametri di un processo ISO esterno

○ GetSubSystemStatus

Legge lo stato di un sottosistema

STDMETHODIMP GetSubSystemStatus(
int iNumberSubSystem,
int * piStatus,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Driver di accesso alle risorse non trovato

Parametri

int iNumberSubSystem — Tipo di identificazione del sottosistema
1 AxesBrainL
2 AxesBrainISO
3 AxesBrainAWL
101 Primo sottosistema OEMCapability
102 Secondo sottosistema OEMCapability
…
109 Nono sottosistema OEMCapability

int *piStatus — puntatore ad una variabile int dove viene caricato lo stato
I bit seguenti indicano lo stato del sistema:

2 SubSystem wait
3 SubSystem running

int * piError — Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:
1 SubSystem non attivo
2 iNumberSubSystem errato

Esempio

```
int             iStatus;
int             iError;
HRESULT         hr;

 hr =  m_GMSSrv->GetSubSystemStatus(   1,
                                       &iStatus ,
                                       &iError
                                       );
if ( hr == S_OK )
{
        char sString[128];

        sprintf(sString, ”Error GetSubSystemStatus =%ld”, iError);
        AfxMessageBox( sString  );
}
else
{
        char sStringa[128];

        if ( iStatus & 2 )
        {
                AfxMessageBox( “SubSystem wait” );
        }
        if ( iStatus & 4 )
        {
                AfxMessageBox( “SubSystem running” );
        }
}
```

○ SubSystemCommand

Invia un comando al sotto sistema

STDMETHODIMP SubSystemCommand(
int iNumberSubSystem,
BSTR sCommandStr,
BSTR sParameterStr,
int iNumberParameterDouble,
VARIANT* pdVarParameter,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem Tipo di identificazione del sottosistema
1 AxesBrainL
2 AxesBrainISO
3 AxesBrainAWL
101 Primo sottosistema OEMCapability
102 Secondo sottosistema OEMCapability
...
...
109 Nono sottosistema OEMCapability

BSTR sCommandStr Stringa comando
Hold
CycleStart
SetStepByStep
NextStep
RemoveStepByStep

BSTR sParameterStr Stringa parametri comando

Int iNumberParameterDouble Numero double parametri aggiuntivi

VARIANT* pdVarParameter Parametri double

int * piError Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:
1 SubSystem non attivo
2 iNumberSubSystem errato
3 sCommandStr errato
4 sParameterStr errato
5 iNumberParameterDouble errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 3;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 3 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr   =  m_GMSSrv->SubSystemCommand(
                                1,
                                CComBSTR("CycleStart"),
                                CCOmBSTR("" ),
                                3,
                                &Var,
                                &iError
                                  );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf(sString, "Error GetSubSystemStatus =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "SubSystemCommand OK" );
        }
}
```

○ GetSubSystemRegistry

Legge i registri delle sotto sistema

STDMETHODIMP GetSubSystemRegistry(

int		**iNumberSubSystem,**
BSTR		**sNameRegistry,**
int		**iFirstElementRegistry,**
int		**iLastElementRegistry,**
int	*	**piNumberValue,**
VARIANT*		**pVar,**
int	*	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem — Tipo di identificazione del sottosistema
1 AxesBrainL
2 AxesBrainISO
3 AxesBrainAWL
101 Primo sottosistema OEMCapability
102 Secondo sottosistema OEMCapability
...
...
109 Nono sottosistema OEMCapability

BSTR sNameRegistry — Nome del tabella registro,
ogni sottosistema ne ha diverse di tabelle.

AxesBrainL	Global
	Dis
	Drt
	Drt Color
AxesBrainISO	Variable
AxesBrainAWL	Accumulator
	Counter
	Timer
	Digital Input
	Digital Output
	Analog Input
	Analog Output
	Merker
	Variable

int	iFirstElementRegistry	Primo elemento della tabella registro
int	iLastElementRegistry	Ultimo elemento della tabella registro
int *	piNumberValueDouble	Ritorna il numero di elementi
VARIANT*	pVar	Variant con i valori degli elementi
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sNameRegistry errato 3 iFirstElementRegistry errato 4 iLastElementRegistry errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
char                    sString[128];
int                     iError;
int                     iNumberGlobalsLoaded;
long                    lNumElement;
double                  pdVettValue;
long                    lLBound;
long                    lUBound;

VariantInit(&Var);
hr  =  m_GMSSrv->GetSubSystemRegistry(
                                        1,
                                        CComBSTR("Global"),
                                        23,
                                        43,
                                        &iNumberGlobalsLoaded,
                                        &Var,
                                        &iError
                                    );
if ( hr != S_OK )
{
        sprintf( sString, "Error GetSubSystemRegistry =%ld", iError );
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "GetSubSystemRegistry OK" );
        if ( iNumberGlobalsLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        .......
                        .......
                        delete pdVettValue;
                }
        }
}
```

○ SetSubSystemRegistry

Scrive i registri di un sotto sistema

STDMETHODIMP SetSubSystemRegistry(

int	**iNumberSubSystem,**
BSTR	**sNameRegistry,**
int	**iFirstElementRegistry,**
int	**iLastElementRegistry,**
int	**iNumberValue,**
VARIANT*	**pVar,**
int *	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sNameRegistry	Nome del tabella registro,ogni sottosistema ne ha diverse di tabelle. AxesBrainL Global AxesBrainISO Variable AxesBrainAWL Accumulator Counter Timer Digital Input Digital Output Analog Input Analog Output Merker Variable
int iFirstElementRegistry	Primo elemento della tabella registro
int iLastElementRegistry	Ultimo elemento della tabella registro
int iNumberValue	Numero elementi da caricare
VARIANT* pVar	Variant con i valori degli elementi

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sNameRegistry errato 3 iFirstElementRegistry errato 4 iLastElementRegistry errato 5 iNumberValue errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 10;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 10 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr   = m_GMSSrv->SetSubSystemRegistry(
                                        1,
                                        CComBSTR("Global"),
                                        23,
                                        42,
                                        10,
                                        &Var,
                                        &iError
                                                );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf(sString, "Error SetSubSystemRegistry =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "SetSubSystemRegistry OK" );
        }
}
```

○ GetSubSystemArray

Legge le tabelle del sotto sistema

STDMETHODIMP GetSubSystemArray(

int		**iNumberSubSystem,**
BSTR		**sNameArray,**
int		**iFirstElementArrayI,**
int		**iLastElementArrayI,**
int		**iFirstElementArrayJ,**
int		**iLastElementArrayJ,**
int	*	**piNumberValue,**
VARIANT*		**pVar,**
int	*	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... 109 Nono sottosistema OEMCapability
BSTR sNameArray	Nome del tabella registro,ogni sottosistema ne ha diverse di tabelle. AxesBrainL SetPoint AxesBrainISO Tool Origin SetPoint AxesBrainAWL
iint iFirstElementArrayI	Primo elemento dell'Array 1 ordine
int iLastElementArrayI	Ultimo elemento dell'Array 1 ordine
int iFirstElementArrayJ	Primo elemento dell'Array 2 ordine
int iLastElementArrayJ	Ultimo elemento dell'Array 2 ordine
int * piNumberValue	Ritorna il numero di elementi
VARIANT* pVar	Variant con i valori degli elementi

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sNameArray errato 3 iFirstElementArrayI errato 4 iLastElementArrayI errato 5 iFirstElementArrayJ errato 6 iLastElementArrayJ errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
char                    sString[128];
int                     iError;
int                     iNumberGlobalsLoaded;
long                    lNumElement;
double                  pdVettValue;
long                    lLBound;
long                    lUBound;

VariantInit(&Var);
hr  =  m_GMSSrv->GetSubSystemArray(
                                1,
                                CComBSTR("SetPoint"),
                                3,
                                23,
                                43,
                                &iNumberGlobalsLoaded,
                                &Var,
                                &iError
                          );
if ( hr != S_OK )
{
        sprintf(sString, "Error GetSubSystemArray =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "GetSubSystemArray OK" );
        if ( iNumberGlobalsLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr        = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr        = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        .......
                        .......
                        delete pdVettValue;
                }
        }
}
```

○ SetSubSystemArray

Scrive le tabelle del sotto sistema

```
STDMETHODIMP SetSubSystemArray(
                                    int         iNumberSubSystem,
                                    BSTR        sNameArray,
                                    int         iFirstElementArrayI,
                                    int         iLastElementArrayI,
                                    int         iFirstElementArrayJ,
                                    int         iLastElementArrayJ,
                                    int         iNumberValue,
                                    VARIANT*    pVar,
                                    int  *      piError
                                     )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sNameArray	Nome del tabella registro, ogni sottosistema ne ha diverse di tabelle. AxesBrainL SetPoint AxesBrainISO Tool Origin SetPoint AxesBrainAWL

iint	iFirstElementArrayI	Primo elemento dell'Array 1 ordine
int	iLastElementArrayI	Ultimo elemento dell'Array 1 ordine
int	iFirstElementArrayJ	Primo elemento dell'Array 2 ordine
int	iLastElementArrayJ	Ultimo elemento dell'Array 2 ordine
int	iNumberValue	Numero elementi
VARIANT* pVar		Variant con i valori degli elementi
int * piError		Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sNameArray errato 3 iFirstElementArrayI errato 4 iLastElementArrayI errato 5 iFirstElementArrayJ errato 6 iLastElementArrayJ errato 7 iNumberValue errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 10;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 10 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr  = m_GMSSrv->SetSubSystemArray(
                                1,
                                CComBSTR("SetPoint"),
                                2,
                                2,
                                12,
                                21
                                10,
                                &Var,
                                &iError
                                  );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf(sString, "Error SetSubSystemArray =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "SetSubSystemArray OK" );
        }
}
```

○ ExecSubSystemProcess

Richiede l'esecuzione di un processo

```
STDMETHODIMP ExecSubSystemProcess(
                                    int         iNumberSubSystem,
                                    BSTR        sPathProcess,
                                    BSTR        sNameProcess,
                                    BSTR        sParameterProcess,
                                    int         iMaskMode,
                                    int         iNumberParameter,
                                    VARIANT*    pVarParameter,
                                    int         iKeyNotifyEndProcess,
                                    int  *      piError    )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sParameterProcess	Stringa parametri del processo
int iMaskMode	Bit di maschera del processo
int iNumberParameter	Numero dei parametri
VARIANT* pVarParameter	Parametri
int iKeyNotifyEndProcess	Numero della chiave con cui si vuole essere notificati o 0 se non si vuole notifica

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sParameterProcess errato 5 iNumberSubSystem errato 6 iMaskMode errato 7 iKindParameter errato 8 iNumberParameter errato 9 pVarParameter errato 10 iKeyNotifyEndProcess errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iMaskMode,
int                     iNumberParameterDouble,
int                     iKeyNotifyEndProcess,
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 10;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 10 ; ip++ )
        {
                hr          = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        iMaskMode            = 0;
        iNumberParameterDouble = 10;
        iKeyNotifyEndProcess   = 0;  // No notify
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr   = m_GMSSrv->ExecSubSystemProcess(
                                        1,
                                        CComBSTR("PathDefault"),
                                        CComBSTR("WorkAuto.pp"),
                                        CComBSTR(""),
                                        iMaskMode,
                                        iNumberParameterDouble,
                                        &Var,
                                        iKeyNotifyEndProcess,
                                        &iError
                                    );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf( sString, "Error ExecSubSystemProcess =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "ExecSubSystemProcess OK" );
        }
}
```

○ ExecSubSystemSequence

Richiede l'esecuzione di un processo

```
typedef struct  tagLOCALI
   {
   double dLocali[ 16 ];
   }  LOCALI;
```

STDMETHODIMP ExecSubSystemSequence (

```
                                        int          iNumberSubSystem,
                                        BSTR         sPathProcess,
                                        BSTR         sNameProcess,
                                        BSTR         sParameterProcess,
                                        int          iMaskMode,
                                        int          iNumberParameter,
                                        LOCALI *     pParameter,
                                        int          iKeyNotifyEndProcess,
                                        int  *       piError    )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sParameterProcess	Stringa parametri del processo
int iMaskMode	Bit di maschera del processo
int iNumberParameter	Numero dei parametri
LOCALI * pParameter	Parametri
int iKeyNotifyEndProcess	Numero della chiave con cui si vuole essere notificati o 0 se non si vuole notifica

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sParameterProcess errato 5 iNumberSubSystem errato 6 iMaskMode errato 7 iKindParameter errato 8 iNumberParameter errato 9 pVarParameter errato 10 iKeyNotifyEndProcess errato

Esempio

```
HRESULT                 hr;
double                  dVal;
int                     iMaskMode,
int                     iNumberParameterDouble,
int                     iKeyNotifyEndProcess,
int                     iError;
char                    sString[128];
tagLOCALI               SequenceLocali;

iMaskMode                =  0;
iNumberParameterDouble  = 10;
iKeyNotifyEndProcess     = 0;  // No notify
for (int ip = 0; ip < 16; ip++ )
{
        SequenceLocali.dLocali[ip]= xxxxx;
}
hr   = m_GMSSrv-> ExecSubSystemSequence (
                                1,
                                CComBSTR("PathDefault"),
                                CComBSTR("WorkAuto.pp"),
                                CComBSTR(""),
                                iMaskMode,
                                iNumberParameterDouble,
                                &SequenceLocali,
                                iKeyNotifyEndProcess,
                                &iError
                                        );
SafeArrayDestroyData( psa  );
SafeArrayDestroyDescriptor( psa  );
if ( hr != S_OK )
{
        sprintf( sString, "Error ExecSubSystemSequence =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( " ExecSubSystemSequence  OK" );
}
```

○ KillSubSystemProcess

Sospende e cancella l'esecuzione di un processo

```
STDMETHODIMP KillSubSystemProcess(
                                    int     iNumberSubSystem,
                                    BSTR    sPathProcess,
                                    BSTR    sNameProcess,
                                    int  *  piError
                                  )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato

Esempio

```
HRESULT          hr;
int              iError;
char     sString[128];

hr  =  m_GMSSrv->KillSubSystemProcess(
                                    1,
                                    CComBSTR("PathDefault"),
                                    CComBSTR("WorkAuto.pp"),
                                    &iError
                                );
if ( hr != S_OK )
{
        sprintf(sStringa, "Error KillSubSystemProcess =%ld", iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( "KillSubSystemProcess OK" );
}
```

○ GetSubSystemRunningProcesses

Legge i processi in esecuzione nel sotto sistema

```
STDMETHODIMP GetSubSystemRunningProcesses(
                                    int         iNumberSubSystem,
                                    int    *    piNumberProcess,
                                    VARIANT*    pVarPathAndNameProcess,
                                    int *       piError
                                                )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
int* piNumberProcess	Ritorna il numero di processi del sottosistema attivi
VARIANT* pVarPathAndNameProcess	Elenco del Path + Name dei processi
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato

Esempio

```
CString                 csStringaPath;
CString                 csStringaName;
CComBSTR                bstrElem;
VARIANT                  Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
int                     iError;
char                    sString[128];
int                     iNumberProcess;
long                    lNumElement;
long                    lLBound;
long                    lUBound;

VariantInit(&Var);
hr  =  m_GMSSrv->GetSubSystemArray(
                        1,
                        &iNumberProcess,
                        &Var,
                        &iError
                          );
if ( hr != S_OK )
{
        sprintf(sString, "Error GetSubSystemRunningProcesses =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "GetSubSystemRunningProcesses OK" );
        if ( iNumberProcess )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        hr        = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr        = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip ++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringaPath = CString( bstrElem );
                                                bstrElem.Empty();
                                                ip++;
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringaName = CString( bstrElem );
                                                bstrElem.Empty();
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                }
        }
}
```

○ GetSubSystemProcessRegistry

Legge i registri di un processo

```
STDMETHODIMP GetSubSystemProcessRegistry(
                                        int      iNumberSubSystem,
                                        BSTR     sPathProcess,
                                        BSTR     sNameProcess,
                                        BSTR     sNameRegistry,
                                        int      iFirstElementRegistry,
                                        int      iLastElementRegistry,
                                        int      iKindRegistry
                                        int *    pNumberValue,
                                        VARIANT* pVar,
                                        int  *   piError
                                                        )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sNameRegistry	Nome del tabella registro, ogni sottosistema ne ha diverse di tabelle per i processi. AxesBrainL Local AxesBrainISO AxesBrainAWL
int iFirstElementRegistry	Numero del primo elemento del registro
int iLastElementRegistry	Nome dell'ultimo elemento da leggere

int iKindRegistry	Tipo di registro 1 Bit 2 Byte 3 Word 4 DWord 5 Float 6 Double 7 String
int * pNumberValue	Numero di registri letti
VARIANT* pVar	Variant con il valore dei registri
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sNameRegistry errato 5 iFirstElementRegistry errato 6 iLastElementRegistry errato 7 iKindRegistry errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
int                     iError;
int                     iNumberLocalLoaded;
long                    lNumElement;
double                  pdVettValue;
long                    lLBound;
long                    lUBound;
char                    sString[128];

VariantInit(&Var);
hr  =  m_GMSSrv->GetSubSystemProcessRegistry(
                                                1,
                                                CComBSTR("PathDefault"),
                                                CComBSTR("WorkAuto.pp"),
                                                CComBSTR("Local"),
                                                1,
                                                32,
                                                6,                  // double
                                                &iNumberLocalLoaded,
                                                &Var,
                                                &iError
                                              );
if ( hr != S_OK )
{
        sprintf(sString, "Error GetSubSystemProcessRegistry =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "GetSubSystemProcessRegistry OK" );
        if ( iNumberLocalLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr          = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr          = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        .......
                        .......
                        delete pdVettValue;
                }
        }
}
```

○ SetSubSystemProcessRegistry

Scrive i registri di un processo

```
STDMETHODIMP SetSubSystemProcessRegistry(
                                    int      iNumberSubSystem,
                                    BSTR     sPathProcess,
                                    BSTR     sNameProcess,
                                    BSTR     sNameRegistry,
                                    int      iFirstElementRegistry,
                                    int      iLastElementRegistry,
                                    int      iKindRegistry,
                                    int      iNumberValue,
                                    VARIANT* pVar,
                                    int  *   piError
                                      )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sNameRegistry	Nome del tabella registro, ogni sottosistema ne ha diverse di tabelle per i processi. AxesBrainL Local AxesBrainISO AxesBrainAWL
int iFirstElementRegistry	Numero del primo elemento del registro
int iLastElementRegistry	Nome dell'ultimo elemento da leggere

int iKindRegistry	Tipo di registro 1 Bit 2 Byte 3 Word 4 DWord 5 Float 6 Double 7 String
int iNumberValue	Numero di registri da scrivere
VARIANT* pVar	Variant con il valore dei registri
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sNameRegistry errato 5 iFirstElementRegistry errato 6 iLastElementRegistry errato 7 iKindRegistry errato 8 iNumberValue errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 10;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 10 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr         = m_GMSSrv->SetSubSystemProcessRegistry(
                                                1,
                                                CComBSTR("PathDefault"),
                                                CComBSTR("WorkAuto.pp"),
                                                CComBSTR("Local"),
                                                23,
                                                32,
                                                6,                  // double
                                                10,
                                                &Var,
                                                &iError
                                                        );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf(sString, "Error SetSubSystemProcessRegistry =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "SetSubSystemProcessRegistry OK" );
        }
}
```

○ GetSubSystemProcessParameters

Legge i parametri di un processo

```
STDMETHODIMP GetSubSystemProcessParameters(
                                int       iNumberSubSystem,
                                BSTR      sPathProcess,
                                BSTR      sNameProcess,
                                BSTR      sNameParameter,
                                int       iFirstParameter,
                                int       iLastParameter,
                                int       iKindParameter,
                                int   *   piNumberValue,
                                VARIANT*  pdVar,
                                int  *    piError    )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sNameParameter	Nome del tipo di parametri , ogni sottosistema ne ha diverse AxesBrainL ProgramCounter InstuctionInProgress AxesBrainISO ProgramCounter InstuctionInProgress AxesBrainAWL ProgramCounter InstuctionInProgress

int	iFirstParameter	Numero primo parametro
int	iLastParameter	Numero ultimo parametro
int	iKindParameter	Tipo di parametro 1 Bit 2 Byte 3 Word 4 DWord 5 Float 6 Double 7 String
int*	piNumberValue	Numero di parametri da leggere
VARIANT*	pVar	Variant con il valore dei parametri
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sNameParameter errato 5 iFirstParameter errato 6 iLastParameter errato 7 iKindParameter errato 8 iNumberSubSystem errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
int                     iError;
char                    sString[128];
int                     iNumberLocalLoaded;
long                    lNumElement;
double                  pdVettValue;
long                    lLBound;
long                    lUBound;

VariantInit(&Var);
hr  =  m_GMSSrv->GetSubSystemProcessParameters(
                                        1,
                                        CComBSTR("PathDefault"),
                                        CComBSTR("WorkAuto.pp"),
                                        CComBSTR("ProgramCounter"),
                                        1,
                                        32,
                                        &iNumberLocalLoaded,
                                        &Var,
                                        &iError
                                          );
if ( hr != S_OK )
{
        sprintf(sString, "Error GetSubSystemProcessParameters =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "GetSubSystemProcessParameters OK" );
        if ( iNumberLocalLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        .......
                        .......
                        delete pdVettValue;
                }
        }
}
```

○ SetSubSystemProcessParameters

Scrive i parametri di un processo

STDMETHODIMP SetSubSystemProcessParameters(

int	**iNumberSubSystem,**
BSTR	**sPathProcess,**
BSTR	**sNameProcess,**
BSTR	**sNameParameter,**
int	**iFirstParameter,**
int	**iLastParameter,**
int	**iKindParameter,**
int	**iNumberValueDouble,**
VARIANT*	**pdVar,**
int *	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sNameParameter	Nome del tipo di parametri , ogni sottosistema ne ha diverse AxesBrainL CloseProgram AxesBrainISO AxesBrainAWL
int iFirstParameter	Numero primo parametro
int iLastParameter	Numero ultimo parametro
int iKindParameter	Tipo di registro 1 Bit 2 Byte 3 Word

		4 DWord 5 Float 6 Double 7 String
int	iNumberValue	Numero di parametri da scrivere
VARIANT*	pVar	Variant con il valore dei parametri
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sNameParameter errato 5 iFirstParameter errato 6 iLastParameter errato 7 iKindParameter errato 8 iNumberSubSystem errato 9 iNumberValue errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt    = VT_ARRAY | VT_R8 ;
dVal      = 0;
sabound[0].cElements = 10;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 10 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr   = m_GMSSrv->SetSubSystemProcessParameters(
                                                1,
                                                CComBSTR("PathDefault"),
                                                CComBSTR("WorkAuto.pp"),
                                                CComBSTR("CallParameter"),
                                                1,
                                                10,
                                                6,                  // double
                                                10,
                                                &Var,
                                                &iError
                                              );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf( sString, "Error SetSubSystemProcessParameters =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "SetSubSystemProcessParameters OK" );
        }
}
```

○ ExecSubSystemInstruction

Richiede l'esecuzione di una istruzione

```
STDMETHODIMP ExecSubSystemInstruction(
                                    int         iNumberSubSystem,
                                    BSTR        sNameInstruction,
                                    BSTR        sParameterInstruction,
                                    int         iMaskMode,
                                    int         iNumberParameter,
                                    VARIANT*    pdVarParameter,
                                    int         iKeyInstructionIdentify,
                                    int         iKeyNotifyEndInstruction,
                                    int  *      piError
                                                )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sNameInstruction	Stringa istruzione
BSTR sParameterInstruction	Stringa parametri istruzione
int iMaskMode	Maschera per esecuzione istruzione
int iNumberParameter	Numero di parametri
VARIANT* pVarParameter	Variant valore di parametri
int iKeyInstructionIdentify	Chiave d'identificazione
int iKeyNotifyEndInstruction	Numero della chiave con cui si vuole essere notificati o 0 se non si vuole notifica

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sNameInstruction errato 3 sParameterInstruction errato 4 iMaskMode errato 5 iNumberParameter errato 6 iKeyInstructionIdentify errato 7 iKeyNotifyEndInstruction errato

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iMaskMode,
int                     iNumberParameterDouble,
int                     iKeyInstructionIdentify,
int                     iKeyNotifyEndProcess,
int                     iError;
char                    sString[128];

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 16;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 16 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        iMaskMode             = 0;
        iNumberParameterDouble  = 16;
        iKeyInstructionIdentify = 4356;
        iKeyNotifyEndProcess    = 0;  // No notify
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr   = m_GMSSrv->ExecSubSystemInstruction(
                                        1,
                                        CComBSTR("-MOV/A1,100,B1,200"),
                                        CComBSTR(""),
                                        iMaskMode,
                                        iNumberParameterDouble,
                                        &Var,
                                        iKeyInstructionIdentify,
                                        iKeyNotifyEndProcess,
                                        &iError
                                          );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf( sString, "Error ExecSubSystemInstruction =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "ExecSubSystemInstruction OK" );
        }
}
```

○ KillSubSystemInstruction

Sospende e cancella l'esecuzione di una istruzione

```
STDMETHODIMP KillSubSystemInstruction(
                                int         iNumberSubSystem,
                                int         iKeyInstructionIdentify,
                                int  *      piError
                                )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem — Tipo di identificazione del sottosistema
- 1 AxesBrainL
- 2 AxesBrainISO
- 3 AxesBrainAWL
- 101 Primo sottosistema OEMCapability
- 102 Secondo sottosistema OEMCapability
- ...
- ...
- 109 Nono sottosistema OEMCapability

int iKeyInstructionIdentify — Chiave d'identificazione dell'istruzione

int * piError — Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:
- 1 iNumberSubSystem errato
- 2 iKeyInstructionIdentify errato

Esempio

```
HRESULT         hr;
int             iError;
int             iKeyInstructionIdentify,
char            sString[128];

iKeyInstructionIdentify = 1345;
hr  =  m_GMSSrv->KillSubSystemProcess( 1,
                                       iKeyInstructionIdentify,
                                       &iError
                                     );
if ( hr != S_OK )
{
        sprintf( sString, "Error KillSubSystemInstruction =%ld", iError );
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "KillSubSystemInstruction OK" );
}
```

○ SendOrClearSubSystemEventSignal

Usata come sincronizzazione semplice con il sottosistema segnalando un evento, se il processo del sottosistema era in attesa dell'evento viene riattivato, altrimenti l'evento viene memorizzato.

```
STDMETHODIMP SendOrClearSubSystemEventSignal(
                                    int         iNumberSubSystem,
                                    BSTR        sNameClassEvent,
                                    int         iSetOrClear,
                                    int         iNumberEvent,
                                    VARIANT*    piVarEvent,
                                    int  *      piError
                                                )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sNameClassEvent	Nome della classe eventi AxesBrainL Event AxesBrainISO Event AxesBrainAWL Event
int iSetOrClear	1 indica il set dell'evento 0 il reset
int iNumberEvent	Numero di eventi da impostare
VARIANT* piVarEvent	Variant con il numero del evento
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sNameClassEvent errato 3 iSetOrClear errato 4 iNumberEvent errato

Note:
come se se avesswe mandato OnRequestServiceEvent(
BSTR sNameClassEvent,
-1 ,
-199199 ,
int *iNumberEvent,
VARIANT* piVarEvent,
)

Esempio

```
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
char                    sString[128];
int                     iVal;
int                     iNumberParameter,
int                     iError;
long                    ip;
int                     iSetOrClear,


VariantInit(&Var);
Var.vt    = VT_ARRAY | VT_I4 ;
iSetOrClear   = 1               // Set event
iVal  = 1;
sabound[0].cElements = 1;
sabound[0].lLbound    = 0;
psa = SafeArrayCreate( VT_I4 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 1 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &iVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        iNumberParameter  = 1;
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr         = m_GMSSrv->SendOrClearSubSystemEventSignal(
                                                        1,
                                                        CComBSTR( "Event" ),
                                                        iSetOrClear,
                                                        iNumberParameter,
                                                        &Var,
                                                        &iError
                                                                );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf( sString, "Error SendOrClearSubSystemEventSignal =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "SendOrClearSubSystemEventSignal OK" );
        }
}
```

○ WriteServiceParameterAndContinue

Il processo del sotto sistema si pone in attesa inviando una richiesta si servizio, sarà appunto il l'esecutore del servizio a chiamare il metodo:
"WriteServiceParameterAndContinue"

STDMETHODIMP WriteServiceParameterAndContinue(

int	**iNumberSubSystem,**
BSTR	**sMsg,**
int	**lParam1,**
int	**lKey,**
int	**iNumberParameter,**
ELEMENTS *	**pElements,**
int *	**piError**

)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sMsg	Messaggio
int lParam1	Parametro
int lKey	Chiave di riconoscimento
int iNumberParameter	Numero di parametri
ELEMENTS * pElements	Struttura pElementi
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sMsg errato 3 lParam1 errato 4 lKey errato 5 iNumberParameter errato

Esempio

```
HRESULT CNotify::OnRequestServiceEvent(  BSTR      sMsg,
                                         int       lParam1 ,
                                         int       lKey,
                                         int       iNumberParameter.
                                         VARIANT * pVar
                                        )
{
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iNumberParameterDouble,
int                     iError;
double *                 pdVettValue;
int                     lParam1,
int                     lKey,
CString                 csMsg;
long                    ip;
char                    sString[128];

csMsg = CString(sMsg);
if ( iNumberParameter )
{
        psa  = pVar->parray ;
        if ( psa )
        {
                pdVettValue = new double[iNumberGlobalsLoaded];
                hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                lNumElement = lUBound - lLBound + 1;
                if ( lNumElement )
                {
                        if ( !FAILED ( hr ))
                        {
                                for ( ip=0 ; ip < (long)lNumElement; ip++ )
                                {
                                        SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                }
                        }
                }
                SafeArrayDestroyData( psa  );
                SafeArrayDestroyDescriptor( psa  );
                .......
                .......
                delete pdVettValue;
        }
}

VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 16;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 16 ; ip++ )
        {
                hr = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
```

```
                        break;
                }
        }
}
if ( psa )
{
        iNumberParameterDouble  = 16;
        Var.vt     = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        hr   = m_GMSSrv->WriteServiceParameterAndContinue (
                                                        1,
                                                        CComBSTR( csMsg ),
                                                        lParam1,
                                                        lKey,
                                                        iNumberParameterDouble,
                                                        &Var,
                                                        &iError
                                                                );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf(sString, "Error WriteServiceParameterAndContinue =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "WriteServiceParameterAndContinue OK" );
        }
}
```

Note:

```
OnRequestServiceEvent(
                        BSTR    sMsg,
                        int     lParam1,
                        int     lKey,
                        int       *       iNumberParameterDouble,
                       VARIANT* pdVarParameter,
                )
```

OnRequestServiceEvent Richiede un servizio agli utenti connessi,
l'applicazione dell'utente predisposto ad eseguire la richiesta,
una volta che ha terminato il compito deve segnalare tramite la funzione
"WriteServiceParameterAndContinue" che l'operazione è stata eseguita.

○ WriteServiceElementAndContinue

Il processo del sotto sistema si pone in attesa inviando una richiesta si servizio, sarà appunto il l'esecutore del servizio a chiamare il metodo:
"WriteServiceElementAndContinue"

```
typedef struct  tagELEMENTS
{
   double dElements[ 32 ];
}      ELEMENTS;
```

STDMETHODIMP WriteServiceElementAndContinue (

```
                                    int         iNumberSubSystem,
                                    BSTR        sMsg,
                                    int         lParam1,
                                    int         lKey,
                                    int         iNumberParameter,
                                    VARIANT*    pVarParameter,
                                    int  *      piError
                                          )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sMsg	Messaggio
int lParam1	Parametro
int lKey	Chiave di riconoscimento
int iNumberParameter	Numero di parametri
VARIANT* pVarParameter	Variant con il valore di ogni parametro
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sMsg errato

3 lParam1 errato
4 lKey errato
5 iNumberParameter errato

Esempio

```
HRESULT CNotify::OnRequestServiceEvent(  BSTR     sMsg,
                                         int      lParam1 ,
                                         int      lKey,
                                         int      iNumberParameter.
                                         VARIANT * pVar
                                        )
{

VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iNumberParameterDouble,
int                     iError;
double *                 pdVettValue;
int                     lParam1,
int                     lKey,
CString                 csMsg;
long                    ip;
char                    sString[128];

csMsg = CString(sMsg);
if ( iNumberParameter )
{
        psa  = pVar->parray ;
        if ( psa )
        {
                pdVettValue = new double[iNumberGlobalsLoaded];
                hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                lNumElement = lUBound - lLBound + 1;
                if ( lNumElement )
                {
                        if ( !FAILED ( hr ))
                        {
                                for ( ip=0 ; ip < (long)lNumElement; ip++ )
                                {
                                        SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                }
                        }
                }
                SafeArrayDestroyData( psa  );
                SafeArrayDestroyDescriptor( psa  );
                .......
                .......
                delete pdVettValue;
        }
}

SAri * pAri = m_aSARI.GetHead();

long            ip;
int             iError;
ELEMENTS        Elements;

for ( ip=0; ip< pAri->iNumeroOut ; ip++ )
{
        Elements.dElements[ip] = pAri->dValori[ip];
}

if (m_pDG->m_pServer->WriteServiceElementAndContinue(
                                                        1,
                                                                CComBSTR(pAri->sMsg),
                                                                pAri->iAmbiente,
```

```
                                                                    pAri->iChiave,
                                                                    pAri->iNumeroOut,
                                                                    &Elements,
                                                                    &iError) !=  S_OK
                                                                  )
{
        AfxMessageBox(CaricaStringa(IDS_ERRORE_WRITE));
}
delete pAri;
m_aSARI.RemoveHead();
```

Note:

```
OnRequestServiceEvent(
                            BSTR    sMsg,
                            int     lParam1,
                            int     lKey,
                            int       *       iNumberParameterDouble,
                           VARIANT* pdVarParameter,
                    )
```

OnRequestServiceEvent Richiede un servizio agli utenti connessi,
l'applicazione dell'utente predisposto ad eseguire la richiesta,
una volta che ha terminato il compito deve segnalare tramite la funzione
"WriteServiceElementAndContinue" che l'operazione è stata eseguita.

○ SetBreakPoint

Seleziona o rimuove delle linee di breakpoint

```
STDMETHODIMP SetBreakPoint (
                                int     iNumberSubSystem,
                                int     iMode,
                                BSTR    sPathProcess,
                                BSTR    sNameProcess,
                                int     iLineNumber,
                                int *   piError
                            )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
int	ìMode	Modalità del comando 0 Disattiva Debug 1 Attiva Debug 2 Aggiungi un breakpoint 3 Toglie un breakpoint 99 Toglie tutti i breakpoint
BSTR	sPathProcess	Path delle directory del processo o PathDefault
BSTR	sNameProcess	Nome del processo
int	iLineNumber	Numero della linea del breakpoint
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 iMode non corretto 3 Break point non trovato 4 Break point gia esistente

5	iNumberParameterDouble errato

Esempio

```
m_pDG->m_pServer->SetBreakPoint(1,
                                    2, // Set BreakPoint
                                    CComBSTR( m_pDG->m_sPathPartProgramGP ),
                                    CComBSTR( m_sNomeProgramma ),
                                    curPos.y,
                                    &iErrore );


m_pDG->m_pServer->SetBreakPoint( 1,
                                    99, // Reset All BreakPoint
                                    CComBSTR( _T("") ),
                                    CComBSTR( _T("") ),
                                    0,
                                    &iErrore );
```

○ GetSubSystemBreakPointsProcesses

Richiede i break points attivati

STDMETHODIMP GetSubSystemBreakPointsProcesses (

int	**iNumberSubSystem,**
int *	**piBreakPoint,**
VARIANT *	**pVarPathAndNameProcess,**
VARIANT *	**pVarRecordNumber,**
int *	**piError**

)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
int *	piBreakPoint	Numero dei break points
VARIANT *	pVarPathAndNameProcess	Stringhe con PathAndName file
VARIANT *	pVarRecordNumber	double con il numero di riga
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 SubSystem non attivo 2 iNumberSubSystem errato 3 sCommandStr errato 4 sParameterStr errato 5 iNumberParameterDouble errato

Esempio

```
VARIANT                 VarPathAndNameProcess;
VARIANT                 VarRecordNumber;
int                     iNumeroValoriRitorno;
int                     iErrore;
HRESULT                 hr;
long                    ip;
long                    lNumElement;
long                    lLBound;
long                    lUBound;
SAFEARRAY FAR*          psa;
SAFEARRAY FAR*          psaRecordNumber;
CComBSTR                bstr;
CString                 csStringa;
double                  dVal;
CString                 sPathNameProgramma;
int                     iContaBP;

iContaBP = 0;
*iNumBP = 0;
int * piBP = NULL;
sPathNameProgramma = m_pDG->m_sPathPartProgramGP + m_sNomeProgramma;
sPathNameProgramma.MakeUpper();

VariantInit(&VarPathAndNameProcess);
VariantInit(&VarRecordNumber);

hr = m_pDG->m_pServer->GetSubSystemBreakPointsProcesses( 1,
                                                         &iNumeroValoriRitorno,
                                                         &VarPathAndNameProcess,
                                                         &VarRecordNumber,
                                                         &iErrore
                                                       ) ;
if ( hr == S_OK )
{
        psa = VarPathAndNameProcess.parray;
        psaRecordNumber = VarRecordNumber.parray;
        if ( psa && psaRecordNumber )
        {
                if ( VarPathAndNameProcess.vt  == ( VT_ARRAY | VT_BSTR)  )
                {
                        hr= SafeArrayGetLBound( psa, 1, &lLBound );
                        hr= SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;

                        if ( lNumElement )
                        {
                                if ( ! FAILED ( hr ) )
                                {
                                        piBP = new int[lNumElement];

                                        for ( ip=0; ip < (long)lNumElement ; ip++ )
                                        {
                                                hr= SafeArrayGetElement( psa , &ip, &bstr );
                                                if ( ! FAILED ( hr ) )
                                                {
                                                        csStringa = bstr;
                                                        bstr.Empty();

                                                        csStringa.MakeUpper();

                                                        if (csStringa == sPathNameProgramma)
                                                        {
                                                                SafeArrayGetElement(
                                                                                psaRecordNumber,
```

```
                                                                    &ip,
                                                                    &dVal );

                                                            piBP[iContaBP] = (int)dVal;

                                                            iContaBP ++;
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                        SafeArrayDestroyData( psa );
                                        SafeArrayDestroyDescriptor( psa );
                                        SafeArrayDestroyData( psaRecordNumber );
                                        SafeArrayDestroyDescriptor( psaRecordNumber );
                                    }
                                }
                            }
                        }
                    }
                }
                *iNumBP = iContaBP;
```

○ GetValueDebug

Ottiene il valore di una stringa marcata dentro una istruzione

STDMETHODIMP GetValueDebug(int iNumberSubSystem,
BSTR sPathProcess,
BSTR sNameProcess,
BSTR sLinea,
BSTR sStringaSelezionata,
int iNumLinea,
double * dRetValue,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR	sPathProcess	Path della directory del part program
BSTR	sNameProcess	Nome del processo
BSTR	sLinea	Stringa istruzione
BSTR	sStringaSelezionata	Stringa selezionata
int	iNumLinea	Numero linea nel file
double*	dRetValue	Ritorna il numero double del valore della stringa selezionata
int *	piError	Ritorna il tipo di errore ed in particolare: 0 Trovato il valore dRetValue significativo -1 Non trovato valore

○ GetSubSystemProcessRecord

Legge i registri di un processo

STDMETHODIMP GetSubSystemProcessRegistry(
int iNumberSubSystem,
BSTR sPathProcess,
BSTR sNameProcess,
BSTR sNameRegistry,
BSTR * sValue,
ELEMENTS * pElements,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sPathProcess	Path delle directory del processo o PathDefault
BSTR sNameProcess	Nome del processo
BSTR sNameRegistry	Nome del tabella registro, ogni sottosistema ne ha diverse di tabelle per i processi. AxesBrainL Local AxesBrainISO AxesBrainAWL
BSTR* SValue	Ritorna il **record in esecuzione**
ELEMENTS * pElements	Ritorna il vettore caico dei seguenti valori:

pElements->dElements[0] = -1 o Numero blocco
pElements->dElements[1] = SistemaRiferimento[iS].iIDOriginRunning
pElements->dElements[2] = SistemaRiferimento[iS].iIDToolRunning
pElements->dElements[3] = SistemaRiferimento[iS].iM
pElements->dElements[4] = SistemaRiferimento[iS].iG
pElements->dElements[5] = SistemaRiferimento[iS].dSpeed
pElements->dElements[6] = SistemaRiferimento[iS].dFeed
pElements->dElements[7] = SistemaRiferimento[iS].Tool[Tool].dFreeValue1
pElements->dElements[8] = SistemaRiferimento[iS].Tool[Tool].dFreeValue2;
pElements->dElements[9] = SistemaRiferimento[iS].Tool[Tool].dFreeValue3;
pElements->dElements[10]= SistemaRiferimento[iS].Tool[Tool].dFreeValue4;
pElements->dElements[11]= SistemaRiferimento[iS].dTeoricoX
pElements->dElements[12]= SistemaRiferimento[iS].dTeoricoY
pElements->dElements[13]= SistemaRiferimento[iS].dTeoricoZ
pElements->dElements[14]= SistemaRiferimento[iS].dTeoricoA
pElements->dElements[15]= SistemaRiferimento[iS].dTeoricoB
pElements->dElements[16]= SistemaRiferimento[iS].dTeoricoC
pElements->dElements[17]= SistemaRiferimento[iS].dTeoricoU
pElements->dElements[18]= SistemaRiferimento[iS].dTeoricoV
pElements->dElements[19]= SistemaRiferimento[iS].dTeoricoW
pElements->dElements[20]= SistemaRiferimento[iS].Tool[Tool].dRT;
pElements->dElements[21]= SistemaRiferimento[iS].Tool[Tool].dLC;
pElements->dElements[22]= SistemaRiferimento[iS].dTeoricoXP;
pElements->dElements[23]= SistemaRiferimento[iS].dTeoricoYP;
pElements->dElements[24]= SistemaRiferimento[iS].dTeoricoZP;
pElements->dElements[25]= SistemaRiferimento[iS].dTeoricoAP;
pElements->dElements[26]= SistemaRiferimento[iS].dTeoricoBP;
pElements->dElements[27]= SistemaRiferimento[iS].dTeoricoCP;
pElements->dElements[28]= SistemaRiferimento[iS].dTeoricoUP;
pElements->dElements[29]= SistemaRiferimento[iS].dTeoricoVP;
pElements->dElements[30]= SistemaRiferimento[iS].dTeoricoWP;
pElements->dElements[31]= SistemaRiferimento[iS].iIDOriginRunning;

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sPathProcess errato 3 sNameProcess errato 4 sNameRegistry errato

Esempio

```
CString csStringa;
BSTR sTmp;
CString sIDHead;
CString sStringa;


tagELEMENTS  Elements;

sIDHead.Format(_T("%d"), m_pActiveHead->m_iIDTesta);

if (m_pPagina->m_pDG->m_pServer->GetSubSystemProcessRecord( -2,
                                                            CComBSTR(""),

                                                            CComBSTR(sIDHead),

                                                            CComBSTR(_T("INSTRUCTIONINPROGRESS")),

                                                            &sTmp,

                                                            &Elements,

                                                            &iErrore

                                                            ) == S_OK )
{
        csStringa = OLE2A(sTmp);
}
else
{
        csStringa.Empty();;
        for (int i=0; i<32; i++)
        {
                Elements.dElements[i] = 0;
        }
}
m_Static_Valore_Blocco.SetWindowText(csStringa);

int iN                  = (int)Elements.dElements[0];
int iT                  = (int)Elements.dElements[2];
int iM                  = (int)Elements.dElements[3];
int iG                  = (int)Elements.dElements[4];
int iF                  = (int)Elements.dElements[6];
double dTF1             = Elements.dElements[7];
double dTF2             = Elements.dElements[8];
double dCR              = Elements.dElements[20];
double dCL              = Elements.dElements[21];
int iO                  = (int)Elements.dElements[31];

if ( iG == 84 )
{
        if ( Elements.dElements[5] )
        {
                if (m_dOldQuotaSPDSS !=  Elements.dElements[5])
                {
```

```
                        m_dOldQuotaSPDSS =  Elements.dElements[5];

                        sStringa.Format(_T("%.0lf"),  Elements.dElements[5]);
                        m_Static_Valore_S.SetWindowText(sStringa);
                }
        }
}
else
{
        if (m_dOldQuotaSPDSS )
        {
                m_dOldQuotaSPD   =  m_dOldQuotaSPDSS;
                _dOldQuotaSPDSS =  0;
        }
}
```

Capitolo

5

6.Segnalazione d'impianto

Quando il sistema "AxesBrain" rileva anomalie, stati d'errore o segnalazioni le registra in un file di LOG e le segnale per mezzo di un FIRE apposito, le applicazioni possono catturare queste segnalazioni per poterle a loro volta segnalare in modo grafico.

1-GetEventsLogFile	Legge le segnalazioni del LOGFILE
2-DeleteEventsLogFile	Cancella le segnalazioni dal LOGFILE

Note:

OnSystemEvent	Evento inviato da parte del sistema per segnalare anomalie

○ GetEventsLogFile

Legge le segnalazioni del LGFILE

```
STDMETHODIMP GetEventsLOGFILE (
                                int          iKindCommand,
                                int          iNumberMsg,
                                int    *     piNumberMsgRetrive,
                                VARIANT *    pMsg,
                                int *        piNumberElementMsg,
                                int *        piError
                                     )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iKindCommand	Tipo di comando per ottenere il messaggio 0 numero identificato da **iNumberMsg** 1 Primo messaggio 9 Ultimo messaggio
int iNumberMsg	Numero del messaggio (iKindCommand = 0)
int* piNumberMsgRetrieve	Ritorna il numero del messaggio ottenuto
VARIANT * pMsg	VARIANT di BSTR degli elementi del messaggio
int * piNumberElementMsg	Ritorna il numero di elementi del VARIANT * pMsgEvents
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Comando errato 2 Numero di messaggio errato

Esempio

```
VARIANT         Var;
int             iKindCommand;
int             iNumberMsgRetrieve;
int             iNumberElementMsg;
int             iError;
HRESULT         hr;

VariantInit(&Var);
Var.vt        = VT_ARRAY | VT_BSTR;
Var.parray  = NULL;
iKindCommand = 9;    // Last Msg
hr = m_GMSSrv->GetEventsLOGFILE(      iKindCommand.
                                      0 ,
                                      &iNumberMsgRetrieve,
                                      &Var,
                                      &iNumberElementMsg,
                                      &iError
                                  );
                );
 if ( hr != S_OK )
{
        char sString[128];

        sprintf(sStringa, ”Error GetEventsLOGFILE =%ld”, iError);
        AfxMessageBox( sStringa );
}
else
{
        AfxMessageBox( “GetEventsLOGFILE OK” );
}
```

○ DeleteEventsLOGFILE

Cancella le segnalazioni dal LOGFILE

STDMETHODIMP DeleteEventsLOGFILE (
int iNumberMsg,
int * piError)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberMsg	Numero del messaggio da cancellare se –1 Cancella tutti i messaggi.
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Elemento non trovato

Esempio

```
HRESULT         hr;
int             iError;

hr =  m_GMSSrv->DeleteEventsLOGFILE(  -1,
                                      &iError    );
 if ( hr != S_OK )
{
        AfxMessageBox( "DeleteEventsLOGFILE non OK" );
}
else
{
        AfxMessageBox( "DeleteEventsLOGFILE OK" );
}
```

Capitolo

6

7.Configurazione del sistema

La configurazione delle risorse, dei dispositivi di controllo ed altre caratteristiche di sistema sono registrate nel "AxesBrain", apposite funzione ne permettono la lettura, il cambiamento, la cancellazione e l'aggiunta di nuove configurazioni.

1-ReadSystemConfiguration	Legge la configurazione di sistema
2-EditSystemConfiguration	Scrive la configurazione di sistema
3-GetNumberAndNameKindElement	Legge il numero e i nomi di un tipo di elementi
3-ReadElementConfiguration	Legge la configurazione di un elemento
4-EditElementConfiguration	Cambia,cancella o aggiunge la configurazione di un elemento

○ ReadSystemConfiguration

Legge la configurazione di sistema

```
STDMETHODIMP ReadSystemConfiguration(
                                    int         iKindObject,
                                    BSTR        sNameSession,
                                    int    *    piNumberBSTR,
                                    VARIANT*    pVarBSTR,
                                    int  *      piError
                                          )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iKindObject	Tipo di identificazione del sottosistema 1 Ottieni le sessioni 2 Ottieni tutti i campi di una sessione
BSTR sNameSession	Nome della sessione
int * piNumberBSTR	Numero delle stringhe BSTR
VARIANT* pVarBSTR	Parametri BSTR
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iKindObject errato 2 sNameSession errato

Esempio

```
CString                 csStringDescription;
CString                 csStringDefinition;
CComBSTR                bstrElem;
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
int                     iError;
char                    sString[128];
int                     iNumberStringLoaded;
long                    lNumElement;
long                    lLBound;
long                    lUBound;

VariantInit(&Var);
hr  =  m_GMSSrv->ReadSystemConfiguration(
                                3,     //   Field of session
                                CComBSTR("BaseConfiguration"),
                                &iNumberStringLoaded,
                                &Var,
                                &iError
                            );
if ( hr != S_OK )
{
        sprintf(sString, "Error ReadSystemConfiguration =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "ReadSystemConfiguration OK" );
        if ( iNumberLocalLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr        = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr        = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringDescription = CString( bstrElem );
                                                bstrElem.Empty();
                                                ip++;
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringDefinition = CString( bstrElem );
                                                bstrElem.Empty();
                                                .....
                                                .....
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        delete pdVettValue;
                }
        }
}
```

○ EditSystemConfiguration

Scrive la configurazione di sistema

```
STDMETHODIMP EditSystemConfiguration(
                                        BSTR      sNameSession,
                                        BSTR      sEntry
                                        BSTR      sParameter
                                        int  *    piError
                                              )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

BSTR sNameSession	Nome della sessione
BSTR sEntry	Entry
BSTR sParameter	Parametri della entry
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 Aggiornamento non avvenuto

Esempio

```
CString                 csStringDescription;
CString                 csStringDefinition;
BSTR                    bstr = NULL;
char                    sString[128];

hr  =  m_GMSSrv->EditSystemConfiguration(
                                        CComBSTR("BaseConfiguration"),
                                        CComBSTR("Campionatura”
                                        CComBSTR("100"),
                                        &iError
                                    );
if ( hr != S_OK )
{
        sprintf(sString, "Error EditSystemConfiguration =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "EditSystemConfiguration OK" );
}
```

○ GetNumberAndNameKindElement

Legge il numero e i nomi di un tipo di elementi

STDMETHODIMP GetNumberAndNameKindElement(
int iKindOfElement,
int * piNumberBSTR,
VARIANT* pVarBSTR,
int * piError
)

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iKindOfElement — Tipo di elementi
- 1 Assi fisici
- 2 Assi virtuali
- 3 Mandrini
- 4 Input digitali fisici e virtuali
- 5 Output digitali fisici e virtuali
- 6 Input Analogici
- 7 Output Analogici
- 14 Input digitali fisici
- 15 Output digitali fisici

int * piNumberBSTR — Numero delle stringhe BSTR

VARIANT* pVarBSTR — Parametri BSTR

int * piError — Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare:
- 1 iKindOfElement errato

Esempio

```
CString                 csStringDescription;
CString                 csStringDefinition;
CComBSTR                bstrElem;
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
int                     iError;
char                    sString[128];
int                     iNumberStringLoaded;
long                    lNumElement;
long                    lLBound;
long                    lUBound;
int                     iKindOfElement

VariantInit(&Var);
iKindOfElement     = 1; // Phisical Axis
hr  =  m_GMSSrv->GetNumberAndNameKindElement(
                        iKindOfElement,
                        &iNumberStringLoaded,
                        &Var,
                        &iError
                        );
if ( hr != S_OK )
{
        sprintf(sString, "Error GetNumberAndNameKindElement =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        if ( iNumberLocalLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringDescription = CString( bstrElem );
                                                bstrElem.Empty();
                                                ip++;
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringDefinition = CString( bstrElem );
                                                bstrElem.Empty();
                                                .....
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        delete pdVettValue;
                }
        }
}
```

○ ReadElementConfiguration

Legge la configurazione di un elemento

```
STDMETHODIMP ReadElementConfiguration(
                                        int     iKindOfElement,
                                        int     iModeNumberOrName,
                                        int     iNumber,
                                        BSTR sName,
                                        int     * piNumberBSTR,
                                        VARIANT* pVarBSTR,
                                        int  *  piError
                                              )
```

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindOfElement	Tipo di elemento
		1 Asse fisico
		2 Asse virtuale
		3 Mandrino
		4 Input digitale
		5 Output digitale
		6 Input Analogico
		7 Output Analogico
int	iModeNumberOrName	Modo di identificazione
		1 Per numero
		2 Per nome
int	iNumber	Numero di identificazione
BSTR	sName	Nome di identificazione
int *	piNumberBSTR	Numero delle stringhe BSTR
VARIANT*	pVarBSTR	Parametri BSTR

int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iKindOfElement errato 2 iModeNumberOrName errato 3 iNumber errato 4 sName errato

Esempio

```
CString                 csStringDescription;
CString                 csStringDefinition;
CComBSTR                bstrElem;
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAY FAR*          psa;
int                     iError;
char                    sString[128];
int                     iNumberStringLoaded;
long                    lNumElement;
long                    lLBound;
long                    lUBound;
int                     iKindOfElement;
int                     iModeNumberOrName;
int                     iNumber;

VariantInit(&Var);
iKindOfElement     = 1; // Phisical Axis
iModeNumberOrName  = 1; // By Number
iNumber                    = 3; // Third Phisical Axis
hr  = m_GMSSrv->ReadElementConfiguration        (
                                    iKindOfElement
                                    iModeNumberOrName
                                    iNumber,
                                    CComBSTR(""),
                                    &iNumberStringLoaded,
                                    &Var,
                                    &iError
                                    );
```

```
if ( hr != S_OK )
{
        sprintf(sString, "Error ReadElementConfiguration =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "ReadElementConfiguration OK" );
        if ( iNumberLocalLoaded )
        {
                psa  = Var.parray ;
                if ( psa )
                {
                        pdVettValue = new double[iNumberGlobalsLoaded];
                        hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                        hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                        lNumElement = lUBound - lLBound + 1;
                        if ( lNumElement )
                        {
                                if ( !FAILED ( hr ))
                                {
                                        for ( long ip=0 ; ip < (long)lNumElement; ip++ )
                                        {
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringDescription = CString( bstrElem );
                                                bstrElem.Empty();
                                                ip++;
                                                SafeArrayGetElement( psa, &ip , &bstrElem );
                                                csStringDefinition = CString( bstrElem );
                                                bstrElem.Empty();
                                                .....
                                                .....
                                        }
                                }
                        }
                        SafeArrayDestroyData( psa  );
                        SafeArrayDestroyDescriptor( psa  );
                        delete pdVettValue;
                }
        }
}
```

○ EditElementConfiguration

Cambia,cancella o aggiunge la configurazione di un elemento

STDMETHODIMP EditElementConfiguration(

int	**iKindOfElement,**
int	**iModeNumberOrName,**
int	**iNumber,**
BSTR	**sName,**
int	**iMode,**
BSTR	**sEntry,**
BSTR	**sParameter,**
int *	**piError)**

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int	iKindOfElement	Tipo di elemento 1 Asse fisico 2 Asse virtuale 3 Mandrino 4 Input digitale 5 Output digitale 6 Input Analogico 7 Output Analogico
int	iModeNumberOrName	Modo di identificazione 1 Per numero 2 Per nome
int	iNumber	Numero di identificazione
BSTR	sName	Nome di identificazione
int	iMode	Tipo di operazione 1 Modifica 2 Inserisci 3 Cancella l'elemento
BSTR	sEntry	Entry
BSTR	sParameter	Parametri dell'Entry
int *	piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iKindOfElement errato 2 iModeNumberOrName errato 3 iNumber errato 4 sName errato

Esempio

```
int                         iModeNumberOrName;
int                         iNumber;
int                         iMode;


iKindOfElement     = 1; // Phisical Axis
iModeNumberOrName  = 1; // By Number
iNumber                    = 3; // Third Phisical Axis
iMode                      = 1; // Modify
hr        =  m_GMSSrv->EditElementConfiguration(
                                          iKindOfElement
                                          iModeNumberOrName
                                          iNumber,
                                          CComBSTR(""),
                                          iMode,
                                          CComBSTR("Kind_axis_0"),
                                          CComBSTR("0x00001001"),
                                          &iError
                                            );
if ( hr != S_OK )
{
        sprintf(sStringa, "Error EditElementConfiguration =%ld", iError);
        AfxMessageBox( sString );
}
else
{
        AfxMessageBox( "EditElementConfiguration OK" );
}
```

Capitolo

7

8.Eventi dal server

Il server invia in modo asincrono, delle segnalazioni agli utenti connessi attraverso degli "EVENTI". Gli eventi si suddividono in tre tipologie:

- Eventi di segnalazione di anomalie del sistema
- Eventi di segnalazione di richieste asincrone espletate
- Eventi di richieste da parte del Server, per ottenere dei servizi dai Client

Evento	Segnalazione dal sistema con double
OnSystemEvent	Segnalazione dal sistema con Variant
OnCompleteCommand	Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta) (Variant)
OnCompleteElement	Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta) (Vettore di double)
OnRequestServiceEvent	Richiede un servizio dal applicativo utente in termini di dati, che quando reperiti verranno forniti tramite l'opportuna funzione di scrittura (Variant)
OnRequestServiceEventElement	Richiede un servizio dal applicativo utente in termini di dati, che quando reperiti verranno forniti tramite l'opportuna funzione di scrittura (Vettore di double).
EventError	Segnalazione d'errore dal sistema

○ Evento

Segnalazione dal sistema

HRESULT Evento(short iEvento,
BSTR sMsg,
double dValore
)

Parametri

short	iEvento	Evento	
		EVENTO_TEST	0
		EVENTO_PP_GP	1
		EVENTO_INIZIO_FO	2
		EVENTO_VISIONE	3
		EVENTO_ELABORAZIONE_VISIONE	4
		EVENTO_ERRORE	5
		EVENTO_ATTESA_DA_TASTIERA	6
		EVENTO_DISPLAY	7
		EVENTO_TRACE	8
		EVENTO_BREAPOINT	9
BSTR	sMsg	Messaggio	
double	dValore	Valore double inviato	

Esempio

```
HRESULT CNotifica:: Evento (short iEvento, BSTR sMsg, double dValore)
{
	bool bPropaga = true;

	CString sMessaggio = CString(sMsg);

	switch (iEvento)
	{
		case EVENTO_TEST:
		{
			AfxMessageBox(sMessaggio);
		}
		break;

		case EVENTO_DISPLAY:
		{
			m_pDG->m_pVisualizzatoreMsg->VisualizzaMessaggio(sMessaggio);
			bPropaga = false;
		}
		break;

		case EVENTO_ERRORE:
		{
			if (theApp.m_pDlgErrori)
			{
				theApp.m_pDlgErrori->ScriviErrore(sMessaggio);
			}
			else
			{
				theApp.m_pDlgErrori = new CDlgErrori;
				theApp.m_pDlgErrori->Create(IDD_DLG_ERRORI);
				theApp.m_pDlgErrori->ShowWindow(SW_SHOW);
				theApp.m_pDlgErrori->ScriviErrore(sMessaggio);
			}
			bPropaga = false;
		}
		break;

		case EVENTO_ATTESA_DA_TASTIERA:
		{
			if (theApp.m_pDlgTastiera)
			{
				theApp.m_pDlgTastiera->ScriviDicitura(sMessaggio);
			}
			else
			{
				theApp.m_pDlgTastiera = new CDlgTastiera;
				theApp.m_pDlgTastiera->m_sMsg = sMessaggio ;
				theApp.m_pDlgTastiera->m_iAmbiente = -9696 ;
				theApp.m_pDlgTastiera->m_iChiave = -9696 ;
				theApp.m_pDlgTastiera->m_dValoreInput = dValore ;
				theApp.m_pDlgTastiera->Create(IDD_DLG_TASTIERA);
				theApp.m_pDlgTastiera->ShowWindow(SW_SHOW);
				theApp.m_pDlgTastiera->ScriviDicitura(sMessaggio);
				theApp.m_pDlgTastiera->m_pNotifica = m_pDG;
			}
			bPropaga = false;
		}
		break;
	}

	if (bPropaga)
	{
```

```
		short i,j,iNumElementi,iNumPagine;

		iNumElementi = theApp.m_aAmbienti.GetSize();

		for (i=0; i<iNumElementi; i++)
		{
			iNumPagine = theApp.m_aAmbienti.ElementAt(i)->aPagine.GetSize();

			for (j=0; j<iNumPagine; j++)
			{
				if (theApp.m_aAmbienti.ElementAt(i)->aPagine.ElementAt(j)->iTipo == 1)
				{
					((CPagina *)theApp.m_aAmbienti.ElementAt(i)->aPagine.ElementAt(j)-
>pPagina)->Evento(iEvento, sMessaggio, dValore);
				}
			}
		}
	}

	return S_OK;
}}
```

○ OnSystemEvent

Segnalazioni dal sistema

```
HRESULT OnSystemEvent(   BSTR   sMsg,
                         int    lSubSystem ,
                         int    lNumberMsg,
                         int    iNumberParameter.
                         VARIANT * pVar
                      )
```

Parametri

BSTR	sMsg	""
int	lSubSystem	Sottosistema che ha prodotto il messaggio 0 Sistema centrale 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
int	lNumberMsg	Numero del messaggio nel LOGFILE
int	iNumberParameter	Numero di stringhe del messaggio
VARIANT * pVar		Stringhe del messaggio

Esempio

```
HRESULT CNotify::OnSystemEvent(  BSTR    sMsg,
                                 int     lParam1 ,
                                 int     lParam2,
                                 int     iNumberParameter.
                                 VARIANT * pVar
                               )
{
HRESULT                 hr;
SAFEARRAY FAR*          psa;
double                  dVal;
CString                 csMsg;
CString                 csStringDescription
long                    ip;

csMsg = CString(sMsg);
if ( iNumberParameter )
{
        psa  = pdVar->parray ;
        if ( psa )
        {
                hr        = SafeArrayGetLBound( psa, 1, &lLBound );
                hr        = SafeArrayGetUBound( psa, 1, &lUBound );
                lNumElement = lUBound - lLBound + 1;
                if ( lNumElement )
                {
                        if ( !FAILED ( hr ))
                        {
                                for ( ip=0 ; ip < (long)lNumElement; ip++ )
                                {
                                        SafeArrayGetElement( psa, &ip , &bstrElem );
                                        csStringDescription = CString( bstrElem );
                                        bstrElem.Empty();
                                        .......
                                        .......
                                }
                        }
                }
                SafeArrayDestroyData( psa  );
                SafeArrayDestroyDescriptor( psa  );
        }
}
```

○ OnCompleteCommand

Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta)

```
HRESULT OnCompleteCommand( BSTR      sMsg,
                           int       lParam1,
                           int       lParam2,
                           int       iNumberElement.
                           ELEMENTS * pVar
                         )
```

Parametri

BSTR	sMsg	Messaggio dipendente dalla funzione completata
int	lParam1	Parametro 1
int	lParam2	Parametro 2
int	iNumberParameter	Numero di parametri della funzione completata
VARIANT *	pVar	Parametri della funzione completata

Esempio

```
HRESULT CNotify::OnCompleteCommand(  BSTR      sMsg,
                                     int iParam1,
                                     int iParam2,
                                     int       iNumberParameter.
                                     VARIANT * pVar
                                  )
{
HRESULT                 hr;
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iError;
double *                pdVettValue;
CString                 csMsg;
long                    ip;

csMsg = CString(sMsg);
if ( iNumberParameter )
{
        psa  = pVar->parray ;
        if ( psa )
        {
                pdVettValue = new double[iNumberParameter];
                hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                lNumElement = lUBound - lLBound + 1;
                if ( lNumElement )
                {
                        if ( !FAILED ( hr ))
                        {
                                for ( ip=0 ; ip < (long)lNumElement; ip++ )
                                {
                                        SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                }
                        }
                }
                SafeArrayDestroyData( psa  );
                SafeArrayDestroyDescriptor( psa  );
                .......
                .......
                delete pdVettValue;
        }
}
```

○ OnCompleteElement

Segnala il completamento di una funzione asincrona (con ID dato all'atto della richiesta)

```
typedef struct  tagELEMENTS
{
   double dElements[ 32 ];
}      ELEMENTS;
```

```
HRESULT OnCompleteElement(   BSTR        sMsg,
                             int         lParam1,
                             int         lParam2,
                             int         iNumberElement.
                             ELEMENTS * pVar
                           )
```

Parametri

BSTR	sMsg	Messaggio dipendente dalla funzione completata
int	lParam1	Parametro 1
int	lParam2	Parametro 2
int	iNumberElementi	Numero di parametri della funzione completata
ELEMENTS * pVar		Parametri della funzione completata

Esempio

```
HRESULT CNotifica::OnCompleteElement(   BSTR sMsg,
                                        int iParam1,
                                        int iParam2,
                                        int iNumElem,
                                        ELEMENTS * pElements
                                        )
{
        bool bPropaga = true;

        CString sMessaggio = CString(sMsg);

        if (bPropaga)
        {
                short i,j,iNumElementi,iNumPagine;

                iNumElementi = theApp.m_aAmbienti.GetSize();

                for (i=0; i<iNumElementi; i++)
                {
                        iNumPagine = theApp.m_aAmbienti.ElementAt(i)->aPagine.GetSize();

                        for (j=0; j<iNumPagine; j++)
                        {
                                if (theApp.m_aAmbienti.ElementAt(i)->aPagine.ElementAt(j)->iTipo == 1)
                                {
                                }
                        }
                }
        }

        return S_OK;
}
```

○ OnRequestServiceEvent

Richiede un servizio dal applicativo utente in termini di dati, che quando reperiti verranno forniti tramite l'opportuna funzione di scrittura.
L'applicazione dell'utente predisposto ad eseguire la richiesta,
una volta che ha terminato il compito deve segnalare tramite la funzione "WriteServiceParameterAndContinue" che l'operazione è stata eseguita.

```
HRESULT OnRequestServiceEvent(  BSTR   sMsg,
                                BSTR   sMsgFunz,
                                int    lSubSystem ,
                                int    lKey,
                                int    iNumberParameter.
                                VARIANT * pVar
                              )
```

Parametri

BSTR	sMsg	Messaggio dipendente dal servizio richiesto
BSTR	sMsgFunz	Messaggio dipendente dalla funzione richiedente
int	lSubSystem	Sottosistema che ha prodotto il messaggio 0 Sistema centrale 1 AxesBrainL 2 AxesBrainISO 3 AxesBrainAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
int	lKey	Chiave per didtinguere le richieste 101 Richiesta di un servizio generico 102 Richiesta di un input da tastiera 103 Richiesta di una funzione VisAlgo 104 Richiesta di una procedura VisAlgo
int	iNumberParameter	Numero di parametri
VARIANT * pVar		Parametri della richiesta

Esempio

```
HRESULT CNotify::OnRequestServiceEvent(
                                        BSTR     sMsg,
                                        BSTR     sMsgF,
                                        int      lParam1 ,
                                        int      lKey,
                                        int      iNumberParameter.
                                        VARIANT * pVar
                                       )
{
VARIANT                 Var;
HRESULT                 hr;
SAFEARRAYBOUND          sabound[1];
SAFEARRAY FAR*          psa;
double                  dVal;
int                     iNumberParameterDouble,
int                     iError;
double *                pdVettValue;
int                     lParam1,
int                     lKey,
CString                 csMsg;
long                    ip;
char                    sString[128];

csMsg = CString(sMsg);
if ( iNumberParameter )
{
        psa  = pVar->parray ;
        if ( psa )
        {
                pdVettValue = new double[iNumberParameter];
                hr         = SafeArrayGetLBound( psa, 1, &lLBound );
                hr         = SafeArrayGetUBound( psa, 1, &lUBound );
                lNumElement = lUBound - lLBound + 1;
                if ( lNumElement )
                {
                        if ( !FAILED ( hr ))
                        {
                                for ( ip=0 ; ip < (long)lNumElement; ip++ )
                                {
                                        SafeArrayGetElement( psa, &ip , &pdVettValue[ip] );
                                }
                        }
                }
                SafeArrayDestroyData( psa  );
                SafeArrayDestroyDescriptor( psa  );
                .......
                .......
                delete pdVettValue;
        }
}
```

```
VariantInit(&Var);
Var.vt     = VT_ARRAY | VT_R8 ;
dVal       = 0;
sabound[0].cElements = 16;
sabound[0].lLbound   = 0;
psa = SafeArrayCreate( VT_R8 , 1, sabound );
if ( psa )
{
        for ( ip = 0; ip < 16 ; ip++ )
        {
                hr         = SafeArrayPutElement(psa , &ip , &dVal);
                if ( FAILED(hr))
                {
                        SafeArrayDestroy(psa);
                        psa = NULL;
                        break;
                }
        }
}
if ( psa )
{
        iNumberParameterDouble  = 16;
        Var.vt       = VT_ARRAY | VT_R8 ;
        Var.parray = psa;
        Hr           = m_GMSSrv->WriteServiceParameterAndContinue(
                                                        1,
                                                        CComBSTR( csMsg ),
                                                        lParam1,
                                                        lKey,
                                                        iNumberParameterDouble,
                                                        &Var,
                                                        &iError
                                                        );
        SafeArrayDestroyData( psa  );
        SafeArrayDestroyDescriptor( psa  );
        if ( hr != S_OK )
        {
                sprintf(sString, "Error WriteServiceParameterAndContinue =%ld", iError);
                AfxMessageBox( sString );
        }
        else
        {
                AfxMessageBox( "WriteServiceParameterAndContinue OK" );
        }
}
```

○ OnRequestServiceEventElement

Richiede un servizio dal applicativo utente in termini di dati, che quando reperiti verranno forniti tramite l'opportuna funzione di scrittura.
L'applicazione dell'utente predisposto ad eseguire la richiesta,
una volta che ha terminato il compito deve segnalare tramite la funzione "WriteServiceElementAndContinue" che l'operazione è stata eseguita.

```
typedef struct  tagELEMENTS
{
   double dElements[ 32 ];
}      ELEMENTS;
```

HRESULT OnRequestServiceEventElement (
BSTR sMsg,
BSTR sMsgFunz,
int lParam1,
int lParam2,
int iNumberElement.
ELEMENTS * pVar
)

Parametri

BSTR sMsg	Messaggio dipendente dalla funzione richiedente
BSTR sMsgFunz	Messaggio dipendente dalla funzione richiedente
int lParam1	Parametro 1
int lParam2	Parametro 2
int iNumberElementi	Numero di parametri della funzione completata
ELEMENTS * pVar	Parametri della funzione completata

Esempio

```
HRESULT CNotifica::OnRequestServiceEventElement(
                                                BSTR sMsg,
                                                BSTR sMsgFunz,
                                                int iParam1,
                                                int iParam2,
                                                int iNumElem,
                                                ELEMENTS * pElements
                                                        )
{
        bool bPropaga = true;

        CString sMessaggio = CString(sMsg);
        CString sMessaggioFunz = CString(sMsgFunz);

        if (bPropaga)
        {
                short i,j,iNumElementi,iNumPagine;

                iNumElementi = theApp.m_aAmbienti.GetSize();

                for (i=0; i<iNumElementi; i++)
                {
                        iNumPagine = theApp.m_aAmbienti.ElementAt(i)->aPagine.GetSize();

                        for (j=0; j<iNumPagine; j++)
                        {
                                if (theApp.m_aAmbienti.ElementAt(i)->aPagine.ElementAt(j)->iTipo == 1)
                                {
                                }
                        }
                }
        }
}
```

○ EventError

Segnalazione di un errore dal sistema

```
HRESULT EventError(     short   iErrorCode,
                        short   iNumVal,
                        double  dVal1,
                        double  dVal2,
                        double  dVal3,
                        double  dVal4,
                        double  dVal5,
                        short   iNumStr,
                        BSTR    sStr1,
                        BSTR    sStr2,
                        BSTR    sStr3,
                        BSTR    sStr4,
                        BSTR    sStr5
                   )
```

Parametri

short	iErrorCode	Codice dell'errore
short	iNumVal	Numero di valori significati
double	dVal1	
double	dVal2	
double	dVal3	
double	dVal4	
double	dVal5	
short	iNumStr	Numero di stringhe significative
BSTR	sStr1	
BSTR	sStr2	
BSTR	sStr3	
BSTR	sStr4	
BSTR	sStr5	

Esempio

```
HRESULT CNotifica::EventError(
                                    short iErrorCode,
                                    short iNumVal,
                                    double dVal1,
                                    double dVal2,
                                    double dVal3,
                                    double dVal4,
                                    double dVal5,
                                    short iNumStr,
                                    BSTR sStr1,
                                    BSTR sStr2,
                                    BSTR sStr3,
                                    BSTR sStr4,
                                    BSTR sStr5
                                   )
{
        return S_OK;
}
```

Appendice

A

9. Collegamento a Visual Basic

○ Funzionalità per l'utilizzo del Server con Visual Basic

a) Andare sul menu Progetto, selezionare la voce Riferimenti e selezionare dalla lista:
AxesBrainSRV ...

b) Andare nelle dichiarazioni generali del Form e inserire la seguente riga:

Dim WithEvents Server As AxesBrain.

c) Nel Form_Load() inserire la seguente riga:

```
Private Sub Form_Load()

    Set Server = New AxesBrain

    Server.SystemCommand 2 ' 2 Servo ON

End Sub
```

- ## Esempio di uso del server in Visual Basic:

```
Private Sub Command1_Click()

        Dim iHandle1 As Long
        Dim iHandle2 As Long

        Dim iErrore As Long

        Dim PosizA As String
        Dim PosizB As String

        PosizA = "X,100"
        PosizB = "Y,200"

        Server.OpenHandleAxes iHandle1, iErrore
        Server.OpenHandleAxes iHandle2, iErrore

        Server.CommandMotionAxes iHandle1, "MOV", PosizA, 12, iErrore, -1
        Server.CommandMotionAxes iHandle2, "MOV", PosizB, 13, iErrore, -1

End Sub

Private Sub Command2_Click()

        Dim iErrore As Long
        Dim dValori(2) As Double

        dValori(0) = 10
        dValori(1) = 20

        'Lancia un part program
        Server.ExecSubSystemProcess 1, "c:\ProvaVB\", "Test.pp", "", 0, 2, dValori, 14,
        iErrore, -1

        ' Scrive un output
        Server.WriteIO 2, "", 1, 2, 1, 0, iErrore

End Sub
```

```
Private Sub Timer1_Timer()

        Dim dValore As Double
        Dim iErrore As Long
        Dim lRet As Long

        `Legge quota asse X
        Server.ReadAxisRegister 1, "X", 0, 1, dValore, iErrore
        Label1.Caption = Format(dValore, "0.000")

        `Legge quota asse Y
        Server.ReadAxisRegister 1, "Y", 0, 1, dValore, iErrore
        Label2.Caption = Format(dValore, "0.000")

        `Legge input
        Server.ReadIO 2, "", 1, 1, 1, dValore, iErrore

        Label3.Caption = dValore

End Sub

Private Sub Form_Unload(Cancel As Integer)

        Server.SystemCommand 1 ' 1 Servo OFF

End Sub

Private Sub Server_OnCompleteCommand(ByVal sMsg As String, ByVal iParam1 As Long,
ByVal iParam2 As Long, ByVal iNumElem As Long, pParam3 As Variant)

        Dim iErrore As Long

        If iParam2 = 12 Then
           'Finito asse X
           Server.CloseHandleAxes iParam1, iErrore
        End If

        If iParam2 = 13 Then
           'Finito asse Y
           Server.CloseHandleAxes iParam1, iErrore
        End If

        If iParam2 = 14 Then
           'Finito part program
        End If

End Sub
```

B

10. Elenco Errori

Errore	1	Funzionalità ancora attiva
Errore	2	Nome asse/mandrino/IO non corretto
Errore	3	OVERFLOW Tabelal nomi assi
Errore	4	OVERFLOW Tabella assi
Errore	5	Tipo asse incongruente
Errore	6	MANCA ASSE FISICO PER SISTEMA VIRTUALE
Errore	7	ASSE HELP NON USABILE (CYL_TRASLANTE)
Errore	8	OVERFLOW Tabella JOB di movimentazione
Errore	9	JOB di movimentazione NON DEFINITO
Errore	10	OVERFLOW AREA movimentazione
Errore	11	PARAMETRI Start continuo ERRATI
Errore	16	PARAMETRI Movimentazione ERRATI
Errore	17	TROPPE TPE CONTEMPORANEE
Errore	18	ASSE GIA' USATO IN ALTRI JOBS
Errore	19	ASSE FUORI CAMPO DI LAVORO
Errore	20	RAGGIO FINALE <> RAGGIO INIZIALE
Errore	21	JOB NON DEFINIBILE
Errore	22	TROPPI ASSI (MAX 8)
Errore	23	NON CI SONO PIU' TIMER
Errore	24	ASSI IN MOVIMENTO
Errore	25	STATO INCORRETTO
Errore	26	ERRORE DI SISTEMA NELL'USO DEL JOB
Errore	27	ASSE FRENATO
Errore	28	CLOSE O OPEN POSITION LOOP INUTILE
Errore	29	MANCA AREA SLAVE
Errore	30	CAMPIONATURA NON ATTIVA
Errore	31	TROPPI PARAMETRI
Errore	32	MANCA ASSERVIMENTO ATTIVO
Errore	33	TROPPE PIASTRE I/O CONFIGURATE
Errore	34	PIASTRA I/O SBAGLIATA
Errore	35	BIT NON SU PIASTRA
Errore	36	OVERFLOW DEL BIT LOGICO
Errore	37	BIT I/O SBAGLIATO
Errore	38	TROPPE PIASTRE ANALOGICHE CONFIGURATE
Errore	39	PIASTRA ANALOGICA SBAGLIATA
Errore	40	CANALE ANALOGICO NON SU PIASTRA
Errore	41	OVERFLOW DEL CANALE ANALOGICO (LOGICO)
Errore	42	CANALE ANALOGICO SBAGLIATO
Errore	43	SCELTA SCONOSCIUTA
Errore	44	TEMPO SCADUTO (Time OUT)
Errore	45	INTERPOLAZIONE SPLINE ERRATA
Errore	47	AZZERAMENTO ASSE NON FATTO (MANCA HOMING)
Errore	48	OVERFLOW TABLE SPINDLE
Errore	49	MANDRINO IN ATTIVITA' SAMPLE
Errore	50	MANDRINO IN ROTAZIONE
Errore	51	ASSERVIMENTO DISABILITATO
Errore	52	QUOTA CORREZIONE ERRATA (UGUALE)
Errore	53	OVERFLOW AREA CORREZIONE
Errore	54	BUFFER CIRCOLARE NON ATTIVO
Errore	55	OVERFLOW COUNTER NELL'ASSE STEPPER
Errore	56	COUNTER NON DEFINITO NELL'ASSE STEPPER
Errore	58	MOVIMENTO IN DEADLOCK
Errore	1001	HANDLE INCORRETTO
Errore	1002	DRIVER OCCUPATO
Errore	1003	INIZIALIZZAZIONE ERRATO
Errore	1004	ANOMALIA IN FASE DI AZZERAMENTO
Errore	1006	TROPPI HANDLER DI MOVIMENTAZIONE ATTIVI

Errore Driver Fault	0x5	Configuration error - Decision Table
	0x7	Calibration error
	0x10E	Fiber not connected to digital driver
	0x111	Digital driver assh't hardware enable
	0x113	Fiber rx Frame error
	0x114	Fiber rx Parity error
	0x116	Zero command refused for movement
	0x117	Lost zero axis