Programming Manual

SmartSEM Remote API

Programming the API

ZEISS

SmartSEM Remote API

Programming the API

Abstract

This document describes the SmartSEM® Application Programming Interface which allows external programs to communicate with SmartSEM® software.

Part Number: 351434-6036-006

Date: 22 November 2011

Issue: 3.6

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 Carl Zeiss Scanning Electron Microscopes (SEM). The application programming interface (API) consists of an Active X control that provides access to the SmartSEM software and via this software to the SEM microscope settings. The following series of microscopes are supported:

- 1400 Series
- 1500 Series
- EVO Series
- Supra Series
- Ultra Series
- CrossBeam Series
- Merlin
- Auriga
- Sigma

As an Active X control the API can be accessed by various programming languages such as Visual Basic, Visual C++, 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++ or C# so the example code that is on the installation CD-ROM is written in both C++ and C# (Visual Studio 2008).

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

Carl Zeiss NTS 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 SmartSEM software is the EM Server. The Server maintains a list of states and parameters of the SEM. The SmartSEM 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.

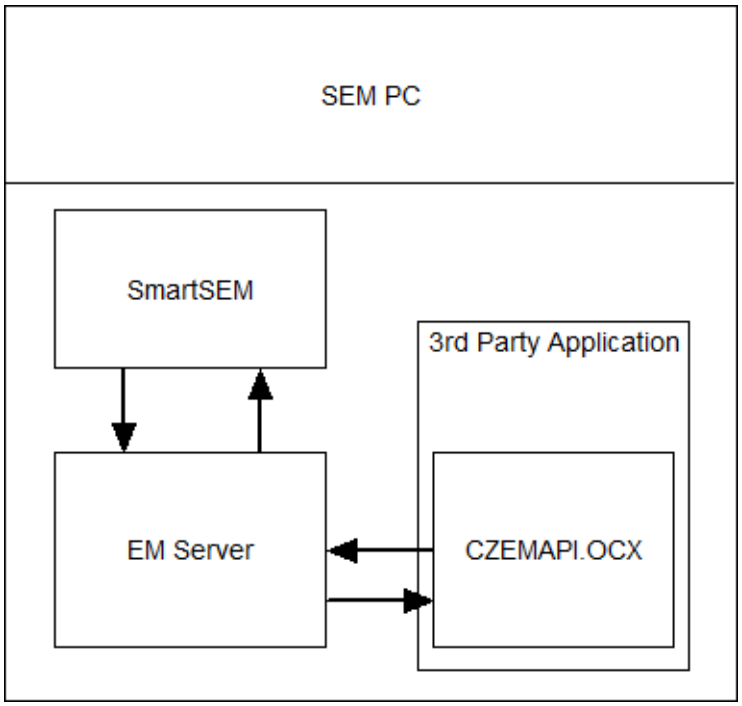

Figure 1: 3rd Party App running on SEM PC

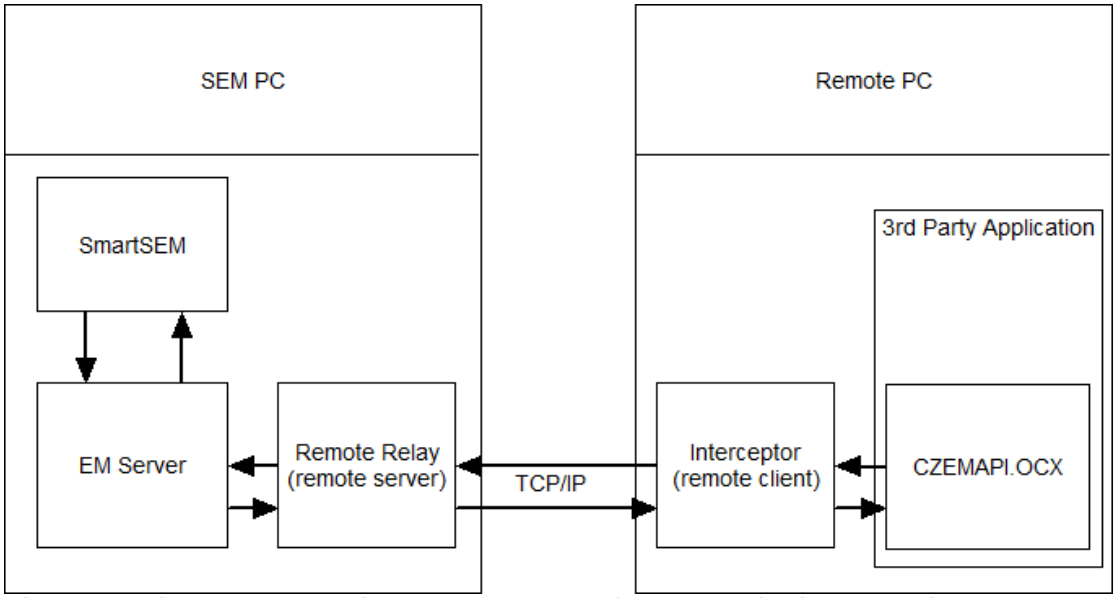

Figure 2: 3rd Party App running on remote PC. The communication goes via a LAN network.

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 SEM 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 client 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 CZ EM API Control you just refer to the parameter symbol as a string (e.g. “AP_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 value are two very different 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 SEM, 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_ 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 Stage 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 Stage X position.

2.2.2 Digital Parameters (states)

Digital parameters are those parameters that have a discrete set of values. For example, the digital parameter “Spot Mode” (DP_SPOT) 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_STAGE_IS. 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 “Beam State” which includes not only the static states like “EHT Off” or “Beam 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 String Parameters

There are a small number of string parameters, e.g. SV_SERIAL_NUMBER = “1540XB-29-03”. Some are read-only and others are editable.

2.2.4 Commands

Commands are functions that initiate some action. E.g. “Beam On”.

Every command has a unique numerical identity which is associated with a symbol starting CMD_ , e.g. CMD_BEAM_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)

2.2.5 Groups

These are a special combination of a Read-Only state and the group commands that affect the state. Example is “Beam State” and the associated commands are “Shutdown Gun”, “EHT Off” and “Gun On”.

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 SmartSEM 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 SmartSEM software

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

The SmartSEM installation will also install the remote server program on the PC. This will include the .NET Framework 3.5 (Service Pack 1). The remote server is called “Remote Relay.exe” and 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/7 security will detect this activity and block it. You must now manually allow RemoteRelay to access the network and instruct Windows 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 Remote API 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 security will detect this activity and block it. You must now manually allow Interceptor to access the network and instruct Windows to remember this decision.

3.3 SmartSEM versions

In the table below we list the versions of SmartSEM and the compatible version of Remote API. It is important to note that compatibility is generally only really an issue between versions 1.x and 2.x (and above) of Remote API as the architecture was significantly (and unavoidably) changed between these versions. We would, however, advise that versions of the server and client should match where possible.

SmartSEM Version	Patch/ Service Pack	Remote API	
v5.01	all	1.0	
v5.02	-	2.0	
v5.02	1	2.2	
v5.02	2	2.2	
v5.02	3	2.3	
v5.03	all	2.4.1	
v5.04	All	2.9	
y5.05	Beta 14	3.0	Note: Beta release for Merlin and Auriga 60
v5.05	-	3.6	

3.4 Remote API client and server versions

The RConfigure program will display the Remote Client or Server versions, depending whether you run the program on the Remote computer or the SmartSEM computer. You can also read the version in the registry:

Client: HKEY_CLASSES_ROOT\Zeiss Cambridge Virtualised Store\Carl Zeiss SMT Ltd\Remoting\Version

Server: HKEY_CLASSES_ROOT\Zeiss Cambridge Virtualised Store\Carl Zeiss SMT Ltd\Remoting\ServerVersion

Note for 3rd parties: It is the responsibility of the 3rd party software integrator to make sure their PC is correctly configured. This includes installation of the correct Remote Client that is compatible with the Server version that is on the SmartSEM PC.

3.5 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 now deprecated.

3.6 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, CZEMApi.cpp 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.

3.7 Inserting the CZEMApi ActiveX control into a C# Application

From your C# form App in Visual Studio chose the menu "Project>Add Reference". From the "COM" tab select the CZ EM API OLE Control. This will insert the class ApiClass into the project.

If the CZ EM API OLE 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.

Add the following line to insert the namespace into your class

```
using APILib;
```

You can then create an ApiClass object in the form class. 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 "InitialiseRemoting". 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 CZ EM API OCX

Remarks:

This function is now deprecated.

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_MAG”

[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_XXXX.

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_MAG”

[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_WD would return “= 10mm”).

For digital parameters, VT_R4 variants result in a state number and VT_BSTR variants result in a state string (e.g. DP_RUNUPSTATE would return state 0 or “Shutdown” or the equivalent in the language being supported).

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. Shutdown
Digital	VT_R4	VT_R4	State number e.g. 0
String	VT_BSTR	VT_BSTR	string

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_MAG

[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). 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.

SmartSEM stores almost all analogue parameters in SI units. The preferred way is to set them as floating point values with VT_R4.

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_SCAN_ROT can be set to 0 or "Off" or the equivalent in the language being supported).

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)
API_E_PARAMETER_IS_DISABLED:	The command or parameter is currently disabled (you cannot execute or set it).

Note on Errors:
The error API_E_PARAMETER_IS_DISABLED has been introduced in Remote API 3.1. It is quite possible that a parameter or state is disabled. This depends on modes and rules. With the release testing of SmartSEM 5.5 (Compatible with Remote API 3.6) we found incompatibilities with older 3rd party software which did not recognise this error. Therefore in Remote API 3.6 this error is suppressed: the Set Method will fail silently on disabled parameters as before. Use registry key on client side:
[HKEY_CLASSES_ROOT\Zeiss Cambridge Virtualised Store\Carl Zeiss SMT Ltd\Remoting]
"SuppressParameterDisabledErrors"=dword:00000000 // allow error
"SuppressParameterDisabledErrors"=dword:00000001 // suppress error (default)

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_BEAM_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 … SmartSEM\DISTRIB directory or the user directory (e.g. C:\Program Files\Carl Zeiss NTS Ltd\SmartSEM\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_)

API_E_PARAMETER_IS_DISABLED: The command or parameter is currently disabled (you cannot execute or set it).

Note on Errors:
The error API_E_PARAMETER_IS_DISABLED has been introduced in Remote API 3.1. It is quite possible that a command is disabled. This depends on modes and rules. With the release testing of SmartSEM 5.5 (Compatible with Remote API 3.6) we found incompatibilities with older 3rd party software which did not recognise this error. Therefore in Remote API 3.6 this error is suppressed: the Execute Method will fail silently on disabled parameters as before. Use registry key on client side:
[HKEY_CLASSES_ROOT\Zeiss Cambridge Virtualised Store\Carl Zeiss SMT Ltd\Remoting]
"SuppressParameterDisabledErrors"=dword:00000000 // allow error
"SuppressParameterDisabledErrors"=dword:00000001 // suppress error (default)

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 tilt of the stage/degrees

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

[out] *m* is the master-z of the stage/metres

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

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 master-z of the stage/metres

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

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 with the offset, dimensions and subsampling supplied and saves it as the file specified by lpszFilename as either a Windows Bitmap or a Tiff image depending on the given file extension. Tiff has the Zeiss header with acquisition parameter information. 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.

When this command is used on the same PC as the SmartSEM is running you can give “CZ.MMF” as lpszFilename parameter to instruct SmartSEM to produce a memory map file (MMF) containing the bitmap bits. The image data in the MMF can then be used for your own needs. NOTE: When the MMF data is no longer used Grab must be called a second time with “CZ.MMF” as lpszFilename parameter and 0 for all other parameters to instruct SmartSEM to free the memory allocated to the MMF.

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 SmartSEM 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 SmartSEM user name and password. It is not needed to run the SmartSEM UIF if you logon via this method. If SmartSEM 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 SmartSEM 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 SmartSEM 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 is something 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 (SmartSEM V5.2 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 CApi:

```
#include "api_err.h" // Error codes

API = new CApi;
// 'this' specifies a CWnd derived class
// IDD_API is the identifier of the CZ EM API 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 CZ EM API OCX control");
     return;
     }

// Initialise the control to connect to the local CZ EM
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 projects "TestAPI_2004" (VS2003) and SmartSEM_API_Test (VS2005) on the Remote API Installation disk for more C++ examples

5.3 VBScript/JavaScript/JScript

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>
```

5.4 Visual C#

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

```
using APILib;

ApiClass CZEMApi = new ApiClass();

// Initialise the control to connect to the local CZ EM
microscope
long lReturn = CZEMApi.Initialise("");
if (lReturn != 0)
{
    MessageBox.Show("Failed to initialise control");
    return;
}
// Get the magnification value from the microscope
object varfloat = new VariantWrapper((float)0.0f);

// get param (numeric)value
long lResult = CZEMApi.Get("AP_MAG", ref varfloat);
if (lResult != 0)
{
    MessageBox.Show("Failed to get magnification");
    return;
}

MessageBox.Show("Magnification = " + varfloat.ToString());

// Tidy up
lResult = CZEMApi.ClosingControl();
if (lResult != 0)
{
    MessageBox.Show("Failed to close control");
    return;
}
```

See the VC.NET projects SmartSEM_API_Test (VS2008) on the Remote API Installation disk for more C# examples

6. Parameter Mnemonics

The concept of commands, analogue and digital parameters has been explained in section 2.2. This section provides a list of most commonly used parameter mnemonics, together with a brief description. If you want to know more about the functionality and validity of parameters then please ask the Carl Zeiss NTS software team for more information. Also, if you want to know about other parameters that are not listed here, then contact us.

Most parameters and commands have restrictions on their use. RO means „Read-only“; „EVO“ means „EVO or 1400 machine type“; „Lic“ means that a Licence is required for this functionality; „Sup“ means that „Supervisor privilege is needed for this functionality“; „FIB“ means that a „Focused Ion Beam must be fitted“. Apart from these restictions the validity of parameters and commands often depends on other modes and states.

6.1 Analogue Parameters

6.1.1 Gun

Mnemonics	Description	Units	Restrictions
AP_GUNTILT_X, Y	Gun Tilt	%	EVO
AP_GUNSHIFT_X, Y	Gun Shift	%	EVO
AP_GUNALIGN_X, Y	Gun Align	%	
AP_EXTCURRENT	Extractor Current	Amperes	RO
AP_MANUALEXT	Extractor V Target	Volts	
AP_MANUALKV	EHT Target	Volts	
AP_ACTUALKV	EHT	Volts	RO
AP_ACTUALCURRENT	Filament Current	Amperes	RO
AP_FILAMENT_AGE	Filament Age	Hours	

6.1.2 Beam

AP_BEAMSHIFT_X, Y	Beam Shift	%	
AP_BEAM_OFFSET_X, Y	Beam Offset	Metres	RO
AP_BRIGHTNESS	Brightness	%	
AP_CONTRAST	Contrast	%	
AP_MAG	Magnification	X	
AP_WD	Working Distance	Metres	
AP_SPOT	Spot Signal		RO
AP_PIXEL_SIZE	Pixel Size	Metres	RO
AP_SCM	Specimen Current	Amperes	RO
AP_SPOTSIZE	Relative Spot Size	-	
AP_IPROBE	Probe Current	Amperes	
AP_STIG_X, Y	Stigmation	%	
AP_AUTO_BRIGHT	Auto Brightness Target	%	
AP_AUTO_CONTRAST	Auto Contrast Target	%	
AP_ZOOM_FACTOR	Zoom Factor	X	
AP_TILT_ANGLE	Tilt Angle	Degrees	

6.1.3 Scanning

AP_SPOT_POSN_X, Y Spot PositionScreen co-ordinates	Spot at X, Y	Screen pixels	
AP_LINE_POSN_X, Y	Line position	Screen pixels	
AP_LINE_LENGTH	Line Length	Screen pixels	
AP_SCANROTATION	Scan Rotation	Degrees	
AP_PIXEL_SIZE	Pixel Size	Metres	RO
AP_LINE_TIME	Line Time	Milliseconds	RO
AP_FRAME_TIME	Cycle Time	Milliseconds	RO
AP_FRAME_AVERAGE_COUNT	Frames to average	-	
AP_FRAME_INT_COUNT	Frames to integrate	-	
AP_LINE_INT_COUNT	Line	-	

	integration count		
AP_RED_RASTER_POSN_X, Y	Reduced Raster Position	Screen pixels	
AP_RED_RASTER_W, H	Reduced Raster Width, Height	Screen pixels	
AP_LINE_AVERAGE_COUNT	Line average count	-	
AP_NR_COEFF	Noice Reduction Co-efficient	-	
AP_WIDTH	Width (of field of view)	Metres	
AP_HEIGHT	Height (of field of view)	Metres	

6.1.4 Apertures

AP_APERTURESIZE	Aperture Size	Metres	EVO
AP_APERTURE_ALIGN_X, Y	Aperture Align	%	
AP_APERTUREPOSN_X, Y	Aperture Position	Metres	

6.1.5 Detectors / Camera

AP_PHOTO_NUMBER	Photo number	-	
AP_GAMMA	Gamma	-	LIC
AP_COLLECTOR_BIAS	Collector Bias	Volts	

6.1.6 Stage

AP_STAGE_AT_X, Y, Z, T, R, M	Stage position	Metres	
AP_STAGE_GOTO_X, Y, Z, T, R, M	Target position	Metres	
AP_STAGE_HIGH_X, Y, Z, T, R, M	High limit	Metres	
AP_STAGE_LOW_X, Y, Z, T, R, M	Low limit	Metres	
AP_PIEZO_AT_X	Peizo position	Metres	
AP_PIEZO_GOTO_X, Y	Piezo target	Metres	

6.1.7 Vacuum

AP_HP_TARGET	Variable Pressure Target	Bar	
AP_SYSTEM_VAC	System vacuum	Bar	RO
AP_COLUMN_VAC	Gun Vacuum	Bar	RO
AP_CHAMBER_PRESSURE	Chamber pressure	Bar	RO, VP/EP

6.1.8 FIB

AP_FIB_MAGNIFICATION	FIB Mag	X	FIB
AP_FIB_OBJECTIVE_POTENTIAL	FIB Focus	Volts	FIB
AP_FIB_EMISSION_CURRENT	FIB Emission Current	Amperes	RO, FIB
AP_FIB_STIGMATOR_X, Y	FIB Stig	%	FIB
AP_FIB_BEAM_SHIFT_X, Y	FIB Beam	Metres	FIB

	Shift		
AP_FIB_SCAN_ROTATION	FIB Scan Rot	Degrees	FIB
AP_FIB_EHT_TARGET	FIB EHT Target	Volts	FIB
AP_FIB_EHT_ACTUAL	FIB EHT	Volts	RO, FIB
AP_FIB_DRIFT_X, Y	FIB Drift	Metres	FIB

6.2 Digital Parameters

6.2.1 General

Mnemonic	Description	Rest.	Value	Text
DP_COLUMN_TYPE	Column		5 6 7 8 9 10 11 12 13 14 15	1400 1500 1400VP 1500VP 1400EP 1500XB 1400XB EVO MeRiT 1500XBVP NVision
DP_DC_SEM_STATE	Sem drift status		0 1 2 3 4 5	Not Available Busy No Reference Failed Success Ready
DP_PANEL_DISABLE	Panel Disable		0 1	No Yes
DP_JOYSTICK_DISABLE	Joystick Disable		0 1	No Yes

6.2.2 Modes

DP_OPERATING_MODE	Operating Mode	RO	0 1 2 3 4 5 6	Normal Split Quad Emission Small Reduced TV
DP_IMAGE_MODE	Image Mode	RO	0 1	Full Reduced
DP_SPOT	Spot Mode		0 1	Off On
DP_STEM	STEM		0 1	Absent Present

6.2.3 Gun

DP_FIL_BLOWN	Filament blown	RO	0 1 2	No Yes unknown
DP_RUNUPSTATE	Beam State	RO	0 1 2 3 4	Shutdown Beam On Running Up Shutting Down Running Down

			5 6 7 8 9 10 11 12 13 14 15	Beam Going Off Restoring Beam Standby EHT going off Beam Off EHT Off Run up Waiting Vac Service Runup Emergency Rundown Conditioning
DP_HIGH_CURRENT	High Current	LIC	0 1	Off On

6.2.4 Beam

DP_BEAM_BLANKED	Beam Blanked		0 1	No Yes
DP_BEAM_BLANKING	Beam Blanking		0 1	Off On
DP_AUTO_FUNCTION	Auto Function	RO	0 1 2 3 4 5 6 7 8 9 10	Idle Focus Gun Align Stigmation Saturate Find Peak Find Mean Find Image Focus Scan Focus+Stig ABC
DP_SCM_RANGE	Specimen Current Measurement range	RO	0 1 2 3 4 5 6 7 8 9 10 11 12	3->10 pA 10->30 pA 30->100 pA 100->300 pA 300->1000 pA 1->3 nA 3->10 nA 10->30 nA 30->100 nA 100->300 nA 300->1000 nA 1->3 microAmps 3->10 microAmps
DP_SCM	SCM fitted		0 1	No Yes
DP_AUTO_VIDEO	Auto Brightness Contrast		0 1 2 3	Off B C BC

6.2.5 Scanning

DP_SCAN_ROT	Scan Rotation		0 1	Off On
DP_FREEZE_ON	Freeze on		0 1 2	End Frame End Line Command
DP_LINE_SCAN	Line Scan		0 1	Off On

DP_EXT_SCAN_CONTROL	Ext.Scan Control		0 1	Off On
DP_MAX_RATE	Max Scan Speed	RO	0 1 2 3 4	0 1 2 3 4
DP_SCANRATE	Scan Speed	RO	1-15 16-21	1-15 16-21
DP_NOISE_REDUCTION	Noise Reduction	RO	0 1 2 3 4 5 6 7	Frame Int. Busy Line Int. Busy Frame Avg Line Int. Done Frame Int. Done Line Avg Pixel Avg. Continuous Avg.
DP_IMAGE_STORE	Store resolution		0 1 2 3 4 5 6	1024 * 768 512 * 384 2048 * 1536 3072 * 2304 4096 * 3072 5120 * 3840 6144 * 4608
DP_FROZEN	Image		0 1	Live Frozen
DP_LEFT_FROZEN	All Frozen	RO	0 1	No Yes
DP_RIGHT_FROZEN	Right Frozen	RO	0 1	No Yes
DP_DISPLAY_CHANNELS	Cannels	LIC	0 1 2 3	1 2 3 4
DP_AUTO_FUNCTION	Auto Function	RO	0 1 2 3 4 5 6 7 8 9 10	Idle Focus Gun Align Stigmation Saturate Find Peak Find Mean Find Image Focus Scan Focus+Stig ABC

6.2.6 Apertures

DP_APERTURE	Aperture Number		0 1 2 3 4 5 6	1 2 3 4 5 6 7
DP_APERTURE_STATE	Aperture		0 1	Clean Contaminated
DP_APERTURE_TYPE	Aperture Type		0 1	6 Hole Aperture 7 Hole Aperture

6.2.7 Detectors / Camera

DP_OUT_DEV	Output dev	RO	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	120 mm Camera 35 mm Camera Polaroid 545 Polaroid 550 14/15 inch display 17 inch display 19/21 inch display P68B video print P78B video print CP100B video print CP210U video print CP200BH video print Alden video print P66D video print Video Print Default Printer
DP_4QBSD_Q1, 2, 3, 4	Q1, 2, 3, 4		0 1 2	Off Normal Inverted
DP_4QBSD_VISIBLE	BSD Fast		0 1	No Yes
DP_4QBSD	BSD		0 1	Absent Present
DP_ZONE	Zone		0-7	0-7
DP_DETECTOR_CHANNEL	Signal A		0-14	0-14
DP_DETECTOR_TYPE	Detector		0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	None SE1 SE2 BSD RBSD SCD CL STEM TV InLens VPSE MPSE EP-SED ESB Wedge BSD Aux 1 Aux 2 Aux 3 Aux 4 VPSE G3 EPSE AsB CZ BSD SESI HE-SE2 Jazz BSD0 BSD1 BSD2

			29	BSD3
DP_HRRU_SPEED	Photo Fast		0 1	No Yes
DP_HRRU_PHOTO_STATUS	Camera status		0 1	Idle Busy
DP_HRRU_SOURCE	Camera Image		0 1 2 3	Stored Live Image Cal Bri. Cal Cont.

Note: The detectors need to be configured first in the Administrator program. You can insert detectors in channels 1-7. Then you can select from DP_DETECTOR_CHANNEL the detector you want to use, e.g. a value of "0" represents the detector in Channel 1. If you work with the numbers then you need to know in which channels the detectors are put. It is also possible to set a string value, e.g. “QBSD” but that detector need to be configured first.

6.2.8 Stage

DP_STAGE_TYPE	Stage Type	Don’t change!	7 always	Name of selected stage type
DP_STAGE_BACKLASH	Stage Backlash		0 1	Off On
DP_STAGE_INIT	Stage Initialised	RO	0 1	No Yes
DP_STAGE_IS	Stage Is	RO	0 1	Idle Busy
DP_STAGE_TOUCH	Stage Touching	RO	0 1	No Yes
DP_X_BACKLASH, Y, Z, T, R, M	X, Y,… Backlash		0 1 2 3	Off + - + -
DP_X_LIMIT_HIT, Y, Z, T, R, M	X, Y,… Limit hit	RO	0 1 2 3 4 5 6	None Low outer Low Inner Low user High user high inner high outer
DP_X_AXIS_IS, Y, Z, T, R, M	X, Y,… axis is	RO	0 1	Idle Moving
DP_X_AXIS, Y, Z, T, R, M	X, Y,… Axis	RO	0 1	Absent Present
DP_X_ENABLED, Y, Z, T, R, M	X, Y,… Enabled	Sup	0 1	No Yes
DP_STAGE_TILTED	Stage Tilt	RO	0 1	in X In Y
DP_JOYSTICK_DISABLE	Joystick Disable		0 1	No Yes
DP_STAGE_SCAN	Stage Scan		0 1 2 3 4 5	X Boustrophodon X Raster Y Boustrophodon Y Raster Spiral Minimum Stage Movement
DP_STAGE_SCANNING	Stage Scanning	RO	0 1	Idle Moving

			2 3	at field end scan

6.2.9 Vacuum

DP_COLUMN_CHAMBER_VALVE	Column Chamber valve	RO	0 1 2 3	Closed Open Disconnected Error
DP_COLUMN_PUMPING	Column pumping	RO	0 1 2 3 4	Idle Ready Pumping Error Requested
DP_COLUMN_PUMP	Column Pump	Sup	0 1	Absent Present
DP_HP_STATUS	Chamber Vacuum Status	RO	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Power Up At Air (VP) At Air (HV) Pumping (HV) Going to VP At VP Ready to Vent Ready to Pump Gun enabled Gun Delay Wait > MCP Wait < MCP Error VP mode error At Air (EP) At EP Going to EP Humidifying
DP_VACSTATUS	Vac Status	RO	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	At Air Ready Pumping Error Waiting Turbo Speed Waiting Penning Waiting EHT Vent Inhibited Going VP Repumping Gun Waiting for chamber door to close Waiting for compressed air Waiting for N2 Waiting for PVG Waiting for IGP1 Waiting for IGP2 Waiting for airlock
DP_VAC_MODE	Vacuum Mode		0 1 2	High Vacuum Variable Pressure Extended Pressure
DP_EP_OK	EP Mode OK		0 1	No Yes

DP_AIRLOCK	Airlock Monitor		0 1	Absent Present
DP_AIRLOCK_CONTROL	Airlock Control		0 1	No Yes
DP_AIRLOCK_READY	Airlock Ready	RO	0 1	No Yes
DP_EHT_VAC_READY	EHT Vac ready	RO	0 1	No Yes
DP_BAKEOUT	Bakeout	Sup	0 1 2 3	Quick Overnight Weekend User
DP_BAKEOUT_STATUS	Bakeout state	RO	0 1 2	Idle Heating Cooling

6.2.10 FIB

DP_FIB_MODE	FIB Mode	RO	0 1 2 3 4 5 6 7 8 9	Imaging Remote Milling Deposition To Imaging To Milling To Deposition To Remote To Wide View Wide View
DP_FIB_IMAGE_PROBE	FIB Image Probe Current		0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 - 30	Undefined 1 pA 2 pA 5 pA 10 pA 20 pA 50 pA 100 pA 200 pA 500 pA 1 nA 2 nA 5 nA 10 nA 20 nA 50 nA User probes
DP_FIB_MILL_PROBE	FIB Milling Probe Current		0- 30	As above
DP_FIB_PROBE	FIB Probe Current		0- 30	As above
DP_FIB_IMAGING	FIB Imaging		0 1 2 3 4 5 6 7	SEM FIB Milling SEM + FIB Mill + SEM Drift SEM Drift FIB None

			8 9 10 11 12 13 14	Ext Ext + SEM Wait Argon Argon From Argon Argon + SEM SEM + FIB Spot
DP_FIB_GUN_STATE	FIB Gun State	RO	0 1 2 3 4 5 6 7 8 9	Off On Ramping Error Vac Not Ready Extractor Auto Heating Conditioning Filament Heating Gun cooling down
DP_FIB_BLANKED	FIB Blanked		0 1	No Yes

6.2.11 Drift Correction

DP_DC_SEM_STATE	Sem drift status		0 1 2 3 4 5	Not Available Busy No Reference Failed Success Ready
DP_DC_FIELD_SEARCH	Field Search		0 1	No Yes
DP_DC_USE_STAGE	Use Stage		0 1	No Yes
DP_DC_CANCELLED	DC Cancelled		0 1	No Yes
DP_DC_AUTO_BRIGHTNESS	Auto Brightness		0 1	No Yes

6.3 String Parameters

6.3.1 General

Mnemonic		**Description**
SV_USER_NAME	RO	Logged-on SmartSEM user e.g “SYSTEM”
SV_SERIAL_NUMBER	RO	Microscope serial number e.g. “1540XB-29-03”
SV_VERSION	RO	Software version e.g. “V05.02 23 Jun 2006”
SV_MAG_REF_TO	RO	Magnification defined in reference to e.g. “Polaroid 545”

6.3.2 Image Save

SV_FILE_NAME	RO	Last image saved-in filename
SV_USER_TEXT	RO	User text added to saved Tiff file
SV_SAMPLE_ID		Sample ID
SV_IMAGE_PATH	RO	Default image file location
SV_OPERATOR		A name you can give yourself

6.3.3 User defined

SV_USER_DEV_1		User-defined string
SV_USER_DEV_5		Up to five strings

6.4 Commands

6.4.1 Mode switching

Command Mnemonic	Description	Restriction
CMD_MODE_NORMAL	Go to Normal mode	
CMD_MODE_SPLIT	Go to Split Screen	Lic
CMD_MODE_REDUCED	Go to Reduced Raster Screen mode	
CMD_MODE_EMISSION	Go to Emission mode	

6.4.2 Gun

CMD_SHUTDOWN	Shutdown	
CMD_MANUAL	Beam On	
CMD_RUNUP	Gun On	
CMD_BEAM_ON	EHT On	
CMD_BEAM_OFF	Beam Off	
CMD_EHT_OFF	EHT Off	

6.4.3 Beam

CMD_AUTO_ALIGN_COARSE	Coarse Align	
CMD_AUTO_ALIGN_STANDARD	Standard Align	
CMD_AUTO_ALIGN_FINE	Fine Align	
CMD_AUTO_FOCUS_COARSE	Coarse Focus	
CMD_AUTO_FOCUS_FINE	Fine Focus	
CMD_AUTO_STIG	Auto Stig	
CMD_AUTO_SATURATION	Auto Saturate	
CMD_FOCUS_STIG	Auto Focus+Stig	
CMD_OPTIBEAM_OFF	Optibeam Off	
CMD_OPTIBEAM_LOW_PROBE	Low Probe	
CMD_OPTIBEAM_MEDIUM_PROBE	Medium Probe	
CMD_OPTIBEAM_HIGH_PROBE	High Probe	
CMD_OPTIBEAM_FIELD	Optibeam Field Mode	
CMD_OPTIBEAM_RESOLUTION	Optibeam Resolution	
CMD_DO_SEM_DRIFT_CORRN	Do SEM Drift Correction	

6.4.4 Scanning

CMD_SCANRATEUP	Scan +	
CMD_SCANRATEDOWN	Scan -	
CMD_FREEZE_ZONE	Freeze Zone	
CMD_UNFREEZE_ZONE	Unfreeze Zone	
CMD_TOGGLE_FREEZE	Toggle Freeze / Unfreeze	
CMD_FREEZE_ALL	Freeze All	
CMD_UNFREEZE_ALL	Unfreeze All	
CMD_FRAME_INT	Frame Integration	
CMD_FRAME_AVERAGE	Frame Average	

CMD_ABORT_AUTO	Cancel Auto	
CMD_AUTO_STIG	Auto Stig	
CMD_AUTO_FOCUS_COARSE	Coarse Focus	
CMD_AUTO_FOCUS_FINE	Fine Focus	
CMD_LINE_INT	Line Integration	
CMD_PIXNR	Pixel Noise reduction	
CMD_CONTINUOUS_AVG	Continuous Avg.	
CMD_SCANRATE0...CMD_SCANRATE15	Scan Speed 0 - 15	
CMD_SCANRATEUP	Scan +	
CMD_SCANRATEDOWN	Scan -	

6.4.5 Detectors / Camera

CMD_TAKE_PHOTO	Photo	

6.4.6 Stage

CMD_STAGE_INIT	Stage init.	Priv
CMD_STAGE_BACKLASH	Do Stage Backlash	
CMD_STAGE_ABORT	Stage stop	
CMD_UNDO_STAGE_GOTO	Undo Stage Goto	
CMD_STAGE_NEXT_FIELD	Stage: Next Field	
CMD_STAGE_PREVIOUS_FIELD	Stage: Prev Field	

6.4.7 Vacuum

CMD_PUMP	Pump	
CMD_VENT	Vent	Priv
CMD_START_COLUMN_PUMP	Start Column Pump	
CMD_STOP_COLUMN_PUMP	Stop Column Pump	
CMD_CLOSE_COL_CHAMBER_VALVE	Close Column Chamber Valve	
CMD_OPEN_COL_CHAMBER_VALVE	Open Column Chamber Valve	
CMD_GOTO_EP	Go To EP (extended pressure range)	
CMD_GOTO_HV	Go To HV (high vacuum)	
CMD_GOTO_VP	Go To VP (variable pressure)	
CMD_VENT_AIRLOCK	Vent Airlock	
CMD_PUMP_AIRLOCK	Pump Airlock	

6.4.8 FIB

CMD_FIB_GUN_ON	FIB Gun On	FIB
CMD_FIB_GUN_OFF	FIB Gun Off	FIB
CMD_FIB_START_MILLING	Start Milling	FIB
CMD_FIB_ABORT_MILLING	Cancel Milling	FIB
CMD_FIB_PAUSE_MILLING	Pause Milling	FIB
CMD_FIB_RESUME_MILLING	Resume Milling	FIB
CMD_FIB_SAVE_DRIFT_SEM	FIB Save Drift (SEM)	FIB

CMD_FIB_SAVE_DRIFT_FIB	FIB Save Drift (FIB)	FIB
CMD_FIB_EXIT_DRIFT	FIB Exit Drift	FIB
CMD_DO_FIB_DRIFT_CORRN	Do FIB Drift Corrn	FIB
CMD_FIB_DC_CREATE_REF	FIB Drift Create Ref	FIB
CMD_FIB_DC_CREATE_OPERATIONS	FIB DC Create Operations	FIB
CMD_FIB_NUDGE_LEFT	FIB Nudge Mill Left	FIB
CMD_FIB_NUDGE_RIGHT	FIB Nudge Mill Right	FIB
CMD_FIB_NUDGE_UP	FIB Nudge Mill Up	FIB
CMD_FIB_NUDGE_DOWN	FIB Nudge Mill Down	FIB
CMD_FIB_GUN_VALVE_OPEN	FIB Gun Valve Open	FIB
CMD_FIB_GUN_VALVE_CLOSE	FIB Gun Valve Close	FIB

6.4.9 FIB Mode Switching

CMD_FIB_MODE_SEM	FIB Mode SEM	FIB
CMD_FIB_MODE_FIB	FIB Mode FIB	FIB
CMD_FIB_MODE_MILL	FIB Mode Mill	FIB
CMD_FIB_MODE_SEM_FIB	FIB Mode SEM+FIB	FIB
CMD_FIB_MODE_MILL_SEM	FIB Mode Mill+SEM	FIB
CMD_FIB_MODE_EXT	FIB Mode Ext	FIB
CMD_FIB_MODE_EXT_SEM	FIB Mode Ext+SEM	FIB
CMD_FIB_MODE_ARGON_SEM	FIB Mode Argon + SEM	FIB
CMD_FIB_MODE_DRIFT_SEM	FIB Mode Drift SEM	FIB
CMD_FIB_MODE_DRIFT_FIB	FIB Mode Drift FIB	FIB
CMD_FIB_MODE_ARGON	FIB Mode Argon	FIB
CMD_FIB_MODE_DRIFT_SEM	FIB Mode Drift SEM	FIB
CMD_FIB_MODE_DRIFT_FIB	FIB Mode Drift FIB	FIB

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 (Remoting Interface only)
API_E_PARAMETER_IS_DISABLED	1032	The command or parameter is currently disabled (you cannot execute or set it. Remoting Interface only)

Remoting Errors, MUST be in order and the range defined below

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

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

Carl Zeiss NTS 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@nts.zeiss.com

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

Plus a worldwide network of authorised distributors

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

Printed in the UK

Enabling the Nano-Age World™