ZKTeco EU
Security and Time Management Solutions

Standalone ZK Teco EU Communication SDK 2.0

Documentation for C++ SDK Component and C# SDK

Documentation Ver. 0.5

José Geca
05/06/2013

Index

ZK Teco EU SDK 2.0
Standalone Communication SDK

1. About Standalone ZK Teco EU Communication SDK

Standalone communication ZK Teco EU SDK is an Application Program Interface (API) for communication of the standalone Time Attendance & Job Costing machines, it provides convenience to manage user information and fingerprint, download T&A record and configure machine and Access Control Option.

Main function is as follow:

1. Download T&A record.
2. Download, upload user information, card information and fingerprint.
3. Configure the Access Control machine option.
4. Manage Job Costing data and configuration.

2. SDK Installation

2.1 Folder and contents

a) C++ COM SDK Version

The C++ COM SDK Version has 8 files

zkeuemkeeper.dll → Main C++ COM SDK

DLL Dependency's:

zkeucommpro.dll

zkeucomms.dll

zkeurscagent.dll

zkeurscomm.dll

zkeutcpcomm.dll

zkeuusbcomm.dll

zkeuzkeuemsdk.dll

b) C# DLL SDK Version

The C# SDK Version had 2 files:

ZKEUEmKeeperNet.dll → Main C# SDK DLL

ZkFingerObj.dll → C++ DLL Dependency

2.2 How to install SDK

a) C++ COM SDK Version

Windows 32 bits:

1 - Copy all DLL files to 'system32' folder of windows system directory.

2 – Register 'zkeuemkeeper.dll' COM DLL, executing the 'install_dll_XP_32bit.bat'.

This bat executes the follow command:

regsvr32 /u /s c:\windows\system32\zkeuemkeeper.dll
regsvr32 c:\windows\system32\zkeuemkeeper.dll

Another way to register the COM DLL, is executing Windows run command:

Click start →run regsvr32 zkeuemkeeper.dll

The popup will appear to indicate that register successful.

Finally to use SDK on your project, you only need to import the COM SDK reference from the list of components, the component name is 'ZKEUEMKeeper 2.0 Control'.

Windows 64 bits:

1 - Copy all DLL files to 'SysWOW64' folder of windows system directory.

2 – Register 'zkemjckeeper.dll' COM DLL, executing the 'install_dll_W7_64bit.bat'.

This bat executes the follow command:

regsvr32 /u /s c:\windows\SysWOW64\zkeuemkeeper.dll
regsvr32 c:\windows\SysWOW64\zkeuemkeeper.dll

Another way to register the COM DLL, is executing Windows run command:

Click start →run regsvr32 zkeuemkeeper.dll

The popup will appear to indicate that register successful.

Finally to use the SDK on your project, you only need to import the COM SDK reference from the list of components, the component name is 'ZKEUEMKeeper 2.0 Control'.

b) C# DLL SDK Version

Fist, copy the C++ DLL Dependency to your Windows system folder.

If you use Windows 32 bits, copy 'ZkFingerObj.dll' to 'system32' folder of windows system directory.
If you use Windows 64 bits, copy 'ZkFingerObj.dll' to 'SysWOW64' folder of windows system directory.

Now, to use the SDK on your project, you only need to import the DLL ZKEUEmKeeperNet.dll reference to your project.

3. Professional functions

3.1 Short Key Functions

3.1.1 SSR_GetProfShortKey

[C++ COM SDK Protocol]

SSR_GetProfShortKey (LONG ShortKeyID, BSTR * ShortKeyFun, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetProfShortKey (int ShortKeyID, ref string ShortKeyFun)

[Purpose]

Get a specific function key.

[Parameter (IN)]

ShortKeyID → ID of the specified key to be obtained. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

ShortKeyFun → Function of the specified key.

The functions are as follows: 'Undefine', 'State Key', 'SMS Key', 'Event Code List', 'View Records', 'Show Balance' and 'Help Key'.

[Return Value]

Return True for success, otherwise False.

3.1.2 SSR_SetProfShortKey

[C++ COM SDK Protocol]

SSR_SetProfShortKey (LONG ShortKeyID, BSTR ShortKeyFun, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetProfShortKey (int ShortKeyID, string ShortKeyFun)

[Purpose]

Set a specific function key.

[Parameter (IN)]

ShortKeyID → ID of the specified key to be set. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

ShortKeyFun → Function of the specified shortcut key.

The functions are as follows: 'Undefined', 'State Key', 'SMS Key', 'EventCode List', 'Show Balance' and 'View Records'.

[Return Value]

Return True for success, otherwise False.

3.1.3 SSR_DeleteProfShortKey

[C++ COM SDK Protocol]

SSR_DeleteProfShortKey (LONG ShortKeyID, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_DeleteProfShortKey (int ShortKeyID)

[Purpose]

Reset a specific function key to 'Undefined'.

[Parameter (IN)]

ShortKeyID → ID of the specified key to be set. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

[Return Value]

Return True for success, otherwise False.

3.1.4 SSR_GetProfStateRule

[C++ COM SDK Protocol]

SSR_GetProfStateRule (LONG ShortKeyID, LONG FirstRule, LONG * StateRuleID, LONG * Weekday, BSTR * StartTime, BSTR * EndTime, LONG * StateCode, LONG * WorkCode, BSTR * StateName, LONG * SetDefault, LONG * AllowPunch, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetProfStateRule (int ShortKeyID, int FirstRule, ref int StateRuleID, ref int Weekday, ref string StartTime, ref string EndTime, ref int StateCode, ref int WorkCode, ref string StateName, ref int SetDefault, ref int AllowPunch)

[Purpose]

Get a specific State Rule info for a 'State Key' function key (main group and subgroup info).

[Parameter (IN)]

ShortKeyID → ID of the specified key to be obtained. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

FirstRule → If FirstRule=0 we get only the main group function key info. If FirstRule=1 we get the subgroup info of the specified main group function key (only 'State Key' have subgroup info).

[Return Value]

Return True for success, otherwise False.

3.1.5 SSR_SetProfStateRule

[C++ COM SDK Protocol]

SSR_SetProfStateRule (LONG ShortKeyID, LONG StateRuleID, LONG Weekday, BSTR StartTime, BSTR EndTime, LONG StateCode, LONG WorkCode, BSTR StateName, LONG SetDefault, LONG AllowPunch, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetProfStateRule (int ShortKeyID, int StateRuleID, int Weekday, string StartTime, string EndTime, int StateCode, int WorkCode, string StateName, int SetDefault, int AllowPunch)

[Purpose]

Set a specific State Rule subgroup info for a 'State Key' function key.

[Parameter (IN)]

ShortKeyID → ID of the specified 'State Key' to be modified. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

StateRuleID → If StateRuleID =0 we add a new subgroup info for a 'State Key'. If StateRuleID !=0 we modify a subgroup info of the 'State Key' (only 'State Key' have subgroup info).

Weekday → Day of the Week we have active the 'State Key'.

StartTime → Start Time of activated 'State Key' (Format: HH:MM).

EndTime → End Time of activated 'State Key' (Format: HH:MM).

StateCode → StateCode ID.

WorkCode →WorkCode ID.

StateName → State name of the 'State Key'.

SetDefault → 'State Key' by default or not (values 0 or 1).

AllowPunch → Allow Punch yes or no (values 0 or 1).

[Return Value]

Return True for success, otherwise False.

3.1.6 SSR_DelProfStateRule

[C++ COM SDK Protocol]

SSR_DelProfStateRule (LONG ShortKeyID, LONG StateRuleID, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_DelProfStateRule (int ShortKeyID, int StateRuleID)

[Purpose]

Delete a specific State Rule of a 'State Key' function.

[Parameter (IN)]

ShortKeyID → ID of the specified 'State Key' to be modified. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

StateRuleID → State Rule ID we want to remove from a 'State Key' (subgroup).

[Return Value]

Return True for success, otherwise False.

3.1.7 SSR_ClearProfShortKey

[C++ COM SDK Protocol]

SSR_ClearProfShortKey ()

[C# SDK Protocol]

SSR_ClearProfShortKey ()

[Purpose]

Reset all the function keys to 'Undefined'.

[Return Value]

Return True for success, otherwise False.

3.2 Balance Functions

3.2.1 SSR_GetConceptInfo

[C++ COM SDK Protocol]

SSR_GetConceptInfo (BSTR * AllConcepts, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetConceptInfo (ref string AllConcepts)

SSR_GetConceptInfo (ref string[] Concepts)

[Purpose]

Get global balance concept info list (maximum 20 concepts).

The concepts come separated with char '#' in 'AllConcepts' ref string.

[Return Value]

Return True for success, otherwise False.

3.2.2 SSR_SetConceptInfo

[C++ COM SDK Protocol]

SSR_SetConceptInfo (BSTR AllConcepts, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetConceptInfo (string AllConcepts)

SSR_SetConceptInfo (string[] Concepts)

[Purpose]

Set the global balance concept info list (maximum 20 concepts).

[Parameter (IN)]

AllConcepts → string array with all concepts separated with char '#'. Maximum 20 concepts, maximum length for each concept is 50 chars.

Example: "Concept1#Concept2#Concept3"

Concepts (Only on C# SDK) → String array of maximum 20 concepts. The maximum length for each concept is 50 chars.

[Return Value]

Return True for success, otherwise False.

3.2.3 SSR_GetUserBalance

[C++ COM SDK Protocol]

SSR_GetUserBalance (BSTR UserID, BSTR * Date, BSTR * UserBalance, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetUserBalance (string UserID, ref string Date, ref string UserBalance)

[Purpose]

Get the user balance.

[Parameter (IN)]

UserID → ID of the specified user.

[Parameter (OUT)]

Date → Date of this balance (is a string value, can be other thing).

UserBalance → The User Balance.

It's a string with the next format:

Time1:Times1, Time2:Times2, … , Time20:Times20

Where Time1 is the total minutes (maximum total minutes in one year), for global balance concept number 1, assigned to our user.
Times1 is the number of times, our user use the global balance concept number 1.

If the user don't have any of the total 20 global concepts, Time and Times for these concepts position in the string have value 0 (0:0).

[Return Value]

Return True for success, otherwise False.

3.2.4 SSR_SetUserBalance

[C++ COM SDK Protocol]

SSR_SetUserBalance (BSTR UserID, BSTR Date, BSTR UserBalance, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetUserBalance (string UserID, string Date, string UserBalance)

[Purpose]

Set the user balance.

[Parameter (IN)]

UserID → ID of the specified user.

Date → Date of this balance. It's a free string format with maximum 16 chars , can be other thing.

UserBalance → The User Balance.

It's a string with the next format:

Time1:Times1, Time2:Times2, … , Time20:Times20

Where Time1 is the total minutes (maximum total minutes in one year), for global balance concept number 1, assigned to our user.
Times1 is the number of times, our user use the global balance concept number 1.
If the user don't have any of the total 20 global concepts, Time and Times for these concepts position in the string have value 0 (0:0).

Example: If the user we want to set, only have concepts 1 and 5 of the maximum 20 global balance concepts, the input string have this format:

"10:1,0:0,0:0,0:0,16:2,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0,0:0"

[Return Value]

Return True for success, otherwise False.

3.2.5 SSR_DeleteUserBalance

[C++ COM SDK Protocol]

SSR_DeleteUserBalance (BSTR UserID, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_DeleteUserBalance (string UserID)

[Purpose]

Delete user balance.

[Parameter (IN)]

UserID → ID of the specified user.

[Return Value]

Return True for success, otherwise False.

4. Professional & Job Costing functions

4.1 Send Files Function

4.1.1 SSR_SendFile

[C++ COM SDK Protocol]

SSR_SendFile(LONG FileOption, BSTR FileName, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SendFile (int FileOption, string FileName)

[Purpose]

Send files to Professional or Job Costing Device (this files are a selected type of files supported by the Device in compatible or custom format).

[Parameter (IN)]

FileOption → File Option ID (determine the kind of file we want to upload into Device).

FileName → The path of the file we want to upload into Device.

[Usage]

FileOption = 1 and FileName = 'languages.txt' → Update the list of languages on the system in lang options menu.

FileOption = 2 and FileName = 'languagesEU.tgz' → Update language text pack in the system.

FileOption = 3 and FileName = 'languagesVEU.tgz' → Update language voice pack in the system.

FileOption = 4 → Update Flag language image, in lang options menu. The FileName have the initials in Upper Case for the country and the image is .jpg format. Example: To update Spanish Flag image set FileName = 'ES.jpg'.

FileOption = 5 → Update Background image. The FileName have the ID ('ad_' + ID) of the Background and the image is .jpg format. Example: To update Background ID = 5 set FileName = 'ad_5.jpg'.

FileOption = 6 → Update photo image associated with User ID. The FileName have the User ID and the image is .jpg format. Example: To update photo of UserID = 123, set FileName = '123.jpg'.

FileOption = 7 → Update image associated with WorkCode ID in WorkCode Menu. The FileName have the WorkCode ID and the image is .jpg format. Example: To update image for WorkCodeID = 20, set FileName = '20.jpg'.

IMPORTANT: WorkCode is only supported in Professional Firmware for Time Attendance Devices, in Job Costing Devices we don't support WorkCode. The function 'SSR_SendFile' is the same, so this FileOption ID it's only for your reference.

FileOption = 8 → Update image associated with Task ID in Task Menu. The FileName have the Task ID and the image is .jpg format. Example: To update image for TaskID = 215, set FileName = '215.jpg'.

IMPORTANT: Task is only supported in Job Costing Firmware, in Professional Devices we don't support Task. The function 'SSR_SendFile' is the same, so this FileOption ID it's only for your reference.

[Return Value]

Return True for success, otherwise False.

4.2 Barcode Functions

4.2.1 SSR_GetBarCode

[C++ COM SDK Protocol]

SSR_GetBarCode (LONG Length, BSTR * Name, BSTR * Part1, BSTR * Part2, BSTR * Part3, BSTR * Part4, BSTR * Part5, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetBarCode (int Length, ref string Name, ref string Part1, ref string Part2, ref string Part3, ref string Part4, ref string Part5)

[Purpose]

Get Barcode settings info for the current Barcode Length.

[Parameter (IN)]

Length → Barcode size length settings record. We have support from 5 to 49 possible Barcode size lengths (we can configure 45 different size Barcode types).

[Return Value]

Return True for success, otherwise False.

4.2.2 SSR_SetBarcode

[C++ COM SDK Protocol]

SSR_SetBarCode (LONG Length, BSTR Name, BSTR Part1, BSTR Part2, BSTR Part3, BSTR Part4, BSTR Part5, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetBarCode (int Length, string Name, string Part1, string Part2, string Part3, string Part4, string Part5)

[Purpose]

Set Barcode settings info for the current Barcode Length.

[Parameter (IN)]

Length → Barcode size length settings record. We have support from 5 to 49 possible Barcode size lengths (we can configure 45 different size Barcode types).

Name → Barcode settings description.

Part1 →Part1 of Barcode settings (Format: AABBCC. AA: int value for Part1 Type ID. BB: int value for Part1 Start position. CC: Part1 Length int value.)

Part2 → Part2 of Barcode settings (Format: AABBCC. AA: int value for Part2 Type ID. BB: int value for Part2 Start position. CC: Part2 Length int value.)

Part3 → Part3 of Barcode settings (Format: AABBCC. AA: int value for Part3 Type ID. BB: int value for Part3 Start position. CC: Part3 Length int value.)

Part4 → Part4 of Barcode settings (Format: AABBCC. AA: int value for Part4 Type ID. BB: int value for Part4 Start position. CC: Part4 Length int value.)

Part5 → Part5 of Barcode settings (Format: AABBCC. AA: int value for Part5 Type ID. BB: int value for Part5 Start position. CC: Part5 Length int value.)

[Usage]

We can edit from 5 to 49 possible Barcode size lengths.

In each Barcode setting we have 5 optional Part range values from total Barcode length.

The Part string value has the format: AABBCC.

AA is a 2 digits int value that represents Part Type ID.

Part Type ID:

00 – Undefined (value by default). If we have Undefined, the current Part Start position (BB) and Part Length (CC) are not defined (we had 00 value for each one).

01 – User ID

02 – Card Number ID

03 – Project ID

04 – Job Number ID

05 – Task ID

BB: is a 2 digits int value that represents the current Part Start position.

CC: is a 2 digits int value that represents the current Part Length int value.

We can distribute the total Barcode length for each of the 5 Parts we have.

In each Part we can set the Start position and the length for the current Part Type ID.

The set values on each Part only bellows to this Part, if we define more Parts, the set values for the other Parts must be free positions of the Barcode length.

[Return Value]

Return True for success, otherwise False.

4.2.3 SSR_DeleteBarCode

[C++ COM SDK Protocol]

SSR_DeleteBarCode (LONG Length, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_DeleteBarCode (int Length)

[Purpose]

Reset Barcode settings info for the current Barcode Length.

[Parameter (IN)]

Length → Barcode size length settings record.

[Return Value]

Return True for success, otherwise False.

4.2.4 SSR_ClearBarcode

[C++ COM SDK Protocol]

SSR_ClearBarCode (VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_ClearBarCode ()

[Purpose]

Reset all Barcode settings info for all Barcode Length.

[Return Value]

Return True for success, otherwise False.

4.3 System Options Functions

4.3.1 GetSysOption

[C++ COM SDK Protocol]

GetSysOption (LONG dwMachineNumber, BSTR Option, BSTR * Value, VARIANT_BOOL * pVal)

[C# SDK Protocol]

GetSysOption (int dwMachineNumber, string Option, ref string Value)

[Purpose]

Get the current parameter value from the Device for the imputed Option.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

Option → Parameter Name.

[Return Value]

Return True for success, otherwise False.

Parameter names for Professional and Job Costing Devices:

'~ZKFPVersion' – Get the current version of the ZK Fingerprint algorithm working on Device (Value: 9 or 10). This value is read only.

Parameter names only for Job Costing Devices:

'MustChoiceProject' – Control if it is needed to input Project in Job Menu (Value: 0 or 1).

'MustChoiceTask' – Control if it is needed to input Task in Job Menu (Value: 0 or 1).

'MustChoiceUnits' – Control if it is needed to input Units in Job Menu (Value: 0 or 1).

'MaxStartEndTime' – Maximum time a Job can be opened, if this time is finish, the Job is automatically closed (Value format: HHMM).

'CloseTimeType' – Set a specified time on End Job when we end the job next day with MaxStartEndTime value reached (Value: 0: Last Start Job time, 1: Maximum Time, 2: Current Time).

'JobCostType' – Job Cost imputing mode. There are three kind of modes： Automatic, Manual without control, Manual with control. If it is automatic, it is decided by the last mode of this user. Manual mode needs the short key to do this.

As default it is 0, it means automatic (Value: 0: Automatic, 1: Manual without control, 2: Manual with control).

'DefaultProject' – Project by default if MustChoiceProject=0 (Value: 0 to 99999999). We can set DefaultProject=0 if we don't want to use or create a Project in Device.

'DefaultTask' – Task by default if MustChoiceTask=0 (Value: 1 to 99999).

'VerifyJobNumber' – Flag to verify if the Job Number imputed on Job Menu exist or not , by default the Job Number is controlled (Vale: 0 or 1).

'VerifyRelJobNumberTask' – Flag to verify if the association Job Number - Task imputed on Job Menu exist or not, by default is controlled (Vale: 0 or 1). NOTE: If VerifyRelJobNumberTask is set to 1 and VerifyJobNumber is set to 0 (using SDK), we always verify if Job Number exists too.

'BarcodeJobCheck' – If the user is using Barcode to inputting values in the menu 'Start Job', this parameter control if this is needed to confirm the info. As a default it is 1 and it means Automatic, 0 means Manual (Value: 0 or 1).

'AutomaticStartJob' – If is set to 1, when we End a Job, automatic Start a new one (Value: 0 or 1). By default is not active.

'AlwaysShowState' – If is set to 1, always show defined State Key's on main menu (Value: 0 or 1).

[Return Value]

Return True for success, otherwise False.

4.3.2 SetSysOption

[C++ COM SDK Protocol]

SetSysOption (LONG dwMachineNumber, BSTR Option, BSTR Value, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SetSysOption (int dwMachineNumber, string Option, string Value)

[Purpose]

Set the current parameter value from the Device for the imputed Option.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

Option → Parameter Name.

[Return Value]

Return True for success, otherwise False.

Parameter names only for Job Costing Devices:

'MustChoiceProject' – Control if it is needed to input Project in Job Menu (Value: 0 or 1).

'MustChoiceTask' – Control if it is needed to input Task in Job Menu (Value: 0 or 1).

'MustChoiceUnits' – Control if it is needed to input Units in Job Menu (Value: 0 or 1).

'MaxStartEndTime' – Maximum time a Job can be opened, if this time is finish, the Job is automatically closed (Value format: HHMM).

'JobCostType' – Job Cost imputing mode. There are three kind of modes： Automatic, Manual without control, Manual with control. If it is automatic, it is decided by the last mode of this user. Manual mode needs the short key to do this.

As default it is 0, it means automatic (Value: 0: Automatic, 1: Manual without control, 2: Manual with control).

'DefaultProject' – Project by default if MustChoiceProject=0 (Value: 0 to 99999999). We can set DefaultProject=0 if we don't want to use or create a Project in Device.

'DefaultTask' – Task by default if MustChoiceTask=0 (Value: 1 to 99999).

'VerifyJobNumber' – Flag to verify if the Job Number imputed on Job Menu exist or not , by default the Job Number is controlled (Vale: 0 or 1).

'VerifyRelJobNumberTask' – Flag to verify if the association Job Number - Task imputed on Job Menu exist or not, by default is controlled (Vale: 0 or 1). NOTE: If VerifyRelJobNumberTask is set to 1 and VerifyJobNumber is set to 0 (using SDK), we always verify if Job Number exists too.

'BarcodeJobCheck' – If the user is using Barcode to inputting values in the menu 'Start Job', this parameter control if this is needed to confirm the info. As a default it is 1 and it means Automatic, 0 means Manual (Value: 0 or 1).

'AutomaticStartJob' – If is set to 1, when we End a Job, automatic Start a new one (Value: 0 or 1). By default is not active.

'AlwaysShowState' – If is set to 1, always show defined State Key's on main menu (Value: 0 or 1).

Value → Parameter Value we want to set (see Value options in previous Option → Parameter Name).

[Return Value]

Return True for success, otherwise False.

4.4 Pen Drive Setting Functions

4.4.1 SSR_GetPenDriveSetting

[C++ COM SDK Protocol]

SSR_GetPenDriveSetting (LONG * FileMode, BSTR * FileFix, LONG * DelAtt, LONG * Seplit, LONG * SeplitValue, BSTR * Field, BSTR * FieldFix, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetPenDriveSetting (ref int FileMode, ref string FileFix, ref int DelAtt, ref int Seplit, ref int SeplitValue, ref string Field, ref string FieldFix)

[Purpose]

Get the current settings for Pen Drive exportation log data.

[Parameter (OUT)]

FileMode → File Mode ID.

Values:

0 – Attlog

1 – Attlog & DateTime

2 – Attlog & Date & Dev. ID

3 – Attlog & DateTime & Dev. ID

4 – Fixed

5 – Fixed & DateTime

6 – Fixed & Date & Dev. ID

7 – Fixed & DateTime & Dev. ID

FileFix → Custom Fixed Text (1-16 char) if we use FileMode = 4 to 7 (Fixed mode).

DelAtt → Indicate if AttLog is deleted after download.

Values: 0 - No, 1 - Yes and 2 - Always ask me.

Seplit → Separator Char Type.

Values: 0 – Tab Char, 1 – Semicolon, 2 – Comma, 3 – Space and 4 – ASC Value.

SeplitValue →Custom ASC Value (1-255 char) if we use Seplit = 4.

Field → Field Info Setting Values. It's a string of 10 char values, char position 0 is ID for Field 1 and char position 10 is ID for Field 10.

These ID's are (char value):

'0' – Undefined

'1' – Fixed

'2' – UserID

'3' – DateTime

'4' – Dev. ID

'5' – Dev. Name

'6' – Att. Code

'7' – Att. Name

'8' – Ver. Type

'9' – Event ID

':' – Event Name

';' – Project

'<' – Job Number

'=' – Task

'>' – Units

'?' – Other Units

IMPORTANT: Char ID's for 'Project', 'Job Number', 'Task', 'Units' and 'Other Units' are only supported in Job Costing Devices.

FieldFix → Custom Fixed Text 2 (1-16 char), used in Field Info Settings.

4.4.2 SSR_SetPenDriveSetting

[C++ COM SDK Protocol]

SSR_SetPenDriveSetting (LONG FileMode, BSTR FileFix, LONG DelAtt, LONG Seplit, LONG SeplitValue, BSTR Field, BSTR FieldFix, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetPenDriveSetting (int FileMode, string FileFix, int DelAtt, int Seplit, int SeplitValue, string Field, string FieldFix)

[Purpose]

Set the current settings for Pen Drive exportation log data.

[Parameter (IN)]

FileMode → File Mode ID.

Values:

0 – Attlog

1 – Attlog & DateTime

2 – Attlog & Date & Dev. ID

3 – Attlog & DateTime & Dev. ID

4 – Fixed

5 – Fixed & DateTime

6 – Fixed & Date & Dev. ID

7 – Fixed & DateTime & Dev. ID

FileFix → Custom Fixed Text (1-16 char) if we use FileMode = 4 to 7 (Fixed mode).

DelAtt → Indicate if AttLog is deleted after download.

Values: 0 - No, 1 - Yes and 2 - Always ask me.

Seplit → Separator Char Type.

Values: 0 – Tab Char, 1 – Semicolon, 2 – Comma, 3 – Space and 4 – ASC Value.

SeplitValue →Custom ASC Value (1-255 char) if we use Seplit = 4.

Field → Field Info Setting Values. It's a string of 10 char values, char position 0 is ID for Field 1 and char position 10 is ID for Field 10.

These ID's are (char value):

'0' – Undefined

'1' – Fixed

'2' – UserID

'3' – DateTime

'4' – Dev. ID

'5' – Dev. Name

'6' – Att. Code

'7' – Att. Name

'8' – Ver. Type

'9' – Event ID

':' – Event Name

';' – Project

'<' – Job Number

'=' – Task

'>' – Units

'?' – Other Units

IMPORTANT: Char ID's for 'Project', 'Job Number', 'Task', 'Units' and 'Other Units' are only supported in Job Costing Devices.

FieldFix → Custom Fixed Text 2 (1-16 char), used in Field Info Settings.

4.4.3 SSR_ClearPenDriveSetting

[C++ COM SDK Protocol]

SSR_ClearPenDriveSetting (VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_ClearPenDriveSetting ()

[Purpose]

Reset all current settings for Pen Drive exportation log data.

[Return Value]

Return True for success, otherwise False.

4.5 Multi Verify Setting Functions

4.5.1 GetMultiVerifySetting (GetSysOption)

To Get Multi Verify Setting we use the function already described in **4.3.1 GetSysOption**, but using the options parameter 'VerifyStyle'.

[C++ COM SDK Protocol]

GetSysOption (LONG dwMachineNumber, BSTR Option, BSTR * Value, VARIANT_BOOL * pVal)

[C# SDK Protocol]

GetSysOption (int dwMachineNumber, string Option, ref string Value)

[Purpose]

Get the current parameter value from the Device for the imputed Option.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

Option → Parameter Name. In this case 'VerifyStyle' parameter.

'VerifyStyle' ID's in Professional and Job Costing Devices:

0 - "FP/(PIN&PW)/RF"

1 - "FP"

2 - "PIN"

3 - "PIN&PW"

4 - "RF"

5 - "FP/(PIN&PW)"

6 - "FP/RF"

7 - "(PIN&PW)/RF"

8 - "PIN&FP"

9 - "FP&(PIN&PW)"

10 - "FP&RF"

11 - "(PIN&PW)&RF"

12 - "FP&PW&RF",

13 - "PIN&PW&FP",

14 - "FP&(RF/PIN)",

In Job Costing Devices we have one more ID:

15 - "FP/PIN/RF"

[Return Value]

Return True for success, otherwise False.

4.5.2 SetMultiVerifySetting (GetSysOption)

To Set Multi Verify Setting we use the function already described in **4.3.2 SetSysOption**, but using the options parameter 'VerifyStyle'.

[C++ COM SDK Protocol]

SetSysOption (LONG dwMachineNumber, BSTR Option, BSTR Value, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SetSysOption (int dwMachineNumber, string Option, string Value)

[Purpose]

Set the current parameter value from the Device for the imputed Option.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

Option → Parameter Name. In this case 'VerifyStyle' parameter.

'VerifyStyle' ID's in Professional and Job Costing Devices:

0 - "FP/(PIN&PW)/RF"

1 - "FP"

2 - "PIN"

3 - "PIN&PW"

4 - "RF"

5 - "FP/(PIN&PW)"

6 - "FP/RF"

7 - "(PIN&PW)/RF"

8 - "PIN&FP"

9 - "FP&(PIN&PW)"

10 - "FP&RF"

11 - "(PIN&PW)&RF"

12 - "FP&PW&RF",

13 - "PIN&PW&FP",

14 - "FP&(RF/PIN)",

In Job Costing Devices we have one more ID:

15 - "FP/PIN/RF"

[Return Value]

Return True for success, otherwise False.

4.5.3 SetPersonalVerifySetting (SetBatchUserInfoEx)

[C++ COM SDK Protocol]

SetBatchUserInfoEx (LONG dwMachineNumber, LONG dwFromEnrollNumber, LONG dwToEnrollNumber, LONG VerifyStyle, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SetBatchUserInfoEx (int dwMachineNumber, int dwFromEnrollNumber, int dwToEnrollNumber, int VerifyStyle)

[Purpose]

Set the current VerifyStyle value for a range of Users (the Users ID's must exist).

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

dwFromEnrollNumber → Start User ID (values between 1 and 999999999).

dwToEnrollNumber → End User ID (values between 1 and 999999999).

VerifyStyle → VerifyStyle ID.

'VerifyStyle' ID's in Professional and Job Costing Devices:

0 - "FP/(PIN&PW)/RF"

1 - "FP"

2 - "PIN"

3 - "PIN&PW"

4 - "RF"

5 - "FP/(PIN&PW)"

6 - "FP/RF"

7 - "(PIN&PW)/RF"

8 - "PIN&FP"

9 - "FP&(PIN&PW)"

10 - "FP&RF"

11 - "(PIN&PW)&RF"

12 - "FP&PW&RF",

13 - "PIN&PW&FP",

14 - "FP&(RF/PIN)",

In Job Costing Devices we have one more ID:

15 - "FP/PIN/RF"

[Return Value]

Return True for success, otherwise False.

5. Job Costing functions

5.1 Project Functions

5.1.1 ReadAllProjectID_JC

[C++ COM SDK Protocol]

ReadAllProjectID_JC (VARIANT_BOOL *pVal)

[C# SDK Protocol]

ReadAllProjectID_JC ()

[Purpose]

Read in all Project information to the memory of PC, Include Project ID, Description, Start Date and Estimated Finish Date.

[Return Value]

Return True for success, otherwise False.

5.1.2 SSR_GetAllProjectInfo_JC

[C++ COM SDK Protocol]

SSR_GetAllProjectInfo_JC (BSTR* ProjectCode, BSTR* ProjectName, BSTR* StartDate, BSTR* FinishDate, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetAllProjectInfo_JC (ref string ProjectCode, ref string ProjectName, ref string StartDate, ref string FinishDate)

[Purpose]

Obtain the Project information (from the memory of PC), every time this function execute a time, the Project information pointer which the point the memory move to the next record, when complete to read all Project information, returns to False.

IMPORTANT: To use this function you need to call first the function 'ReadAllProjectID_JC', to read all Project information from the Device to the memory of PC.

[Return Value]

Return True for success, otherwise False.

5.1.3 SSR_GetProject_JC

[C++ COM SDK Protocol]

SSR_GetProject_JC (BSTR ProjectCode, BSTR *ProjectName, BSTR *StartDate, BSTR *FinishDate, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetProject_JC (string ProjectCode, ref string ProjectName, ref string StartDate, ref string FinishDate)

[Purpose]

Get Project information.

[Parameters (IN)]

ProjectCode → project ID (values between 1 and 99999999).

[Return Value]

Return True for success, otherwise False.

5.1.4 SSR_SetProject_JC

[C++ COM SDK Protocol]

SSR_SetProject_JC (BSTR ProjectCode, BSTR ProjectName, BSTR StartDate, BSTR FinishDate, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_SetProject_JC (string ProjectCode, string ProjectName, string StartDate, string FinishDate)

[Purpose]

Set Project information.

[Parameter (IN)]

ProjectCode → Project ID (values between 1 and 99999999).

ProjectName → Project description (max. length 40 characters).

StartDate → Project start date (format: DD-MM-YYYY).

FinishDate → Project estimated finish date (format: DD-MM-YYYY).

[Return Value]

Return True for success, otherwise False.

5.1.5 SSR_DeleteProject_JC

[C++ COM SDK Protocol]

SSR_DeleteProject_JC (BSTR ProjectCode, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_DeleteProject_JC (string ProjectCode)

[Purpose]

Delete Project record, and if exist, all the relations between this Project and Job Numbers.

[Parameter (IN)]

ProjectCode → Project ID (values between 1 and 99999999).

[Return Value]

Return True for success, otherwise False.

5.1.6 SSR_ClearAllProjects_JC

[C++ COM SDK Protocol]

SSR_ClearAllProjects_JC (VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_ClearAllProjects_JC ()

[Purpose]

Clear all existing Projects and all 'Project – Job Number' relations.

[Return Value]

Return True for success, otherwise False.

5.2 Job Number Functions

5.2.1 ReadAllJobNumberID_JC

[C++ COM SDK Protocol]

ReadAllJobNumberID_JC (VARIANT_BOOL *pVal)

[C# SDK Protocol]

ReadAllJobNumberID_JC ()

[Purpose]

Read in all Job Number information to the memory of PC, Include Job Number ID, Description, Units, Start Date and Estimated Finish Date.

[Return Value]

Return True for success, otherwise False.

5.2.2 SSR_GetAllJobNumberInfo_JC

[C++ COM SDK Protocol]

SSR_GetAllJobNumberInfo_JC (BSTR* JobNumberCode, BSTR* JobNumberName, int* Units, BSTR* StartDate, BSTR* FinishDate, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetAllJobNumberInfo_JC (ref string JobNumberCode, ref string JobNumberName, ref int Units, ref string StartDate, ref string FinishDate)

[Purpose]

Obtain the Job Number information (from the memory of PC), every time this function execute a time, the Job Number information pointer which the point the memory move to the next record, when complete to read all Job Number information, returns to False.

IMPORTANT: To use this function you need to call first the function 'ReadAllJobNumberID_JC', to read all Job Number information from the Device to the memory of PC.

[Return Value]

Return True for success, otherwise False.

5.2.3 SSR_GetJobNumber_JC

[C++ COM SDK Protocol]

SSR_GetJobNumber_JC (BSTR JobNumberCode, BSTR *JobNumberName, int *Units, BSTR *StartDate, BSTR *FinishDate, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetJobNumber_JC (string JobNumberCode, ref string JobNumberName, ref int Units, ref string StartDate, ref string FinishDate)

[Purpose]

Get Job Number information.

[Parameter (IN)]

JobNumberCode → Job Number ID (values between 1 and 999999999999).

[Return Value]

Return True for success, otherwise False.

5.2.4 SSR_SetJobNumber_JC

[C++ COM SDK Protocol]

SSR_SetJobNumber_JC (BSTR JobNumberCode, BSTR JobNumberName, int Units, BSTR StartDate, BSTR FinishDate, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_SetJobNumber_JC (string JobNumberCode, string JobNumberName, string Units, string StartDate, string FinishDate)

[Purpose]

Set Job Number information.

[Parameter (IN)]

JobNumberCode → Job Number ID (values between 1 and 999999999999).

JobNumberName → Job Number description (max. length 40 characters).

Units → Job Number Units (values between 0 and 9999999).

StartDate → Job Number start date (format: DD-MM-YYYY).

FinishDate → Job Number estimated finish date (format: DD-MM-YYYY).

[Return Value]

Return True for success, otherwise False.

5.2.5 SSR_DeleteJobNumber_JC

[C++ COM SDK Protocol]

SSR_DeleteJobNumber_JC (BSTR JobNumberCode, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_DeleteJobNumber_JC (string JobNumberCode)

[Purpose]

Delete Job Number record, and if exist, all the relations between this Job Number and Projects and all relations between this Job Number and Tasks.

[Parameter (IN)]

JobNumberCode → Job Number ID (values between 1 and 999999999999).

[Return Value]

Return True for success, otherwise False.

5.2.6 SSR_ClearAllJobNumbers_JC

[C++ COM SDK Protocol]

SSR_ClearAllJobNumbers_JC (VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_ClearAllJobNumbers_JC ()

[Purpose]

Clear all existing Job Number, all 'Project – Job Number' relations and all 'Job Number – Task' relations.

[Return Value]

Return True for success, otherwise False.

5.3 Task Functions

5.3.1 ReadAllTaskID_JC

[C++ COM SDK Protocol]

ReadAllTaskID_JC (VARIANT_BOOL *pVal)

[C# SDK Protocol]

ReadAllTaskID_JC ()

[Purpose]

Read in all Task information to the memory of PC, Include Task ID, Description and Standard Time.

[Return Value]

Return True for success, otherwise False.

5.3.2 SSR_GetAllTaskInfo_JC

[C++ COM SDK Protocol]

SSR_GetAllTaskInfo_JC (int* TaskCode, BSTR* TaskName, BSTR* StdTime, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetAllTaskInfo_JC (ref int TaskCode, ref string TaskName, ref string StdTime)

[Purpose]

Obtain the Task information (from the memory of PC), every time this function execute a time, the Task information pointer which the point the memory move to the next record, when complete to read all Task information, returns to False.

IMPORTANT: To use this function you need to call first the function 'ReadAllTaskID_JC', to read all Task information from the Device to the memory of PC.

[Return Value]

Return True for success, otherwise False.

5.3.3 SSR_GetTask_JC

[C++ COM SDK Protocol]

SSR_GetTask_JC (int TaskCode, BSTR *TaskName, BSTR *StdTime, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetTask_JC (int TaskCode, ref string TaskName, ref string StdTime)

[Purpose]

Get Task information.

[Parameter (IN)]

TaskCode → Task ID (values between 1 and 99999).

[Return Value]

Return True for success, otherwise False.

5.3.4 SSR_SetTask_JC

[C++ COM SDK Protocol]

SSR_SetTask_JC (int TaskCode, BSTR TaskName, BSTR StdTime, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_SetTask_JC (int TaskCode, string TaskName, string StdTime)

[Purpose]

Set Task information.

[Parameter (IN)]

TaskCode → Task ID (values between 1 and 99999).

TaskName → Task description (max. length 30 characters).

StdTime → Task Standard Time (format: HHMMSS. Hours from 0 to 99, Minutes and Seconds from 0 to 59).

[Return Value]

Return True for success, otherwise False.

5.3.5 SSR_DeleteTask_JC

[C++ COM SDK Protocol]

SSR_DeleteTask_JC (int TaskCode, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_DeleteTask_JC (int TaskCode)

[Purpose]

Delete Task record, and if exist, all the relations between this Task and Job Numbers.

[Parameter (IN)]

TaskCode → Task ID (values between 1 and 99999).

[Return Value]

Return True for success, otherwise False.

5.3.6 SSR_ClearAllTasks_JC

[C++ COM SDK Protocol]

SSR_ClearAllTasks_JC (VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_ClearAllTasks_JC ()

[Purpose]

Clear all existing Tasks and all 'Job Number - Task' relations.

[Return Value]

Return True for success, otherwise False.

5.4 Relation 'Project – Job Number' Functions

5.4.1 ReadAllRelProjectJobNumberID_JC

[C++ COM SDK Protocol]

ReadAllRelProjectJobNumberID_JC (VARIANT_BOOL *pVal)

[C# SDK Protocol]

ReadAllRelProjectJobNumberID_JC ()

[Purpose]

Read in all relation 'Project – Job Number' information to the memory of PC, Include Project ID and Job Number ID.

[Return Value]

Return True for success, otherwise False.

5.4.2 SSR_GetAllRelProjectJobNumberInfo_JC

[C++ COM SDK Protocol]

GetAllRelProjectJobNumberInfo (BSTR * ProjectCode, BSTR * JobNumberCode, VARIANT_BOOL * pVal);

[C# SDK Protocol]

SSR_GetAllRelProjectJobNumberInfo_JC (ref string ProjectCode, ref string JobNumberCode)

[Purpose]

Obtain the relation 'Project – Job Number' information (from the memory of PC), every time this function execute a time, the relation 'Project – Job Number' information pointer which the point the memory move to the next record, when complete to read all relation 'Project – Job Number' information, returns to False.

IMPORTANT: To use this function you need to call first the function 'ReadAllRelProjectJobNumberID_JC', to read all relation 'Project – Job Number' information from the Device to the memory of PC.

[Return Value]

Return True for success, otherwise False.

5.4.3 SSR_GetJobNumberFromProject_JC

[C++ COM SDK Protocol]

SSR_GetJobNumberFromProject_JC (BSTR ProjectCode, BSTR JobNumberCode, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetJobNumberFromProject_JC (string ProjectCode, string JobNumberCode)

[Purpose]

Get relation 'Project – Job Number' information (we only have ID's on this relation, propose is only to check if the relation exists).

[Parameter (IN)]

ProjectCode → Project ID (values between 1 and 99999999).

JobNumberCode → Job Number ID (values between 1 and 999999999999).

[Return Value]

Return True for success, otherwise False.

5.4.4 SSR_SetJobNumberFromProject_JC

[C++ COM SDK Protocol]

SSR_SetJobNumberFromProject_JC (BSTR ProjectCode, BSTR JobNumberCode, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetJobNumberFromProject_JC (string ProjectCode, string JobNumberCode)

[Purpose]

Set relation 'Project – Job Number' information (we only have ID's on this relation, propose is only to set the relation ID's).

[Parameter (IN)]

ProjectCode → Project ID (values between 1 and 99999999).

JobNumberCode → Job Number ID (values between 1 and 999999999999).

[Return Value]

Return True for success, otherwise False.

5.4.5 SSR_DelJobNumberFromProject_JC

[C++ COM SDK Protocol]

SSR_DelJobNumberFromProject_JC (BSTR ProjectCode, BSTR JobNumbCode, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_DelJobNumberFromProject_JC (string ProjectCode, string JobNumbCode)

[Purpose]

Delete relation 'Project – Job Number' record.

[Parameter (IN)]

ProjectCode → Project ID (values between 1 and 99999999).

JobNumberCode → Job Number ID (values between 1 and 999999999999).

[Return Value]

Return True for success, otherwise False.

5.4.6 SSR_ClearAllRelProjectJobNumber_JC

[C++ COM SDK Protocol]

SSR_ClearAllRelProjectJobNumber_JC (VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_ClearAllRelProjectJobNumber_JC ()

[Purpose]

Clear all 'Project - Job Number' relations.

[Return Value]

Return True for success, otherwise False.

5.5 Relation 'Job Number - Task' Functions

5.5.1 ReadAllRelJobNumberTaskID_JC

[C++ COM SDK Protocol]

ReadAllRelJobNumberTaskID_JC (VARIANT_BOOL *pVal)

[C# SDK Protocol]

ReadAllRelJobNumberTaskID_JC ()

[Purpose]

Read in all relation 'Job Number - Task' information to the memory of PC, Include Job Number ID, Task ID, Task Description relation and Task Standard Time relation.

[Return Value]

Return True for success, otherwise False.

5.5.2 SSR_GetAllRelJobNumberTaskInfo_JC

[C++ COM SDK Protocol]

SSR_GetAllRelJobNumberTaskInfo_JC (BSTR* JobNumberCode, int* TaskCode, BSTR* TaskName, BSTR* TaskStdTime, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetAllRelJobNumberTaskInfo_JC (ref string JobNumberCode, ref int TaskCode, ref string TaskName, ref string TaskStdTime)

[Purpose]

Obtain the relation 'Job Number - Task' information (from the memory of PC), every time this function execute a time, the relation 'Job Number - Task' information pointer which the point the memory move to the next record, when complete to read all relation 'Job Number - Task' information, returns to False.

IMPORTANT: To use this function you need to call first the function 'ReadAllRelJobNumberTaskID_JC', to read all relation 'Job Number - Task' information from the Device to the memory of PC.

[Return Value]

Return True for success, otherwise False.

5.5.3 SSR_GetTaskFromJobNumber_JC

[C++ COM SDK Protocol]

SSR_GetTaskFromJobNumber_JC (long long int JobNumberCode, int TaskCode, BSTR *TaskName, BSTR *StdTime, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetTaskFromJobNumber_JC (Int64 JobNumberCode, int TaskCode, ref string TaskName, ref string StdTime)

[Purpose]

Get relation 'Job Number - Task' information.

[Parameter (IN)]

JobNumberCode → Job Number ID (values between 1 and 999999999999).

TaskCode → Task ID (values between 1 and 99999).

[Return Value]

Return True for success, otherwise False.

5.5.4 SSR_SetTaskFromJobNumber_JC

[C++ COM SDK Protocol]

SSR_SetTaskFromJobNumber_JC (BSTR JobNumberCode, int TaskCode, BSTR TaskName, BSTR StdTime, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetTaskFromJobNumber_JC (string JobNumberCode, int TaskCode, string TaskName, string StdTime)

[Purpose]

Set relation 'Job Number - Task' information.

[Parameter (IN)]

JobNumberCode → Job Number ID (values between 1 and 999999999999).

TaskCode → Task ID (values between 1 and 99999).

TaskName → Task description relation (max. length 30 characters).

StdTime → Task Standard Time relation (format: HHHHMMSS. Hours from 0 to 9999, Minutes and Seconds from 0 to 59).

[Return Value]

Return True for success, otherwise False.

5.5.5 SSR_DelTaskFromJobNumber_JC

[C++ COM SDK Protocol]

SSR_DelTaskFromJobNumber_JC (BSTR JobNumberCode, int TaskCode, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_DelTaskFromJobNumber_JC (string JobNumbCode, int TaskCode)

[Purpose]

Delete relation 'Job Number - Task' record.

[Parameter (IN)]

JobNumberCode → Job Number ID (values between 1 and 999999999999).

TaskCode → Task ID (values between 1 and 99999).

[Return Value]

Return True for success, otherwise False.

5.5.6 SSR_ClearAllRelJobNumberTask_JC

[C++ COM SDK Protocol]

SSR_ClearAllRelJobNumberTask_JC (VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_ClearAllRelJobNumberTask_JC ()

[Purpose]

Clear all 'Job Number - Task' relations.

[Return Value]

Return True for success, otherwise False.

5.6 Job Costing Log Functions

5.6.1 ReadGeneralLogData

[C++ COM SDK Protocol]

ReadGeneralLogData (long dwMachineNumber, VARIANT_BOOL *pVal);

[C# SDK Protocol]

ReadGeneralLogData (int dwMachineNumber);

[Purpose]

Read Job Costing records to the memory of PC, Include User ID, Verify Mode, InOutMode, Year, Month, Day, Hour, Minute, Second, Units, Project ID, Job Number ID and Task ID.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

[Return Value]

Return True for success, otherwise False.

5.6.2 SSR_GetGeneralLogData_JC

[C++ COM SDK Protocol]

SSR_GetGeneralLogData_JC (BSTR* dwEnrollNumber, LONG* dwVerifyMode, LONG* dwInOutMode, LONG* dwYear, LONG* dwMonth, LONG* dwDay, LONG* dwHour, LONG* dwMinute, LONG* dwSecond, LONG* dwUnits, LONG* dwOtherUnits, BSTR* dwProject, BSTR* dwJobNumber, LONG* dwTask, VARIANT_BOOL* pVal)

[C# SDK Protocol]

SSR_GetGeneralLogData_JC (out string dwEnrollNumber, out int dwVerifyMode, out int dwInOutMode, out int dwYear, out int dwMonth, out int dwDay, out ind dwHour, out int dwMinute, out int dwSecond, ref int dwUnits, ref int dwOtherUnits, ref string dwProject, ref string dwJobNumber, ref int dwTask);

[Purpose]

Obtain Job Costing attendance records (from the memory of PC), every time this function execute a time, the Job Costing attendance record information pointer which the point the memory move to the next record, when complete to read all Job Costing attendance record information, returns to False.

IMPORTANT: To use this function you need to call first the function 'ReadGeneralLogData', to read all Job Costing attendance record information from the Device to the memory of PC.

[Parameter (OUT)]

dwEnrollNumber → User ID.

dwVerifyMode → The value is the verification mode of the received Job Costing record.

The values are as follows:

0 – FingerPrint
1 – PIN
2 – RF (Card)
3 – PIN & Password
4 – PIN & FingerPrint
5 – FingerPrint & (PIN & Password)
6 – FingerPrint & RF (Card)
7 – (PIN & Password) & RF (Card)
8 – FingerPrint & Password & RF (Card)
9 – (PIN & Password) & FingerPrint
101 – Slave Device

dwInOutMode → The value is the AttState of the received Job Costing record.

The values are as follows:

10 – Start Job
20 – End Job
21 – End Job next day, ended with last Start Job time
22 – End Job next day, ended with Maximum time (MaxStartEndTime + Start Job Time)
23 – End Job next day, ended with Current time
30 – Reception Units (Units Produced)
40 – Pause Job
50 – Restart Job
60 – Check-IN
70 – Check-OUT

dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond →The values are the date and time of the received Job Costing record.

dwUnits → The value is the number of Units of the received Job Costing record.

dwOtherUnits → The value is the number of Other Units of the received Job Costing record.

dwProject → The value is the Project ID of the received Job Costing record.

dwJobNumber → The value is the Job Number ID of the received Job Costing record.

dwTask → The value is the Task ID of the received Job Costing record.

[Return Value]

Return True for success, otherwise False.

5.6.3 GetDeviceStatus

[C++ COM SDK Protocol]

GetDeviceStatus (long dwMachineNumber, long dwStatus, long * dwValue, VARIANT_BOOL *pVal);

[C# SDK Protocol]

GetDeviceStatus (int dwMachineNumber, int StatusItem, ref int Value);

[Purpose]

Obtain the data storage status of the Device, for example, number of current Job Number records, number of current users, number of administrators, etc.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

dwStatus (StatusItem) → Data to be obtained. The values range from 1 to 22.

The values are as follows:

1 - Number of administrators.
2 - Number of registered users.
3 - Number of fingerprint templates in the device.
4 - Number of passwords.
5 - Number of operation records.
6 - Number of Job Costing records.
7 - Fingerprint template capacity.
8 - User capacity.
9 - Attendance record capacity.
10 - Residual fingerprint template capacity.
11 - Residual user capacity.
12 - Residual attendance record capacity.
21 - Number of faces.
22 - Face capacity.

Return 0 in other cases.

[Return Value]

Return True for success, otherwise False.

5.6.4 ClearGLog

[C++ COM SDK Protocol]

ClearGLog (long dwMachineNumber, VARIANT_BOOL *pVal);

[C# SDK Protocol]

ClearGLog (int dwMachineNumber);

[Purpose]

Clear all Job Costing records from the Device.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

[Return Value]

Return True for success, otherwise False.

5.6.5 RefreshData

[C++ COM SDK Protocol]

RefreshData (long dwMachineNumber, VARIANT_BOOL *pVal);

[C# SDK Protocol]

RefreshData (int dwMachineNumber);

[Purpose]

Refresh the data in the Device. This function is usually called after user information or fingerprints are uploaded. In this way, the modifications take effect immediately.

[Parameter (IN)]

dwMachineNumber → Device number (not used, you can ignore the value).

[Return Value]

Return True for success, otherwise False.

5.7 Short Key Functions

5.7.1 SSR_GetJobCostShortKey

[C++ COM SDK Protocol]

SSR_GetJobCostShortKey (LONG ShortKeyID, BSTR * ShortKeyFun, BSTR * ShortKeyCustomName, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_GetJobCostShortKey (int ShortKeyID, ref string ShortKeyFun, ref ShortKeyCustomName)

[Purpose]

Get a specific function key.

[Parameter (IN)]

ShortKeyID → ID of the specified key to be obtained. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

ShortKeyFun → Function of the specified key.

The functions are as follows: 'Undefined', 'SMS Key', 'Start Job', 'Start Last Job', 'Pause Job', 'Restart Job', 'End Job', 'Reception Units', 'Check-IN' and 'Check-OUT' .

ShortKeyCustomName → Custom Name of the specified key (if exists, replace the default short key function name on main screen).

[Return Value]

Return True for success, otherwise False.

5.7.2 SSR_SetJobCostShortKey

[C++ COM SDK Protocol]

SSR_SetJobCostShortKey (LONG ShortKeyID, BSTR ShortKeyFun, BSTR ShortKeyCustomName, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_SetJobCostShortKey (int ShortKeyID, string ShortKeyFun, string ShortKeyCustomName)

[Purpose]

Set a specific function key.

[Parameter (IN)]

ShortKeyID → ID of the specified key to be set. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

ShortKeyFun → Function of the specified shortcut key.

The functions are as follows: 'Undefined', 'SMS Key', 'Start Job', 'Start Last Job', 'Pause Job', 'Restart Job', 'End Job', 'Reception Units', 'Check-IN' and 'Check-OUT' .

ShortKeyCustomName → Custom Name of the specified key (if exists, replace the default short key function name on main screen).

[Return Value]

Return True for success, otherwise False.

5.7.3 SSR_DeleteJobCostShortKey

[C++ COM SDK Protocol]

SSR_DeleteJobCostShortKey (LONG ShortKeyID, VARIANT_BOOL * pVal)

[C# SDK Protocol]

SSR_DeleteJobCostShortKey (int ShortKeyID)

[Purpose]

Reset a specific function key to 'Undefined' (and delete the Custom Name).

[Parameter (IN)]

ShortKeyID → ID of the specified key to be set. The mappings are as follows:

0: F1, 1: F2, 2: F3, 3: F4, 4: F5, 5: F6, 6: F7, 7: F8, 8: *, 9: #, 10: None, 11: Up, 12: Down, 13: Left, 14: Right

[Return Value]

Return True for success, otherwise False.

5.7.4 SSR_ClearJobCostShortKey

[C++ COM SDK Protocol]

SSR_ClearJobCostShortKey ()

[C# SDK Protocol]

SSR_ClearJobCostShortKey ()

[Purpose]

Reset all the function keys to 'Undefined' (and delete all Custom Names).

[Return Value]

Return True for success, otherwise False.

5.8 Real-Time Event Functions

5.8.1 OnJobCostingTransactionEx

[C++ COM SDK Protocol]

OnJobCostingTransactionExEventHandler (BSTR EnrollNumber, LONG IsInvalid, LONG AttState, LONG VerifyMethod, LONG Year, LONG Month, LONG Day, LONG Hour, LONG Minute, LONG Second, BSTR Project, BSTR JobNumber, LONG Task, LONG Units, LONG OtherUnits)

[C# SDK Protocol]

OnJobCostingTransactionExEvent (uint EnrollNumber, int IsInvalid, int AttState, int VerifyMethod, int Year, int Month, int Day, int Hour, int Minute, int Second, string Project, string JobNumber, int Task, int Units, int OtherUnits)

[Purpose]

This event is triggered after verification succeeds. Get values from Job Costing Logs in real time.

[Return Value]

EnrollNumber → User ID.

IsInvalid → Indicate if the record is valid (User valid/invalid). 0: Valid, 1: Not Valid (If record is not valid we must ignore AttState value).

AttState → The value is the Attendance State (InOutMode) of the received Job Costing record.

The values are as follows:

10 – Start Job
20 – End Job
21 – End Job next day, ended with last Start Job time
22 – End Job next day, ended with Maximum time (MaxStartEndTime + Start Job Time)
23 – End Job next day, ended with Current time
30 – Reception Units (Units Produced)
40 – Pause Job
50 – Restart Job
60 – Check-IN
70 – Check-OUT

VerifyMethod → The value is the verification mode of the received Job Costing record.

The values are as follows:

0 – FingerPrint
1 – PIN
2 – RF (Card)
3 – PIN & Password
4 – PIN & FingerPrint
5 – FingerPrint & (PIN & Password)
6 – FingerPrint & RF (Card)
7 – (PIN & Password) & RF (Card)
8 – FingerPrint & Password & RF (Card)
9 – (PIN & Password) & FingerPrint
101 – Slave Device

Year, Month, Day, Hour, Minute, Second →The values are the date and time of the received Job Costing record.

Project → The value is the Project ID of the received Job Costing record.

JobNumber → The value is the Job Number ID of the received Job Costing record.

Task → The value is the Task ID of the received Job Costing record.

Units → The value is the number of Units of the received Job Costing record.

OtherUnits → The value is the number of Other Units of the received Job Costing record.

IMPORTANT: If you use C# SDK you cannot use more SDK functions directly inside the Event call function. By example, calling a SDK function to make consults using a returned value from Event.

To use SDK functions inside Event call function, you must use a Thread (with parameters if you need to use returned values from Event) and in the created Thread now you can use the SDK functions you need. If you use Windows Forms and want to set Event returned value in a Windows Forms control (like a text in a label), you must invoke a callback function.