Carl Zeiss SMT Ltd

WinTEM Remote API

programming manual

Programming the API

WinTEM Remote API

Programming the API

Abstract

This document is a manual for 3rd party software developers to interface with the WinTEM software. More specifically it describes how to interface to the Carl Zeiss SMT CZEMAPI.OCX in the programming language Visual C++.

Part Number: 354034-6001-006

Date: 5 August 2009

Issue: 2.9

Printed in England

Table of Contents

1. Introduction

This guide is written for 3rd party developers wishing to write software applications that communicate with the new range of Carl Zeiss Transmission Electron Microscopes (TEM). The application programming interface (API) consists of an Active X control that provides access to the WinTEM software and via this software to the TEM microscope settings. The following models of TEMs are supported:

- LIBRA 120
- LIBRA 200 FE / MC
- SESAM
- UHRTEM

As an Active X control the API can be accessed by various programming languages such as Visual Basic, Visual C++ and various scripting languages (VBScript, JavaScript). A small survey has shown us that most 3rd party developers prefer to use MS Visual C++ so the example code that is on the installation CD-ROM is written in C++ (Visual Studio 2005).

In this manual the reader will be made familiar with some concepts of the EM Server – client model, and the role of the API in it. The API exposes a COM Idispatch interface and all methods and Events will be explained. Some snippets of example code will be presented but the user is best referred to the example program. At the end we present a list of Parameters (analogues and digital states) and commands that should operate the machine. Some parameters are read-only, depending on modes and privileges. Same for commands: not all of them are allowed because of the machine type, options and user privileges.

Whilst this document describes the use of the Carl Zeiss SMT API.OCX, it is necessary to have an agreement with Carl Zeiss SMT to develop applications that utilise it. Upon the signing of such an agreement, Carl Zeiss SMT will provide the necessary support files and technical information to allow development to proceed. In the first instance contact the Carl Zeiss SMT software team at:

Carl Zeiss SMT Ltd

511 Coldhams Lane

Cambridge CB1 3JS

United Kingdom

Tel: +44-1223-414166

Email: software@smt.zeiss.com

2. Some concepts

2.1 Client – Server architecture

The main part of the WinTEM software is the EM Server. The Server maintains a list of states and parameters of the TEM. The WinTEM UIF or external applications can send instructions (“Filament Heating On”) or modify parameters (“Magnification = 10000”), which are then processed by the EM Server. The CZEMAPI ActiveX Control is the main channel of communication to the EM Server. It can run on the same computer as the EM Server or on a second computer.

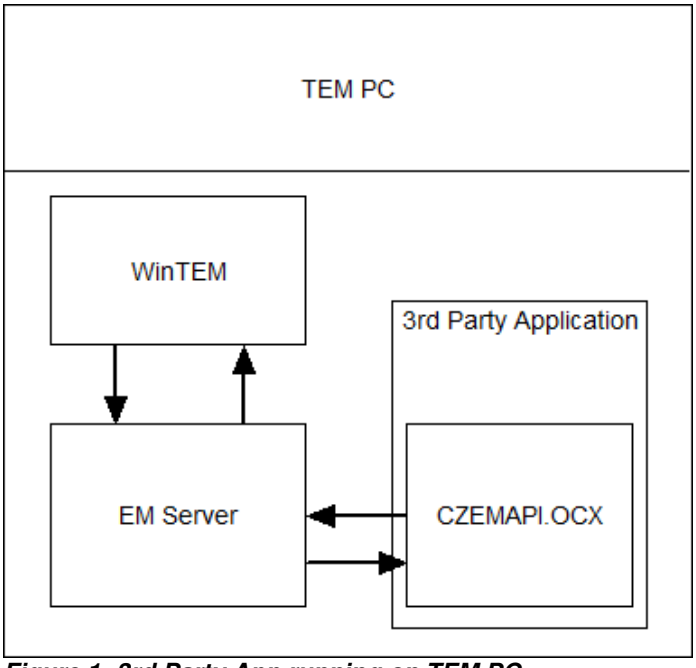

Figure 1: 3rd Party App running on TEM PC

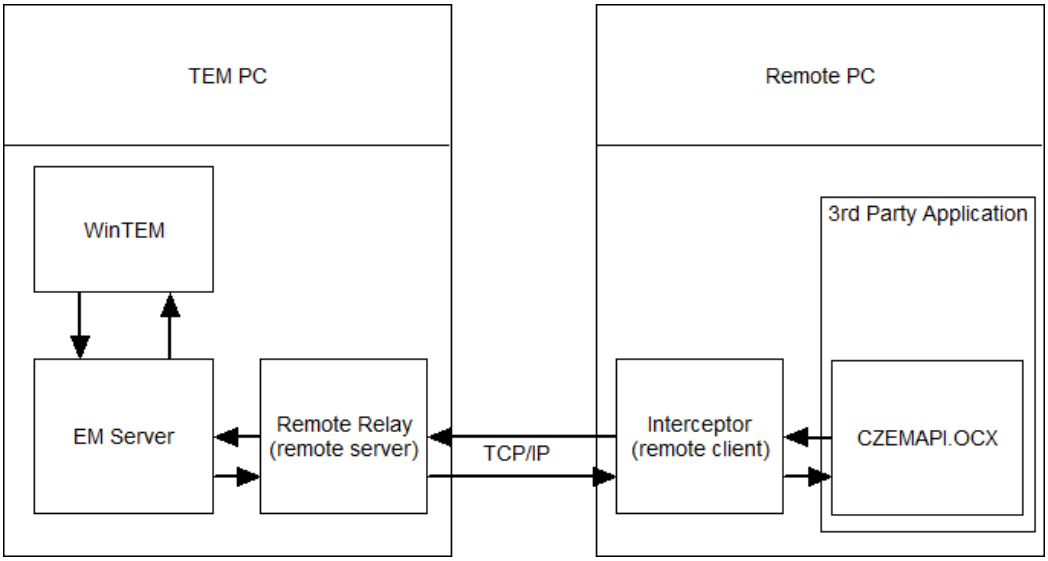

Figure 2: 3rd Party App running on remote PC. The communication goes via a LAN network.

When the 3rd party software runs on a separate computer then the API Active X will take care of the communication between client and EM Server, but that has to be configured first. Most notably the "Remote Client" has to be installed on the remote PC. The Remote Client installer can be found on the Remote API developer disk. On the TEM PC there will always be "Remote Relay" running. This is the Remote Server.

2.2 Parameters, States and Commands

The control of an instrument, and hence the interface between User Interface components and the Instrument control, is based on the concept of parameters, states and commands. All messages to the server from clients applications that refer to any

parameter do so by means of a unique identifier that is derived from the type of parameter and the low-level parameter ID. When using the API Control you just refer to the parameter symbol as a string (e.g. “AP_TEM_MAG”) and the Control will get the parameter ID.

All parameters, states and commands have a visibility attribute that determines if they are used on the current machine type.

Every parameter, state and command may have associated with it a Rule that defines when it is permitted or valid. If a parameter (or state or command) is not permitted or invalid then it should be presented as greyed-out on the UIF. Rules are defined as logical expressions based on states, state values and licences. The EM Server manages and applies the rules.

Setting of a parameter and display of a parameter are two very separate functions:

- If the user sets a value on the UIF this results in a request to the EM Server to set the value. As appropriate the value is subject to limit and validity checking and then routed to the appropriate low-level module to perform the required function, this may or may not result in the parameter being set to the required value. For instance, a value may not be achievable under the current set of conditions or it may be subjected to some time constant.
- The low-level functions detect a parameter value change, either because of setting a value, executing a command or interrogation of hardware. Hence, a notification message is send to the server, which is passed on to API.OCX, which then notifies interested UIF objects that a parameter or state has changed. The UIF element should in turn request the current values (suitably converted) for display.

This separation ensures that all objects are correctly updated and setting of parameters is subjected to consistent checks that are localised to the modules concerned.

Parameters are subject to upper and lower limits and machine states can have only a number of discrete values.

Since there are a very large number of parameters on the TEM, a mechanism is provided to reduce the number of choices presented to the user according to the user’s requirements. Each parameter, state and command is assigned to a category: Novice, Expert or Service and a user may select a ‘User Level’ of Novice/Expert/Service which restricts the number of parameters presented for selection according to category, i.e. Novice presents only Novice category, Expert presents Novice and Expert category, Service presents all parameters.

2.2.1 Analogue Parameters

Analogue parameters are those that have a full numerical range between defined limits. Example: “Image Shift X”.

All parameter values are in standard SI units, e.g. currents are in Amps, distances are in Metres, etc. Many adjustment parameters, like deflector values, are in percentages and are internally converted into currents. When parameters are displayed on the UIF then a suitable display string is made, using for instance mm, µm, nm, etc. rather than 10^{-3} m, 10^{-6} m, 10^{-9} m etc.

Every analogue parameter has a unique numerical identity, which is associated with a symbol starting AP_, e.g. AP_TEM_MAG. This identity is used throughout the Server and lower level modules.

Attributes of an analogue parameter are:

- Name (for display – by default English, but may be alternative language)
- High limit
- Low Limit

- Entry only (if set parameter is not available for mouse adjustment)
- Read Only (Cannot be set by the user; modified internally by EM Server)
- Parameter type (Normal, X of XY pair, Y of XY pair)
- Dynamic characteristics for mouse adjustment (linear/log with constants defining logical to physical relationship)
- Units type. Defines the display format according to current value and data entry format
- Coarse/Fine increments for mouse adjustment (logical units)
- Enabled/Disabled (according to Rules)
- Visible (if not set then does not apply to the current machine type)
- Target/Actual. Identifies if parameter is Target or Actual and identifies the associated parameter – see below

There is a special group of analogue parameters, which consists of pairs related as Target and Actual values.

An Actual value is a Read-Only parameter which displays the current value of some entity, e.g. AP_STAGE_AT_X defines the current Goniometer X position.

A Target value is a Read/Write parameter, which defines the desired value for some entity, e.g. AP_STAGE_GOTO_X defines the requested Goniometer X position.

2.2.2 Digital Parameters (states)

Digital parameters are those parameters that have a discrete set of values. Example "HT Wobble" has the states "On" and "Off"**.**

Every digital parameter has a unique numerical identity which is associated with a symbol starting DP_ , e.g. DP_WOBBLE. This identity is used throughout the Server and lower level modules. Read-Only (RO) states represent the dynamic conditions of the instrument, e.g. the "HT State" which includes not only the static states like "Off" or "On" but also all transitional states such as "ramping" and "shutting down".

Attributes of Digital Parameters:

- Name (for display – by default English, but may be alternative language)
- Read-Only. (cannot be set by the user; modified internally by EM Server)
- Enabled/Disabled (according to Rules)
- Set of state values – see below
- Group state. Identifies this parameter as a Group state (see below) if set the state also has a collection of group command identities.
- Visible (if not set then does not apply to the current machine type)

State Values

State values are the discrete values that a state may have.

Every state value has a unique symbol starting S_ e.g. S_ON, S_IDLE etc.

Attributes of a state value are:

- Value (e.g. 0)
- Name (for display – by default English, but may be alternative language)

2.2.3 Commands

Commands are functions that initiate some action. E.g. "Filament On".

Every command has a unique numerical identity which is associated with a symbol starting CMD_ , e.g. CMD_FILAMENT_ON. This identity is used throughout the Server and lower level modules.

Attributes of a command are:

- Name (for display – by default English, but may be alternative language)
- Enabled/Disabled (according to Rules)
- Group Flag indicating if command is associated with a Group State (see below), if set the command also includes the identity of the Group State.
- Visible (if not set then does not apply to the current machine type)

2.2.4 Groups

These are a special combination of a Read-Only state and the group commands which affect the state. Example is “HT State” and the associated Commands “HT On”, “HT Off” etc.

2.3 Licences

Licences are used to enable optional functions. Each licence has a unique numerical identity, which is associated with a symbol starting LIC_, e.g. LIC_VAR_VOLTS for the Variable Voltage Software licence. Licences are used in rules and a licence state may be queried via the EM Server.

2.4 Macros

Macros are a set of SEM instructions that can include conditionals (if..then…else), states, analogue parameters, commands and special functions. With macros it is possible to bring up SmartSEM dialogs or run programs on the SEM PC. Refer for further instructions to the Macro editor online Help.

3. Installing the CZEMAPI ActiveX control

Firstly, you need to decide how your client application is going to be used. Either it is installed on the same PC as where the EM Server runs, i.e. on the microscope PC, or the client application is installed on a second, remote PC. If you chose the first option then you only need to have the WinTEM software installed on your PC. For the second option, you also need to install the Remote Client on the second PC.

3.1 Installing the WinTEM software

You need the WinTEM installation CD. First install any drivers and then install the WinTEM software. For testing purposes you can install the WinTEM software as „PC-only“ on a PC without microscope attached to it. In order to run the WinTEM software as PC-only you need to obtain a PC dongle from Zeiss. Installing WinTEM will also install and register the CZEMAPI.OCX. It is now ready to be put into your application.

The WinTEM installation will also install the remote server program on the PC. This will include the .NET Framework 1.1 (or later). The remote server is called “Remote Relay.exe” and it will take care of any communication between EM Server and Client(s). Remote Relay.exe starts automatically on Windows start-up and will continue to run in the background. It can handle multiple remote clients, on multiple PCs. On the first installation, RemoteRelay tries to access the network. Windows XP security will detect this activity and block it. You must now manually allow RemoteRelay to access the network and instruct Windows XP to remember this decision.

3.2 Installing the remote client

If the client software runs on a second PC then you need to install the Remote Client software, to be found on the Remoting Installer CD-ROM. Run the RClientInstaller.exe. This will install CZEMApi.ocx, support files and the Client Configuration program. Run RConfigure.exe. This will configure the IP addresses and ports of Server and Client on the Client side. A small program will start running in the background of the client machine, called “Interceptor.exe”. This program actually communicates to “Remote Relay.exe” on the server. CZEMApi.ocx is automatically registered on the Client computer and ready for use. The EM Server can handle more than one remote client at the time. When Interceptor tries to access the network, Windows XP security will detect this activity and block it. You must now manually allow Interceptor to access the network and instruct Windows XP to remember this decision.

3.3 Dot NET Remoting vs. NetDDE

The remote client – server communication is done by the programs “Interceptor.exe” and “Remote Relay.exe”. These use the new Microsoft .NET remoting technology. This is why the .NET runtime is installed on both client and server. For the 3rd party developer there is no need to be concerned how it works under the hood however.

Previous versions of the CZEMApi.ocx, called LeoApi.ocx, also supported the older technology NetDDE. NetDDE is still supported in CZEMApi.ocx but this manual will not explain how to set it up because it will be deprecated in the near future.

3.4 Inserting the CZEMApi ActiveX control into an MFC Application

From your MFC App in Visual Studio 6 chose the menu “Project>Add to Project>Components and Controls”. Chose “CZ EM API Control” from the “Registered Active X Controls” from the Gallery. This will insert the class CZEMApi into the project.

In Visual Studio .NET 2003 you do something similar. Chose the menu “Project > Add Class…”. From the dialog box chose the category “Visual C++ > MFC > MFC Class from Active X Control” and press <Open>. This brings up the “Add class from Active X Control Wizard”. Chose CZ EM API Control from the list of available Active X controls. Rename class and stub files to CZEMApi etc.

If the CZ EM Api Control is not in the list then you need to register the Active X control with RegSvr32. If there is already a file association then double-click on the control program icon, or right-click and select “Open with…” and browse for RegSvr32 in the Windows\System32 directory.

The control can now be dragged on a form and you can associate a control parameter with it from the ClassWizard. In Visual Studio .NET you have to do this manually. Declare a member parameter in the header file:

```
CZEMApi m_cApi;
```

Subsequently create the control object in the form class OnInitDialog(..) member function. See the example code.

4. The API Interface

For initialisation of the control you have to chose between initialising the CZEMApi for use with EM Server running locally, or as a Client App on a second PC. For the former use “Initialise” and for the latter use “InitialiseRemote”. The CZEMApi.ocx has the following methods and events:

4.1 Initialise Method

long Initialise(LPCTSTR lpszMachine)

[in] *lpszMachine* is the name of the machine with the microscope.

Remarks:

Use Initialise(…) to run the client App locally on the same PC as EM Server.

This method must be called before any of the others (except GetVersion). It initialises the communication between the OCX and the EM Server. If a NULL machine name is supplied, the EM Server is assumed to be on the same machine as the CZEMAPI ActiveX Control.

Supplying a machine name will search for microscopes specified by the NetBIOS name, lpszMachine.

Return Value:

If the call succeeds, it returns 0. If the call fails an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED: The control could not be initialised

Error codes are defined in the header file “api_err.h”.

4.2 InitialiseRemoting Method

long InitialiseRemote(void)

Remarks:

Use InitialiseRemoting to run the client App on a second PC and communicate with the EM Server remotely. This method must be called before any of the others (except GetVersion). In contrast to the Initialise() method, no registration is made directly with EM Server. Instead, the Remoting architecture acts as a 'go-between' and will allow initialisation to take place no matter what the state is of the EM Server (loaded, unloaded, logged in, logged out).

You need to have the RemoteRelay.exe running on the same computer as the EM Server for this to work. This program runs from Windows start-up and does not provide a heavy load on computer resources.

If you start the API with InitialiseRemoting() you get more useful information on the status of the EM Server than with Initialise(), because it can distinguish between whether the EM Server has loaded or whether a user has logged On. Initialise() actually bypasses RemoteRelay and will run up the server and show the logon screen if needed. See also the “Notify Event” section and compare the notification messages that you get when using InitialiseRemoting() or Initialise().

InitialiseRemoting() can be called from an application running on the same PC as the EM Server. All that is needed is the RemoteClient to be installed and the client and remote IP addresses to be set to “localhost”.

Return Value:

If the call succeeds, it returns 0. If the call fails an error code is returned from the function.

Errors:

API_E_REMOTING_NOT_CONFIGURED Remoting incorrectly configured, use RConfigure to correct

API_E_REMOTING_FAILED_TO_CONNECT Remoting did not connect to the server

API_E_REMOTING_COULD_NOT_CREATE_INTERFACE Remoting could not start (unknown reason)

Note: the above errors can also occur when you use other methods. Please refer to api_err.h for all error definitions.

4.3 GetVersion Method

long GetVersion(short* Version)

[out] *Version* is the current version of the CZEMAPI OCX

Remarks:

This function has now become redundant.

Return Value:

If the call succeeds, it returns 0.

4.4 GetLimits Method

long GetLimits(LPCTSTR lpszParam, VARIANT* vMinValue, VARIANT* vMaxValue)

[in] *lpszParam* is the name of the parameter e.g. AP_IMAGE_SHIFT_X

[out] *vMinValue* is the minimum value of the parameter

[out] *vMaxValue* is the maximum value of the parameter

Remarks:

This call will get the minimum and maximum value of the parameter specified and will return them in vMinValue and vMaxValue. The values will be returned as type VT_R4. You cannot get limits for digital parameters DP_… .

Return Value:

If the call succeeds, it returns 0. If the call fails, vValue's variant type is set to VT_ERROR and the error code is returned in this variable. The same error code is returned from the function.

Errors:

API_E_NOT_INITIALISED: The control has not been initialised

API_E_GET_TRANSLATE_FAIL:Unrecognised parameter string or unable to translate string

API_E_GET_LIMITS_FAIL: Unable to get limits

4.5 Get Method

long Get(LPCTSTR lpszParam, VARIANT* vValue)

[in] *lpszParam* is the name of the parameter e.g. AP_IMAGE_SHIFT_X

[in][out] *vValue* is the value of the parameter

Remarks:

This call will get the value of the parameter specified and return it in vValue.

In C++, before calling this functions you have to specify the variant type (vValue.vt) to either VT_R4 or VT_BSTR. If no variant type is defined for vValue, it defaults to VT_R4 for analogue parameters (AP_XXXX) and VT_BSTR for digital parameters (DP_XXXX).

If the variant type is VT_R4 for an analogue parameter, then the floating point representation is returned in the variant. If a VT_BSTR variant is passed, analogue values are returned as scaled strings with the units appended (e.g. AP_SPOT_SIZE would return "Spot Size = 10nm").

For digital parameters, VT_R4 variants result in a state number and VT_BSTR variants result in a state string (e.g. DP_IMAGE_MODE would return state 0 or “Image”).

In C++, if the variant type was specified as VT_BSTR then the API will internally allocate a BSTR which the caller has to de-allocate using the SDK call ::SysFreeString (vValue.bstrVal)

Parameter Type	Passed Variant Type	Return Variant Type	Variant Contents
Analogue	None	VT_R4	Floating point value e.g. 0.01
Analogue	VT_BSTR	VT_BSTR	Scaled string e.g. = 10 mm
Digital	None	VT_BSTR	State name e.g. “Off”
Digital	VT_R4	VT_R4	State number e.g. 0

Return Value:

If the call succeeds, it returns 0. If the call fails, vValue’s variant type is set to VT_ERROR and the error code is returned in this variable. The same error code is returned from the function.

Errors:

API_E_NOT_INITIALISED: The control has not been initialised

API_E_GET_TRANSLATE_FAIL:Unrecognised parameter string or unable to translate string

API_E_GET_AP_FAIL: Unable to get the analogue parameter specified

API_E_GET_DP_FAIL: Unable to get the digital parameter specified

API_E_GET_BAD_PARAMETER:Parameter specified is neither analogue (AP_) nor digital (DP_)

4.6 Set Method

long Set(LPCTSTR lpszParam, VARIANT* vValue)

[in] *lpszParam* is the name of the parameter e.g. AP_IMAGE_SHIFT_X

[in] *vValue* is the value of the parameter

Remarks:

This call will set the value of the parameter specified to the value specified in vValue. If the parameter is an analogue value (AP_XXX) then the variant type for vValue can be VT_R4 specifying the floating point value. If vValue has its variant type set to VT_BSTR and lpszParameter is an analogue parameter, the parameter is set to the scaled value specified in vValue (e.g. lpszParam=AP_WD and vValue->bstrVal=“10” would set the working distance to 0.010 i.e. 10 mm becomes 0.010 metres)[2] . There is no easy way to discover the units for the scaled value other than by getting a value using vValue set to VT_BSTR in a Get() and extracting the units from the string value returned.

State parameters are set if the variant type for vValue is VT_BSTR or VT_R4 representing a state string and a state value respectively (e.g. DP_WOBBLE can be set to 0 or “Off”).

Parameter Type	Passed Variant Type	Variant Contents
Analogue	VT_R4	Floating point value e.g. 0.01
Analogue	VT_BSTR	Scaled string e.g. 10
Digital	VT_BSTR	State name e.g. Off
Digital	VT_R4	State number e.g. 0

Return Value:

If the call succeeds, it returns 0. If the call fails, vValue's variant type is set to VT_ERROR and the error code is returned in this variable. The same error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised
API_E_SET_TRANSLATE_FAIL:	Unrecognised parameter string or unable to translate string
API_E_SET_STATE_FAIL:	Unable to set digital parameter state
API_E_SET_FLOAT_FAIL:	Unable to set analogue parameter value
API_E_SET_BAD_VALUE:	Unrecognised value submitted (neither a string nor a float)

4.7 GetMulti Method

Not implemented.

4.8 SetMulti Method

Not implemented.

4.9 Execute Method

long Execute(LPCTSTR lpszCommand)

[in] *lpszCommand* is the name of the command e.g. CMD_FILAMENT_ON

Remarks:

This call will execute the command specified. If lpszCommand has a prefix of MCF_ or MCL_ then a macro file or macro library command will be executed (e.g. MCF_FREEZE will run the macro file FREEZE). The macro needs to be located in the … WinTEM\DISTRIB directory or the user directory (e.g. C:\Program Files\Carl Zeiss SMT Ltd\WinTEM\user\default\My Macro.MLF) of the currently logged-on microscope user.

Return Value:

If the call succeeds, it returns 0. If the call fails an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised
API_E_EXEC_TRANSLATE_FAIL:	Unrecognised CMD_ string or unable to translate string
API_E_EXEC_CMD_FAIL:	Failed to execute CMD_ command
API_E_EXEC_MCF_FAIL:	Failed to execute MCF_ macro file
API_E_EXEC_MCL_FAIL:	Failed to execute MCL_ macro library
API_E_EXEC_BAD_COMMAND:	Unrecognised command (not CMD_, MCF_ nor MCL_)

4.10 GetStagePosition Method

long GetStagePosition(VARIANT* x, VARIANT * y, VARIANT * z,

VARIANT * t, VARIANT * r, VARIANT * m)

[out] *x* is the x co-ordinate of the stage/metres

[out] *y* is the y co-ordinate of the stage/metres

[out] *z* is the z co-ordinate of the stage/metres

[out] *t* is the alpha-tilt of the stage/degrees

[out] *r* is the rotation of the stage/degrees

[out] *m* is the beta-tilt of the stage/degrees

Remarks:

This call obtains the current position of the stage, if initialised. The stage variables will be returned as pointers to VT_R4 (Single or float) values.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised
API_E_GET_STAGE_FAIL:	Unable to get stage values

4.11 MoveStage Method

If the call succeeds, it returns 0.

long MoveStage(float x, float y, float z, float t, float r, float m)

[in] *x* is the x co-ordinate of the stage/metres

[in] *y* is the y co-ordinate of the stage/metres

[in] *z* is the z co-ordinate of the stage/metres

[in] *t* is the tilt of the stage/degrees

[in] *r* is the rotation of the stage/degrees

[in] *m* is the beta-tilt of the stage/degrees

Remarks:

This call moves the stage to the position specified. The call is non-blocking and hence DP_STAGE_IS should be examined to check whether the stage is busy or idle.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised
API_E_GET_STAGE_FAIL:	Unable to set stage values

4.12 Grab Method

long Grab(short xoff, short yoff, short width, short height, short reduction, LPCTSTR lpszFilename)

[in] *xoff* is the x-offset of the image to grab

[in] *yoff* is the y-offset of the image to grab

[in] *width* is the width of the image to grab

[in] *height* is the height of the image to grab

[in] *reduction* is the subsampling factor for the image to grab (0 means no subsampling)

[in] *lpszFilename* is the filename where the Bitmap image is saved.

Remarks:

This call grabs an image from the microscope (if STEM option) with the offset, dimensions and subsampling supplied and saves it as a Windows Bitmap in the file specified by lpszFilename. If reduction is set to -1, then the image is grabbed with the overlay plane or datazone with no subsampling.

If this command is used over a network (after InitialiseRemote) the image file is created on the local, client machine.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function**.**

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised
API_E_GRAB_FAIL:	Grab failed

4.13 GetLastError Method

long GetLastError(VARIANT* Error)

[out] *Error* is the error string associated with the error code for the last error

Remarks:

This call will return a VT_BSTR VARIANT associated with the last error.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised

4.14 ClosingControl Method

long ClosingControl()

Remarks:

This call must be called before the control is destroyed.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised

4.15 SetNotify Method

long SetNotify(LPCTSTR lpszParameter, boolean bNotify)

[in] *lpszParameter* is the parameter to be notified regarding

[in] *bNotify* is a flag enabling or disabling the notification

Remarks:

This call will enable notification of changes or exceptions in the parameter specified if bNotify is set to TRUE. Setting bNotify to FALSE will disable notifications for that parameter. Notifications are received via the Notify() event.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

Errors:

API_E_NOT_INITIALISED:	The control has not been initialised
API_E_NOTIFY_TRANSLATE_FAIL:	Unrecognised parameter string or unable to translate string
API_E_NOTIFY_SET_FAIL:	Unable to set notification

4.16 GetCurrentUserName

long GetCurrentUserName(BSTR *strServerUserName, BSTR* strNTUserName)

[out]	strServerUserName	Username on Zeiss EM Server
[out]	strNTUserName	Windows XP username

Remarks:

Remoting only. Get both the usernames with which the user is logged onto Windows XP and logged onto the EM Server via the WinTEM application.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

4.17 GetUserIsIdle

long GetUserIsIdle(long* bUserIsIdle)

[out]	bUserIsIdle	User on Server is idle Yes/No

Remarks:

Remoting only. Has the user on the EM Server been idle for more than 30 seconds? This means, has he not been typing or moving the mouse? If “yes”, bUserIsIdle = 1, if “no”, bUserIsIdle = 0.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

4.18 GetLastRemotingConnectionError

long GetRemotingLastConnectionError(BSTR *strError)

[out]	strError	error string

Remarks:

Remoting only. Get the last remoting error as a string, if an error occurred.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

4.19 SetSuppressRemotingConnectionErrors

long SetSuppressRemotingConnectionErrors()

Remarks:

Remoting only. Suppress remoting connection errors to pop up so that the 3rd party application can handle error by itself. You can call this only once and not undo it.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

4.20 StartEMServer

long StartEMServer()

Remarks:

Remoting only. Start the EM Server if it has not started yet. Typically you would call this function if any of the other functions returns error 2030 API_E_REMOTING_EMSERVER_NOT_RUNNING.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function. If you try to start up the EM Server for a second time then you get the error code 2031 API_E_REMOTING_NO_USER_LOGGED_IN.

4.21 LogonToEMServer

long LogonToEMServer(LPCTSTR username, LPCTSTR userpassword)

Remarks:

Remoting only. Logon to EM Server with WinTEM user name and password. It is not needed to run the WinTEM UIF if you logon via this method. If WinTEM already runs then you do not need to call this method.

There are two cases when the EM Server will generate a dialog box on the screen of the WinTEM PC:

1) The username / password are incorrect so the Logon dialog appears to type in the username and passwords manually
2) The user logs on for the first time and a question about the licence agreement appears.

In both these cases the LogonToEMServer method will not return until the user has clicked OK or Cancel on the dialog. Cancelling the EM Server logon will cause the server to shut down.

Return Value:

If the call succeeds, it returns 0. If the call fails, an error code is returned from the function.

2030 API_E_REMOTING_EMSERVER_NOT_RUNNING The EM Server is not running and need to be started first.

4.22 Notify Event

Notify(LPCTSTR lpszParameter, long Reason) [eventid = 1]

[out] *lpszParameter* is the parameter for which this notification has been issued

[out] *Reason* is the numerical value of the cause of the notification

Remarks:

This event is fired in the following cases: Remote API Version 1.0 or when used on the local PC, initialised with Initialise(). In other cases handle the NotifyWithCurrentValue() event.

The Notify event will be fired by the control when a parameter for which it has been asked to notify a change in with the function SetNotify(), actually changes. The parameter is specified in lpszParameter (e.g. “AP_IMAGE_SHIFT_X”) and the reason for the notification is specified in Reason. This event is also fired when the EM Server is closing down in which case lpszParameter is set to “SVR_CLOSING_DOWN”.

The Reason parameter can have the following values:

0	Value changed
1	High limit changed
2	Low limit changed
3	Enabled
4	Disabled
5	Command Group Changed
6	XY Value Changed
1111111	EMSERVER_HAS_LOADED
2222222	EMSERVER_HAS_EXITED
3333333	EMSERVER_USER_LOGGED_ON
4444444	EMSERVER_USER_LOGGED_OFF
5555555	REMOTING_SERVER_EXITED
6666666	EMSERVER_FIBUI_HAS_LOADED

The last six notifications will only come after initialisation with InitialiseRemoting(). If your application gets the EMSERVER_HAS_LOADED notification then all it has to wait for is a user to log in with the EM Server (e.g. via the WinTEM application). If your application is notified EMSERVER_USER_LOGGED_ON then communication with the EM Server is possible. Before that, any attempt to communicate will get the reply "EMServer is not running on the remote machine" or "No user is logged into EM Server on the remote machine". It is not possible to provoke the logon screen via InitialiseRemoting() (as is possible with the Initialise() call). The notification EMSERVER_USER_LOGGED_OFF indicates the user has logged Off. The notification EMSERVER_HAS_EXITED means that the EM Server application has been stopped. The REMOTING_SERVER_EXITED and EMSERVER_FIBUI_HAS_LOADED are notifications that you should not normally get.

In practice, the notifications with “Value Changed” are the most important ones. The software should respond by a Get() method to get the actual value of the parameter.

4.23 NotifyWithCurrentValue Event

NotifyWithCurrentValue (LPCTSTR lpszParameter, long Reason, long paramid, double dLastKnownValue) [eventid = 2]

[out] *lpszParameter* is the parameter for which this notification has been issued

[out] *Reason* is the numerical value of the cause of the notification

[out] paramid the ID of the parameter as it is used in EM Server

[out] dLastKnownValue the current value as stored in EM Server

Remarks:

This event replaces the Notify event above in the following case: Remoting version 2.0 and above (WinTEM V1.3 and above) AND you use InitialiseRemoting(). The non-remoting function Initialise() will still give the Notify(..) event. Of course SetNotify(..) should be called first. It would be best to handle both events in your code.

The advantage of NotifyWithCurrentValue() is that the event delivers the current value of the changed parameter. This makes the subsequent Get() call to get the value unnecessary.

The Reason parameter has the same values as above.

5. Developing applications

5.1 Visual Basic

Once the control has been inserted, you can access its methods:

```
Private Sub Command1_Click()
Dim Reply As Long
Dim fValue As Variant

' Initialise a connection to the microscope named SEM
Reply = Api1.Initialise("")
If Reply = 0 Then
    MsgBox ("Initialised OK")
Else
    Reply = Api1.GetLastError(sError)
    MsgBox ("Unable to Initialise: " + CStr(sError))
End If

' Obtain the magnification value from the microscope
Reply = Api1.Get("AP_MAG", fValue)
If Reply = 0 Then
    MsgBox ("Magnification =" + CStr(fValue))
Else
    Reply = Api1.GetLastError(sError)
    MsgBox ("Unable to obtain magnification: " + CStr(sError))
End If

' Close the control
Reply = Api1.ClosingControl()
If Reply = 0 Then
    MsgBox ("Closed Control OK")
Else
    Reply = Api1.GetLastError(sError)
    MsgBox ("Unable to close control: " + CStr(sError))
End If

End Sub
```

5.2 Visual C++

The control can be inserted in the usual way. This will give you a new class named CZEMApi:

```
#include “api_err.h” // Error codes

API = new CZEMApi;
// ‘this’ specifies a CWnd derived class
// IDD_API is the identifier of the CZEMAPI OCX resource
BOOL bResult = API->Create("API", NULL, CRect(0,0,0,0), this, IDD_API,
                           NULL, FALSE, NULL);
if(bResult == FALSE)
{
       AfxMessageBox("Failed to create CZEMAPI OCX control");
       return;
}

// Initialise the control to connect to the local microscope
long lResult = API->Initialise(“”);
if(lResult != 0)
{
```

```
        AfxMessageBox("Failed to initialise control”);
        return;
    }

    // Get the magnification value from the microscope
    VARIANT vValue;
    lResult = API->Get(“AP_MAG”, &vValue);
    if(lResult != 0)
    {
        AfxMessageBox("Failed to get magnification”);
        return;
    }

    if(vValue.vt == VT_R4)
        *pOutputStream << vValue.fltVal;

    // Tidy up
    lResult = API->ClosingControl();
    if(lResult != 0)
    {
        AfxMessageBox("Failed to close control”);
        return;
    }

    delete API;
```

See the VC.NET project “RemoteAPI_Test” on the Remote API Installation disk for a comprehensive C++ example

5.3 VBScript

The following is a hypertext file which includes the CZEMAPI OCX and calls methods in the control using VBScript. The window_onLoad() subroutine will be executed when the HTML file is loaded into Internet Explorer 3.0 or higher.

```
<HTML>
<HEAD>
    <SCRIPT LANGUAGE="VBScript">
<!--
Sub window_onLoad()
Dim Value
‘ Connect to a local EM Server
call Api1.Initialise("")
call Api1.Get("AP_MAG",Value)
call window.alert("Magnification = " + CStr(Value))
call Api1.ClosingControl()
end sub
-->
    </SCRIPT>
<TITLE>New Page</TITLE>
</HEAD>
<BODY>
    <OBJECT ID="Api1" WIDTH=98 HEIGHT=50
     CLASSID="CLSID:71BD42C4-EBD3-11D0-AB3A-444553540000">
        <PARAM NAME="_Version" VALUE="65536">
        <PARAM NAME="_ExtentX" VALUE="2096">
        <PARAM NAME="_ExtentY" VALUE="1058">
        <PARAM NAME="_StockProps" VALUE="0">
    </OBJECT>
</BODY>
</HTML>
```

6. Parameter Mnemonics

This section contains a list of TEM parameters and commands. The list is by no means complete and certain parameters will be subject to change.

6.1 Analogue Parameters

6.1.1 General

AP_MAG_INDEX		Mag Index
AP_TEM_MAG	RO	Mag
AP_ILL_INDEX		Illum. Index
AP_ILL_ANGLE	RO	Illum. Angle
AP_SPOT_SIZE	RO	Spot Size
AP_CAMERA_LENGTH	RO	CL

6.1.2 Gun

AP_ACTUAL_FIL_CURRENT	RO	Fil I is
AP_TARGET_FIL_CURRENT		Fil I
AP_EHT_INDEX		HT Step
AP_ACTUAL_EHT	RO	HT is
AP_TARGET_EHT	RO	HT
AP_WOBBLE_AMP		EHT Wobble Amplitude
AP_ACTUAL_EMISSION_I	RO	Emission I is
AP_ACTUAL_EMISSION_STEP	RO	Emission Step is
AP_TARGET_EMISSION_STEP		Emission Step
AP_VAR_VOLTAGE		Var. Voltage
AP_DELTA_E_AMP		Delta E is
AP_DELTA_E1		Delta E 1
AP_DELTA_E2		Delta E 2
AP_TARGET_EXTRACTOR_V		Extractor V
AP_ACTUAL_EXTRACTOR_V	RO	Extractor V is
AP_FILAMENT_PRECENTERING_X AP_FILAMENT_PRECENTERING_Y		Gun Shift X Gun Shift Y
AP_BEAM_ALIGN_X AP_BEAM_ALIGN_Y		Gun Tilt X Gun Tilt Y

6.1.3 Column

AP_ILL_SHIFT_X AP_ILL_SHIFT_Y		Illum. Shift X Illum. Shift Y
AP_ILL_TILT_X AP_ILL_TILT_Y		Illum. Tilt X Illum. Tilt Y
AP_ILL_STIG_X AP_ILL_STIG_Y		Illum. Stig X Illum. Stig Y
AP_IMAGE_STIG_X AP_IMAGE_STIG_Y		Obj. Stig X Obj. Stig Y
AP_IMAGE_SHIFT_X AP_IMAGE_SHIFT_Y		Image Shift X Image Shift Y
AP_DEFOCUS		Defocus
AP_FOCUS		Focus
AP_C1_LENS		C1 Lens I
AP_C2_LENS		C2 Lens I
AP_C3_LENS		C3 Lens I
AP_OBJECTIVE_LENS		Objective Lens I
AP_P1_LENS		P1 Lens I
AP_P2_LENS		P2 Lens I
AP_P3_LENS		P3 Lens I
AP_P4_LENS		P4 Lens I
AP_P5_LENS		P5 Lens I
AP_P6_LENS		P6 Lens I
AP_SPEC_MAG_INDEX		Spec.Mag Index
AP_SPECTRUM_MAG	RO	Spec. Mag

6.1.4 Micro Dose Focusing

AP_MDF_ILL_OFFSET_X AP_MDF_ILL_OFFSET_Y		MDF Illumination Offset X MDF Illumination Offset Y
AP_MDF_TILT_CORR_X AP_MDF_TILT_CORR_Y		MDF Tilt Corrn X MDF Tilt Corrn Y
AP_MDF_IMAGE_OFFSET_X AP_MDF_IMAGE_OFFSET_Y		MDF Image Offset X MDF Image Offset Y

6.1.5 Stage / Goniometer

AP_STAGE_AT_X AP_STAGE_AT_Y AP_STAGE_AT_Z AP_STAGE_AT_T AP_STAGE_AT_R AP_STAGE_AT_M	RO	X Y Z Alpha (Tilt) R Beta (Tilt)
AP_STAGE_GOTO_X AP_STAGE_GOTO_Y AP_STAGE_GOTO_Z AP_STAGE_GOTO_T AP_STAGE_GOTO_R AP_STAGE_GOTO_M		Goto X Goto Y Goto Z Goto Alpha Tilt Goto R Goto Beta Tilt
AP_STAGE_HIGH_X AP_STAGE_HIGH_Y AP_STAGE_HIGH_Z AP_STAGE_HIGH_T AP_STAGE_HIGH_M		High X High Y High Z High Alpha T High Beta Tilt
AP_STAGE_LOW_X AP_STAGE_LOW_Y AP_STAGE_LOW_Z AP_STAGE_LOW_T AP_STAGE_LOW_M		Low X Low Y Low Z Low Alpha Tilt Low Beta Tilt
AP_STAGE_DELTA_X AP_STAGE_DELTA_Y AP_STAGE_DELTA_Z AP_STAGE_DELTA_T AP_STAGE_DELTA_R AP_STAGE_DELTA_M		Delta X Delta Y Delta Z Delta Alpha Tilt Delta R Delta Beta Tilt

6.2 Digital Parameters / States

6.2.1 General

DP_SUPERVISOR	RO	Supervisor	0 1	No Yes
DP_CALIBRATION	RO	Calibration	0 1	Disabled Enabled
DP_MOUSE		Mouse	0 1	Coarse Fine
DP_PROGRESS		Progress	0 1 2 3 4 5	Idle Filament.. EHT ... Stage Init... Auto Function Integrating ..

6.2.2 Modes

DP_ILL_MODE	Illumination Mode	0 1	TEM Spot
DP_MAG_MODE	Mag Mode	0 1	Mag Low Mag
DP_IMAGE_MODE	Image Mode	0 1	Image Diffraction
DP_RESOLUTION	Resolution	0 1	Normal High Res
DP_FILTER_SPEC	Spectroscopy mode	0 1	ESI EELS
DP_BFDF	BFDF	0 1	Bright Field Dark Field

6.2.3 Vacuum

DP_VAC_READY	RO	Vac ready	0 1	No Yes
DP_VACSTATUS	RO	VAC Status	0 1 2 3 4 5 6 7 8 9 10 11 12 13	not ready ready wait column wait gun wait camera wait airlock wait V9 wait V3 wait gun pressure wait column pressure wait Camera pressure wait Fil Interlock Error Waiting EHT Clearance

6.2.4 Gun

DP_FILAMENT_IS	RO	Fil State	0 1 2 3 4 5	Off On Ramping Shutting Down To Standby Standby
DP_FILAMENT_FAIL	RO	Filament	0	No

		failed	1	Yes
DP_EHT_IS	RO	HT State	0 1 2 3 4	Off On Ramping Shutting Down Paused
DP_WOBBLE		EHT Wobble	0 1	Off On
DP_DELTA_E		Delta E	0 1	Off On
DP_DELTAE_USE		Delta E use	0 1	Delta E1 Delta E2
DP_HCI		HCl/0eV	0 1 2	HCl 0 eV User
DP_VAR_VOLTS		Var. Volts	0 1	Off On

6.2.5 Column

DP_COLUMN_TYPE		Column Type	0 1 2 3 4 5 6 7	LEO 1901 LEO 910 Libra 120 LEO 1903 Libra 200 LEO 1905 PACEM SESAME
DP_FOCUS_AID		Focus Aid	0 1 2	Off On - X On - Y
DP_CONDENSER_SETUP		Condenser Setup	0 1	Off On
DP_OBJECTIVE_SETUP		Objective Setup	0 1	Off On
DP_PROJ_SETUP		Proj Setup	0 1	Off On
DP_WOBBLE_OBJ		Wobble Objective	0 1	Off On
DP_MAG_CF		Mag CF	0 1	Coarse Fine
DP_WOBBLE_IMAGE		Wobble Image	0 1	Off On
DP_BEAMSTOP		Beam Stop	0 1	Out In
DP_FARADAY_CAGE		Faraday cage	0 1	Absent Present
DP_PROBE_I_METER		Probe I Meter	0 1 2 3	Small Screen Faraday Cage Shutter Collector Off
DP_ILL_CF		Illuminatio n CF	0 1	Coarse Fine
DP_BEAMSTOP_FITTED		Beamstop fitted	0 1	No Yes

6.2.6 Apertures

DP_AIS		AIS	0 1	Off Auto

			2	Manual
DP_AIS_N	R O	AIS No.	0 1 2 3 4 5 6	0 1 2 3 4 5 6
DP_MIS_N		MIS No.	0 1 2 3 4 5 6	0 1 2 3 4 5 6
DP_CONDENSER_AP		Condenser Aperture	0 1 2 3	Out 1 2 3
DP_CA_CONFIG		Cond.Aper.Config.	0 1 2 3 4 5	1, 5, 1 1, 7, 1 7, 1, 7 5, 1, 5 5, 1, 7 7, 1, 5
DP_CA_AUTO		Cond.Aper.Auto	0 1	No Yes
DP_APER_ID		Aperture	0 1 2 3 4 5	Condenser Pre Field Objective S.A. S.E. Slit
DP_APERTURE_STOP		Aperture Stop	0 1 2 3	0 1 2 3
DP_OA_AUTO		Obj.Aper.auto	0 1	No Yes
DP_SA_AUTO		SA Aper.Auto	0 1	No Yes
DP_SE_AUTO		Filter In Aper.Auto	0 1	No Yes
DP_SLA_AUTO		Slit Aper.Auto	0 1	No Yes
DP_SLIT_N		Slit	1 2 3 4 5 6 7	1 2 3 4 5 6 7

6.2.7 Stage / Goniometer

DP_STAGE_SUBSYSTEM	RO	Stage Subsystem	0 1	Absent Present

DP_STAGE_INIT	RO	Stage Initialised	0 1	No Yes
DP_STAGE_IS	RO	Stage Is	0 1	Idle Busy
DP_X_AXIS_IS Etc…	RO	X Axis Is	0 1	Idle Busy
DP_X_LIMIT_HIT Etc…	RO	X Limit Hit	0 1 2 3 4 5 6	None Low Outer Low Inner Low User High User High Inner High Outer
DP_STAGE_UNDO	RO	Stage Undo Goto	0 1	Invalid Valid
DP_X_AXIS Etc…	RO	X Axis	0 1	Absent Present
DP_X_ENABLED Etc…		X Enabled	0 1	No Yes
DP_STAGE_POINTS_LIST		Points List	0 1 2 3 4	Empty Not Started In List At End At Start
DP_STAGE_LIST		Stage List	0 1 2 … 9	Stage Reg 1 Reg 2 … Reg 9

6.2.8 Camera

DP_LARGE_SCREEN		Large Screen	0 1 2 3	Up Down Moving Error
DP_SMALL_SCREEN		Small Screen	0 1 2 3	Up Down Moving Error
DP_SHUTTER_COLLECTOR		Shutter Collector	0 1	Absent Present
DP_BLANK_BEAM		User Blank	0 1	No Yes
DP_CAMERA_SHUTTER		Camera Shutter	0 1	Closed Open
DP_CCD_BLANK		CCD Blank	0 1	No Yes
DP_BLANKED	RO	Blanked	0 1 2 3 4 5 6 7 8 9 10 11	No User Blank Camera Ext Beam Blank Ext Shutter Blanker 5 STEM Frozen Remote CCD Blanker 8 Blanker 9 Blanker 10 Blanker 11

			12 13 14 15 16	Blanker 12 Blanker 13 Sample Holder Gun Valve X-Ray

6.2.9 STEM

DP_STEM_MODE		STEM Mode	0 1	Off On
DP_STEM		STEM Detector	0 1	Absent Present
DP_STEM_Q1		STEM Q1	0 1 2	Off Normal Inverted
DP_STEM_Q2		STEM Q2		
DP_STEM_Q3		STEM Q3		
DP_STEM_FAST		STEM Fast	0 1	No Yes
DP_STEM_GAIN		STEM Gain	0 1 2 3	Low Medium High Very High
DP_STEM_AUTO		STEM Auto	0 1	Off On
DP_DSA_PRESENT		DSA Present	0 1	No Yes

6.2.10 Micro Dose Focusing

DP_MDF		MDF	0 1	Off On
DP_MDF_N		MDF No.	0 1 2 3 4	1 2 3 4 5
DP_MDF_PARALLEL		MDF parallel	0 1	No Yes
DP_MDF_CHECK		MDF Check	0 1	Off On
DP_MDF_MIS_N	RO	MDF At MIS No.	0 1 2 3 4 5 6	0 1 2 3 4 5 6

6.3 Commands

6.3.1 Vacuum

CMD_COLUMN_PUMP	Column Pump
CMD_COLUMN_VENT	Column Vent
CMD_CAMERA_PUMP	Camera Pump
CMD_CAMERA_VENT	Camera Vent
CMD_GUN_PUMP	Gun Pump
CMD_GUN_VENT	Gun Vent

6.3.2 Gun

CMD_FILAMENT_ON	Filament On
CMD_FILAMENT_OFF	Filament Off
CMD_EHT_ON	HT On
CMD_EHT_OFF	HT Off
CMD_20KV	20 kV
CMD_40KV	40 kV
CMD_60KV	60 kV
CMD_80KV	80 kV
CMD_100KV	100 kV
CMD_120KV	120 kV
CMD_160KV	160 kV
CMD_200KV	200 kV
CMD_RECALL_GUN_ALIGN	Recall Gun Alignment
CMD_FEG_ON	FEG On
CMD_FEG_OFF	FEG Off

6.3.3 Column

CMD_ZERO_DEFOCUS	Zero Defocus
CMD_CAL_ILL_DEFL	Cal illumination defl
CMD_CAL_IMAGE_SHIFT	Cal image Shift
CMD_CAL_ILL_STIG	Cal illumination Stig
CMD_CAL_FOCUS	Cal Focus
CMD_SAVE_USER_DATA	Save User Data
CMD_CAL_MAG	Cal mag
CMD_CAL_IMAGE_STIG	Cal image Stig
CMD_CAL_ALL	Calibrate All
CMD_MAG_UP	Mag +
CMD_MAG_DOWN	Mag -
CMD_ILL_UP	Illumination +
CMD_ILL_DOWN	Illumination -
CMD_EMISSION_UP	Emission +
CMD_EMISSION_DOWN	Emission -
CMD_SAVE_LENS_PROG	Save Lens Program
CMD_SAVE_CALIBRATION	Save Calibration
CMD_LOAD_USER_DATA	Load User Data
CMD_TRIGGER_UNBLANK	Trigger Unblank
CMD_STORE_IMAGE_SHIFT	Store Image Shift
CMD_RECALL_IMAGE_SHIFT	Recall Image Shift
CMD_STORE_ILL_SHIFT	Store Illumination Shift
CMD_RECALL_ILL_SHIFT	Recall Illumination Shift
CMD_STORE_ILL_TILT	Store Illumination Tilt
CMD_RECALL_ILL_TILT	Recall Illumination Tilt
CMD_STORE_IMAGE_STIG	Store Image Stig
CMD_RECALL_IMAGE_STIG	Recall Image Stig
CMD_STORE_ILL_STIG	Store Illumination Stig.
CMD_RECALL_ILL_STIG	Recall Illumination Stig
CMD_STORE_ZOOM_VALUES	Store Zoom Values
CMD_RECALL_ZOOM_VALUES	Recall Zoom Values
CMD_STORE_C3	Store C3 Value
CMD_RECALL_C3	Recall C3 Value
CMD_SAVE_DF	Save DF setting
CMD_RESTORE_DF	Restore DF Setting
CMD_MAGNETIC_RESET	Start Magnetic Reset
CMD_ABORT_MAGNETIC_RESET	Magnetic Reset Abort
CMD_CAL_CONDENSER	Cal. Condenser

CMD_STORE_MAG_STEP	Store Mag Step
CMD_RECALL_MAG_STEP	Recall Mag Step
CMD_STORE_P4ZL	Store Proj II values
CMD_RECALL_P4ZL	Recall Proj II values
CMD_RELOAD_CALIBRATION	Reload Calibration
CMD_RELOAD_LENS_PROGRAM	Reload Lens Program
CMD_MDF_SAVE	MDF Save
CMD_MDF_RESTORE	MDF Restore
CMD_LOAD_USERDATA_ID	Load UserData (ID)

6.3.4 Stage / Goniometer

CMD_STAGE_INIT	Stage Initialise
CMD_STAGE_ABORT	Stage Abort
CMD_UNDO_STAGE_GOTO	Undo Stage Goto

6.3.5 Micro Dose Focusing

CMD_MDF_SAVE	MDF Save
CMD_MDF_RESTORE	MDF Restore
CMD_MDF_REVERSE	MDF Reverse

6.3.6 Camera

CMD_PHOTO	Photo
CMD_ABORT_SERIES	Abort Series
CMD_SMALL_SCREEN_UP	Small Screen Up
CMD_SMALL_SCREEN_DOWN	Small Screen Down
CMD_LARGE_SCREEN_UP	Large Screen Up
CMD_LARGE_SCREEN_DOWN	Large Screen Down

6.3.7 Filter

CMD_SAVE_SLIT	Save Slit
CMD_RESTORE_SLIT	Restore Slit
CMD_STORE_SPECTRUM_VALUES	Store Filter Values
CMD_RECALL_SPECTRUM_VALUES	Recall Filter Values
CMD_CAL_FILTER	Cal. Filter

7. Error return codes

API_E_GET_TRANSLATE_FAIL	1000	Failed to translate parameter into an id
API_E_GET_AP_FAIL	1001	Failed to get analogue value
API_E_GET_DP_FAIL	1002	Failed to get digital value
API_E_GET_BAD_PARAMETER	1003	Parameter supplied is not analogue nor digital
API_E_SET_TRANSLATE_FAIL	1004	Failed to translate parameter into an id
API_E_SET_STATE_FAIL	1005	Failed to set a digital state
API_E_SET_FLOAT_FAIL	1006	Failed to set a float value
API_E_SET_FLOAT_LIMIT_LOW	1007	Value supplied is too low
API_E_SET_FLOAT_LIMIT_HIGH	1008	Value supplied is too high
API_E_SET_BAD_VALUE	1009	Value supplied is is of wrong type
API_E_SET_BAD_PARAMETER	1010	Parameter supplied is not analogue nor digital
API_E_EXEC_TRANSLATE_FAIL	1011	Failed to translate command into an id

API_E_EXEC_CMD_FAIL	1012	Failed to execute command
API_E_EXEC_MCF_FAIL	1013	Failed to execute file macro
API_E_EXEC_MCL_FAIL	1014	Failed to execute library macro
API_E_EXEC_BAD_COMMAND	1015	Command supplied is not implemented
API_E_GRAB_FAIL	1016	Grab command failed
API_E_GET_STAGE_FAIL	1017	Get Stage position failed
API_E_MOVE_STAGE_FAIL	1018	Move Stage position failed
API_E_NOT_INITIALISED	1019	API not initialised
API_E_NOTIFY_TRANSLATE_FAIL	1020	Failed to translate parameter to an id
API_E_NOTIFY_SET_FAIL	1021	Set notification failed
API_E_GET_LIMITS_FAIL	1022	Get limits failed
API_E_GET_MULTI_FAIL	1023	Get multiple parameters failed
API_E_SET_MULTI_FAIL	1024	Set multiple parameters failed
API_E_NOT_LICENSED	1025	Missing API license
API_E_NOT_IMPLEMENTED	1026	Reserved or not implemented
API_E_GET_USER_NAME_FAIL	1027	Failed to get user name (Remoting Interface only)
API_E_GET_USER_IDLE_FAIL	1028	Failed to get user idle state (Remoting Interface only)
API_E_GET_LAST_REMOTING_CONNECT_ERROR_FAIL	1029	Failed to get the last remoting connection error string (Remoting Interface Only)
API_E_EMSERVER_LOGON_FAILED	1030	Failed to remotely logon to the EM Server (username and password may be incorrect or EM Server is not running or User is already logged on, Remoting only)
API_E_EMSERVER_START_FAILED	1031	Failed to start the EM Server. This may be because the Server is already running or has an internal error
API_E_REMOTING_NOT_CONFIGURED	2027	Remoting incorrectly configured, use RConfigure to correct
API_E_REMOTING_FAILED_TO_CONNECT	2028	Remoting did not connect to the server
API_E_REMOTING_COULD_NOT_CREATE_INTERFACE	2029	Remoting could not start (unknown reason)
API_E_REMOTING_EMSERVER_NOT_RUNNING	2030	Remoting: Remote server is not running currently
API_E_REMOTING_NO_USER_LOGGED_IN	2031	Remoting: Remote server has no user logged in

Carl Zeiss SMT Ltd
511 Coldhams Lane
Cambridge CB1 3JS
UK
Tel: +44 1223/ 414166
Fax: +44 1223/ 412776
Info-uk@smt.zeiss.com

Carl Zeiss NTS GmbH
A Carl Zeiss SMT AG Company
Carl-Zeiss-Str. 56
73447 Oberkochen
Germany
Tel. +49 73 64 / 20 44 88
Fax +49 73 64 / 20 43 43
info-nts@smt.zeiss.com

Carl Zeiss SMT Inc
One Corporation Way,
Peabody, MA 01960 USA
Tel. (978) 826-1500
Fax (978) 532-5696
info-usa@smt.zeiss.com

Carl Zeiss SMT SAS
Zone d´Activité des Peupliers
27, rue des Peupliers – Batiment A
92000 NANTERRE
France
Tel. +3 3141 / 39 9210
Fax +3 3141 / 39 92 29
info-fr@smt.zeiss.com

Carl Zeiss SMT Pte Ltd
50 Kaki Bukit Place #04-01
Singapore 415926
Tel. +65 65 67 / 3011
Fax +65 65 67 / 5131
info.sea@smt.zeiss.com

Plus a worldwide network of authorised distributors

www.smt.zeiss.com

Due to a policy of continuos development, we reserve the right to change specifications without notice.

Printed in the UK