**ZKTeco Standalone SDK Development Manual**

**Version: 2.2, Rev. A.3**

**Date: 2020.12.16**

**Copyright**

©ZKTECO CO., LTD. 2020.All rights reserved.  
This document may not be reproduced, disseminated or republished in any form without the prior written permission of ZKTECO.

**Release history**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Change** |
| 2018-01-01 | 2.0, Rev. A.1 | New release version |
| 2018-04-15 | 2.1, Rev. A.2 | Distribution group of International |
| 2020-12-16 | 2.2, Rev. A.3 | Add 5.2.1.13, 5.2.14.9, 5.2.20,5.5.4.14  Contains about user visible face photo manager, mask status and temperature check functions . |

Contents

[1 SDK Description 13](#_Toc6905)

[2 Quick Start 13](#_Toc7196)

[2.1 Terms 13](#_Toc11006)

[2.2 Common Processes 14](#_Toc5160)

[2.2.1 Downloading Attendance Records 15](#_Toc28976)

[2.2.2 Downloading Operation Records 15](#_Toc17073)

[2.2.3 Setting Access Control 16](#_Toc15819)

[2.2.4 Downloading User Information, or Fingerprint Templates 16](#_Toc16726)

[2.2.5 Receiving Real-time Events 18](#_Toc19741)

[2.2.6 Enrolling Users Online (Uploading Information,and Fingerprint Templates of Users) 18](#_Toc18372)

[2.2.7 Uploading Short Messages 19](#_Toc6731)

[3 Related Attributes 20](#_Toc17909)

[3.1 AccGroup 20](#_Toc22824)

[3.2 AccTimeZones 20](#_Toc7464)

[3.3 BASE64 20](#_Toc22476)

[3.4 CardNumber 20](#_Toc21922)

[3.5 CommPort 21](#_Toc19916)

[3.6 ConvertBIG5 21](#_Toc24448)

[3.7 PINWidth 21](#_Toc3645)

[3.8 GetStrCardNumber 22](#_Toc7975)

[3.9 SetStrCardNumber 22](#_Toc31479)

[3.10 IsNewFirmwareMachine 23](#_Toc18957)

[3.11 GetDeviceFirmwareVersion 23](#_Toc19112)

[4 Real-time Event Functions 24](#_Toc32497)

[4.1 Obtaining Real-Time Events 24](#_Toc180)

[4.1.1 RegEvent 24](#_Toc7364)

[4.1.2 ReadRTLog 25](#_Toc11517)

[4.1.3 GetRTLog 26](#_Toc13765)

[4.2 Real-Time Events 27](#_Toc31495)

[4.2.1 OnConnected 27](#_Toc10404)

[4.2.2 OnDisConnected 27](#_Toc4881)

[4.2.3 OnAlarm 28](#_Toc30638)

[4.2.4 OnDoor 28](#_Toc9496)

[4.2.5 OnAttTransaction 29](#_Toc30278)

[4.2.6 OnAttTransactionEx 30](#_Toc29926)

[4.2.7 OnEnrollFinger 32](#_Toc26426)

[4.2.8 OnEnrollFingerEx 33](#_Toc14228)

[4.2.9 OnDeleteTemplate 33](#_Toc15030)

[4.2.10 OnFinger 34](#_Toc22061)

[4.2.11 OnFingerFeature 34](#_Toc20709)

[4.2.12 OnHIDNum 35](#_Toc1617)

[4.2.13 OnKeyPress 35](#_Toc3713)

[4.2.14 OnNewUser 36](#_Toc28937)

[4.2.15 OnVerify 36](#_Toc15062)

[4.2.16 OnWriteCard 37](#_Toc8497)

[4.2.17 OnEmptyCard 37](#_Toc6927)

[4.2.18 OnEMData 38](#_Toc32278)

[5 Common Functions 38](#_Toc7369)

[5.1 Device Connection Functions 38](#_Toc31816)

[5.1.1 Connect\_Net 38](#_Toc12851)

[5.1.2 Connect\_Com 39](#_Toc15786)

[5.1.3 Connect\_USB 40](#_Toc21581)

[5.1.4 Disconnect 41](#_Toc25409)

[5.2 Data Management Functions 41](#_Toc1025)

[5.2.1 Attendance Record Data 41](#_Toc9819)

[5.2.1.1 ReadGeneralLogData 41](#_Toc10543)

[5.2.1.2 ReadAllGLogData 42](#_Toc15357)

[5.2.1.3 ReadTimeGLogData 42](#_Toc26336)

[5.2.1.4 ReadNewGLogData 43](#_Toc23745)

[5.2.1.5 GetGeneralLogData 44](#_Toc30262)

[5.2.1.6 GetAllGLogData 47](#_Toc23678)

[5.2.1.7 GetGeneralLogDataStr 49](#_Toc26080)

[5.2.1.8 GetGeneralExtLogData 50](#_Toc22653)

[5.2.1.9 SSR\_GetGeneralLogData 52](#_Toc18960)

[5.2.1.10 ClearGLog 55](#_Toc30724)

[5.2.1.11 DeleteAttlogBetweenTheDate 55](#_Toc2381)

[5.2.1.12 DeleteAttlogByTime 56](#_Toc23161)

[5.2.1.13 SSR\_GetGeneralLogDataWithMask 57](#_Toc5261)

[5.2.2 Operation Record Data 59](#_Toc26485)

[5.2.2.1 ReadSuperLogData 59](#_Toc3460)

[5.2.2.2 ReadAllSLogData 60](#_Toc6258)

[5.2.2.3 GetSuperLogData 61](#_Toc13323)

[5.2.2.4 GetAllSLogData 63](#_Toc20738)

[5.2.2.5 ClearSLog 65](#_Toc31749)

[5.2.2.6 GetSuperLogData2 66](#_Toc16820)

[5.2.3 User Information Functions 68](#_Toc30514)

[5.2.3.1 ReadAllUserID 68](#_Toc29448)

[5.2.3.2 EnableUser 69](#_Toc23242)

[5.2.3.3 SSR\_EnableUser 70](#_Toc16677)

[5.2.3.4 SetUserInfoEx 71](#_Toc26337)

[5.2.3.5 GetAllUserID 72](#_Toc5684)

[5.2.3.6 GetAllUserInfo 73](#_Toc26071)

[5.2.3.7 GetUserInfoEx 74](#_Toc24709)

[5.2.3.8 DeleteUserInfoEx 75](#_Toc27751)

[5.2.3.9 SSR\_GetAllUserInfo 75](#_Toc27371)

[5.2.3.10 GetUserInfo 76](#_Toc1811)

[5.2.3.11 GetUserInfoByPIN2 78](#_Toc12532)

[5.2.3.12 GetUserInfoByCard 78](#_Toc10231)

[5.2.3.13 GetUserIDByPIN2 79](#_Toc21993)

[5.2.3.14 GetPIN2 80](#_Toc86)

[5.2.3.15 SSR\_GetUserInfo 81](#_Toc23733)

[5.2.3.16 SetUserInfo 82](#_Toc2902)

[5.2.3.17 SSR\_SetUserInfo 83](#_Toc28712)

[5.2.3.18 ModifyPrivilege 84](#_Toc31683)

[5.2.4 Registration Data Functions (Including Both User Information and Fingerprint) 85](#_Toc17969)

[5.2.4.1 GetEnrollData 85](#_Toc3287)

[5.2.4.2 SetEnrollData 86](#_Toc15875)

[5.2.4.3 DeleteEnrollData 87](#_Toc30296)

[5.2.4.4 SSR\_DeleteEnrollData 88](#_Toc6207)

[5.2.4.5 SSR\_DeleteEnrollDataExt 89](#_Toc1673)

[5.2.4.6 GetEnrollDataStr 90](#_Toc24987)

[5.2.4.7 SetEnrollDataStr 91](#_Toc14531)

[5.2.5 Fingerprint Template Functions 93](#_Toc11170)

[5.2.5.1 ReadAllTemplate 93](#_Toc5551)

[5.2.5.2 SSR\_GetUserTmp 93](#_Toc19578)

[5.2.5.3 SSR\_GetUserTmpStr 94](#_Toc21111)

[5.2.5.4 SSR\_SetUserTmp 95](#_Toc26067)

[5.2.5.5 SSR\_SetUserTmpStr 96](#_Toc3696)

[5.2.5.6 DelUserTmp 96](#_Toc13430)

[5.2.5.7 SSR\_DelUserTmp 97](#_Toc19200)

[5.2.5.8 SSR\_SetUserTmpExt 98](#_Toc29954)

[5.2.5.9 SSR\_DelUserTmpExt 99](#_Toc22480)

[5.2.5.10 SetUserTmp 99](#_Toc20440)

[5.2.5.11 SetUserTmpStr 100](#_Toc32147)

[5.2.5.12 SetUserTmpEx 101](#_Toc3415)

[5.2.5.13 SetUserTmpExStr 102](#_Toc17375)

[5.2.5.14 GetUserTmp 103](#_Toc7358)

[5.2.5.15 GetUserTmpStr 104](#_Toc21226)

[5.2.5.16 GetUserTmpEx 105](#_Toc2941)

[5.2.5.17 GetUserTmpExStr 106](#_Toc14384)

[5.2.5.18 GetFPTempLength 107](#_Toc14270)

[5.2.5.19 GetFPTempLengthStr 108](#_Toc19791)

[5.2.5.20 FPTempConvert 108](#_Toc8289)

[5.2.5.21 FPTempConvertStr 109](#_Toc14989)

[5.2.5.22 FPTempConvertNew 110](#_Toc1484)

[5.2.5.23 FPTempConvertNewStr 111](#_Toc6902)

[5.2.6 Face Template Functions 111](#_Toc27191)

[5.2.6.1 SetUserFace 111](#_Toc13801)

[5.2.6.2 GetUserFace 112](#_Toc13662)

[5.2.6.3 DelUserFace 113](#_Toc8847)

[5.2.6.4 GetUserFaceStr 114](#_Toc7871)

[5.2.6.5 SetUserFaceStr 115](#_Toc2966)

[5.2.7 User Verify Functions 116](#_Toc23246)

[5.2.7.1 SetUserVerifyStyle 116](#_Toc15900)

[5.2.7.2 GetUserVerifyStyle 117](#_Toc10950)

[5.2.8 Shortcut Keys Functions 118](#_Toc17593)

[5.2.8.1 SSR\_SetShortkey 118](#_Toc11263)

[5.2.8.2 SSR\_GetShortkey 120](#_Toc6144)

[5.2.8.3 EnableCustomizeAttState 121](#_Toc27375)

[5.2.8.4 SetCustomizeAttState 122](#_Toc30714)

[5.2.8.5 DelCustomizeAttState 123](#_Toc8003)

[5.2.8.6 GetAllSFIDName 124](#_Toc24153)

[5.2.8.7 GetShortkey 125](#_Toc25011)

[5.2.8.8 SetShortkey 126](#_Toc14208)

[5.2.9 Work Code Functions 128](#_Toc29052)

[5.2.9.1 SetWorkCode 128](#_Toc16528)

[5.2.9.2 GetWorkCode 129](#_Toc3751)

[5.2.9.3 SSR\_GetWorkCode 130](#_Toc6940)

[5.2.9.4 SSR\_SetWorkCode 130](#_Toc22402)

[5.2.9.5 SSR\_DeleteWorkCode 131](#_Toc27439)

[5.2.9.6 SSR\_ClearWorkCode 132](#_Toc19197)

[5.2.9.7 DeleteWorkCode 132](#_Toc15044)

[5.2.9.8 ClearWorkCode 133](#_Toc24261)

[5.2.9.9 SSR\_GetWorkCodeIDByName 133](#_Toc30194)

[5.2.10 SMS Functions 134](#_Toc20302)

[5.2.10.1 SetSMS 134](#_Toc11588)

[5.2.10.2 SetUserSMS 135](#_Toc16692)

[5.2.10.3 SSR\_SetUserSMS 136](#_Toc30752)

[5.2.10.4 GetSMS 137](#_Toc18071)

[5.2.10.5 DeleteSMS. 138](#_Toc24368)

[5.2.10.6 DeleteUserSMS 138](#_Toc12381)

[5.2.10.7 SSR\_DeleteUserSMS 139](#_Toc14107)

[5.2.10.8 ClearUserSMS 140](#_Toc4737)

[5.2.10.9 ClearSMS 141](#_Toc22045)

[5.2.11 Holiday Functions 141](#_Toc23509)

[5.2.11.1 SetHoliday 141](#_Toc9014)

[5.2.11.2 GetHoliday 142](#_Toc9445)

[5.2.11.3 SSR\_GetHoliday 143](#_Toc13328)

[5.2.11.4 SSR\_SetHoliday 144](#_Toc14042)

[5.2.12 DST Functions 144](#_Toc5832)

[5.2.12.1 SetDaylight 144](#_Toc4751)

[5.2.12.2 GetDaylight 145](#_Toc3895)

[5.2.13 System Data Management Functions 146](#_Toc16695)

[5.2.13.1 ClearKeeperData 146](#_Toc2943)

[5.2.13.2 ClearData 147](#_Toc17218)

[5.2.13.3 GetDataFile 148](#_Toc29143)

[5.2.13.4 SendFile 149](#_Toc22376)

[5.2.13.5 ReadFile 149](#_Toc14838)

[5.2.13.6 RefreshData 150](#_Toc25341)

[5.2.14 User Photo & Attendance Photo 151](#_Toc5766)

[5.2.14.1 UploadUserPhoto 151](#_Toc19967)

[5.2.14.2 DownloadUserPhoto 152](#_Toc22409)

[5.2.14.3 DeleteUserPhoto 153](#_Toc30845)

[5.2.14.4 GetAllUserPhoto 153](#_Toc13466)

[5.2.14.5 GetPhotoNamesByTime 154](#_Toc23622)

[5.2.14.6 GetPhotoByName 155](#_Toc26216)

[5.2.14.7 GetPhotoCount. 156](#_Toc17667)

[5.2.14.8 ClearPhotoByTime 157](#_Toc28897)

[5.2.14.9 GetPhotoByNameToFile 158](#_Toc6694)

[5.2.15 Bell Functions 158](#_Toc19434)

[5.2.15.1 GetBellSchDataEx 158](#_Toc20823)

[5.2.15.2 SetBellSchDataEx 160](#_Toc18067)

[5.2.15.3 GetDayBellSchCount 161](#_Toc5447)

[5.2.15.4 GetMaxBellIDInBellSchData 162](#_Toc8484)

[5.2.15.5 ReadAllBellSchData 163](#_Toc19469)

[5.2.15.6 GetEachBellInfo 163](#_Toc31876)

[5.2.16 UserValidDate Functions 165](#_Toc16761)

[5.2.16.1 SetUserValidDate 165](#_Toc29115)

[5.2.16.2 GetUserValidDate 166](#_Toc7431)

[5.2.17 Personalise Functions 167](#_Toc31390)

[5.2.17.1 UploadTheme 167](#_Toc18515)

[5.2.17.2 UploadPicture 168](#_Toc23798)

[5.2.17.3 DownloadPicture 169](#_Toc21051)

[5.2.18 APP Info Functions 170](#_Toc5007)

[5.2.18.1 GetAllAppFun 170](#_Toc31769)

[5.2.18.2 GetAllRole 170](#_Toc19735)

[5.2.18.3 GetAppOfRole 171](#_Toc18870)

[5.2.18.4 GetFunOfRole 172](#_Toc5823)

[5.2.18.5 SetPermOfAppFun 173](#_Toc29155)

[5.2.18.6 DeletePermOfAppFun 174](#_Toc14572)

[5.2.18.7 IsUserDefRoleEnable 174](#_Toc15585)

[5.2.19 Template Integration Functions 175](#_Toc16005)

[5.2.19.1 SSR\_SetDeviceData 175](#_Toc1788)

[5.2.19.2 SSR\_GetDeviceData 176](#_Toc9121)

[5.2.19.3 SSR\_GetDeviceDataCount 177](#_Toc4356)

[5.2.19.4 SSR\_DeleteDeviceData 178](#_Toc20122)

[5.2.19.5 Variable description : BiometricType 179](#_Toc21897)

[5.2.19.6 Variable description : BiometricVersion 179](#_Toc20184)

[5.2.19.7 Variable description : BiometricMaxCount 180](#_Toc4360)

[5.2.19.8 Variable description : BiometricUsedCount 180](#_Toc30414)

[5.2.20 LiveFace Integration Functions 180](#_Toc23659)

[5.2.20.1 GetUserFacePhotoNames 180](#_Toc21243)

[5.2.20.2 GetUserFacePhotoByName 181](#_Toc29825)

[5.2.20.3 GetUserFacePhotoCount 182](#_Toc19961)

[5.2.20.4 SendUserFacePhoto 183](#_Toc24166)

[5.3 Access Control Functions(Time Zone, Group, Open Door Combination) 183](#_Toc4561)

[5.3.1 GetUserGroup 183](#_Toc22508)

[5.3.2 SetUserGroup 184](#_Toc31029)

[5.3.3 GetTZInfo 185](#_Toc15490)

[5.3.4 SetTZInfo 186](#_Toc14160)

[5.3.5 GetUnlockGroups 187](#_Toc2612)

[5.3.6 SetUnlockGroups 188](#_Toc21322)

[5.3.7 SSR\_SetUnLockGroup 189](#_Toc20401)

[5.3.8 SSR\_GetUnLockGroup 190](#_Toc15539)

[5.3.9 GetGroupTZs 191](#_Toc21372)

[5.3.10 SetGroupTZs 192](#_Toc10346)

[5.3.11 GetGroupTZStr 193](#_Toc1459)

[5.3.12 SetGroupTZStr 194](#_Toc8652)

[5.3.13 SSR\_SetGroupTZ 194](#_Toc6424)

[5.3.14 SSR\_GetGroupTZ 195](#_Toc23071)

[5.3.15 GetUserTZs 197](#_Toc17719)

[5.3.16 SetUserTZs 198](#_Toc20634)

[5.3.17 GetUserTZStr 198](#_Toc732)

[5.3.18 SetUserTZStr 200](#_Toc32641)

[5.3.19 ACUnlock 201](#_Toc23875)

[5.3.20 GetACFun 201](#_Toc24350)

[5.3.21 GetDoorState 202](#_Toc1434)

[5.3.22 UseGroupTimeZone 203](#_Toc21932)

[5.3.23 TurnOffAlarm 203](#_Toc981)

[5.4 Device Management Functions 204](#_Toc9067)

[5.4.1 IsTFTMachine 204](#_Toc30438)

[5.4.2 GetDeviceStatus 205](#_Toc5233)

[5.4.3 GetDeviceInfo 206](#_Toc10187)

[5.4.4 SetDeviceInfo 209](#_Toc19965)

[5.4.5 SetDeviceTime 210](#_Toc3612)

[5.4.6 SetDeviceTime2 211](#_Toc31564)

[5.4.7 GetDeviceTime 212](#_Toc6656)

[5.4.8 GetSerialNumber 213](#_Toc32331)

[5.4.9 GetProductCode 213](#_Toc30373)

[5.4.10 GetFirmwareVersion 214](#_Toc10902)

[5.4.11 GetSDKVersion 215](#_Toc20920)

[5.4.12 GetDeviceIP 215](#_Toc1439)

[5.4.13 SetDeviceIP 216](#_Toc22389)

[5.4.14 GetDeviceMAC 217](#_Toc7420)

[5.4.15 SetDeviceMAC 218](#_Toc20542)

[5.4.16 GetWiegandFmt 218](#_Toc19494)

[5.4.17 SetWiegandFmt 219](#_Toc29656)

[5.4.18 GetCardFun 220](#_Toc21808)

[5.4.19 SetDeviceCommPwd 220](#_Toc26571)

[5.4.20 SetCommPassword 221](#_Toc3139)

[5.4.21 QueryState 222](#_Toc18799)

[5.4.22 GetVendor 223](#_Toc5086)

[5.4.23 GetDeviceStrInfo 223](#_Toc27050)

[5.4.24 GetPlatform 224](#_Toc22569)

[5.4.25 ReadAOptions 225](#_Toc20341)

[5.4.26 GetSysOption 226](#_Toc10722)

[5.4.27 SetSysOption 227](#_Toc32666)

[5.4.28 GetDeviceStatusEx 227](#_Toc7593)

[5.5 Others 228](#_Toc15175)

[5.5.1 Device Control Functions 228](#_Toc15032)

[5.5.1.1 ClearAdministrators 228](#_Toc13428)

[5.5.1.2 EnableDevice 229](#_Toc7240)

[5.5.1.3 EnableClock 229](#_Toc28196)

[5.5.1.4 DisableDeviceWithTimeOut 230](#_Toc820)

[5.5.1.5 PowerOffDevice 231](#_Toc28685)

[5.5.1.6 RestartDevice 231](#_Toc405)

[5.5.2 Online Registration Functions 232](#_Toc28721)

[5.5.2.1 StartEnroll 232](#_Toc30835)

[5.5.2.2 StartEnrollEx 233](#_Toc27230)

[5.5.2.3 StartVerify 234](#_Toc8122)

[5.5.2.4 StartIdentify 235](#_Toc7360)

[5.5.2.5 CancelOperation 235](#_Toc21265)

[5.5.3 Card Operation Functions 236](#_Toc23010)

[5.5.3.1 WriteLCD 236](#_Toc29264)

[5.5.3.2 ClearLCD 236](#_Toc25553)

[5.5.3.3 WriteCard 237](#_Toc23196)

[5.5.3.4 EmptyCard 238](#_Toc24896)

[5.5.4 Others 239](#_Toc25976)

[5.5.4.1 GetLastError 239](#_Toc17625)

[5.5.4.2 GetHIDEventCardNumAsStr 242](#_Toc1353)

[5.5.4.3 CaptureImage 242](#_Toc42)

[5.5.4.4 UpdateFirmware 243](#_Toc4903)

[5.5.4.5 BeginBatchUpdate 244](#_Toc26307)

[5.5.4.6 BatchUpdate 245](#_Toc16358)

[5.5.4.7 CancelBatchUpdate 245](#_Toc2949)

[5.5.4.8 PlayVoice 246](#_Toc27329)

[5.5.4.9 PlayVoiceByIndex 247](#_Toc28981)

[5.5.4.10 ReadAttRule 247](#_Toc29200)

[5.5.4.11 SaveTheDataToFile 248](#_Toc28874)

[5.5.4.12 ReadTurnInfo 249](#_Toc30442)

[5.5.4.13 SSR\_OutPutHTMLRep 249](#_Toc2836)

[5.5.4.14 SetCommuTimeOut 251](#_Toc4774)

[5.5.4.15 SendFileByType 252](#_Toc18604)

[5.5.4.16 SetCommProType 252](#_Toc13721)

[5.5.4.17 SetCommProType 253](#_Toc8778)

[5.5.4.18 GetConnectStatus 254](#_Toc24702)

[5.5.4.19 SetDeviceTableData 254](#_Toc31995)

[5.5.4.20 SearchDevice 255](#_Toc13933)

[6 FAQs 256](#_Toc24775)

[6.1 How to Download Attendance Records? 256](#_Toc4950)

[6.2 How to Create a User Online? 257](#_Toc1480)

[6.3 How to Import or Download Data from USB Disk? 257](#_Toc29509)

[6.4 How to Use Biokey to Write the Collected Fingerprint Templates Offline? 259](#_Toc29381)

[6.5 How to Obtain All Information of All Users? 259](#_Toc6413)

[6.6 How to Connect to the Device? 260](#_Toc458)

[6.7 Password Is Invalid After SetUserInfo Is Used. 260](#_Toc5899)

[6.8 How to Convert an Online Template into an Offline Template? 260](#_Toc593)

[6.9 Demo Program Fails to Connect to the Device 260](#_Toc13075)

[6.10 Offline Fingerprint Device Keeps Working After Being Connected 261](#_Toc11776)

[6.11 Illegal Characters Are Displayed or Screen Display Is Abnormal After Non-English Names or Short Messages Are Uploaded to the Device 261](#_Toc9936)

[6.12 Card Management Problems 261](#_Toc3079)

[6.13 Firewall or Router Traversal 262](#_Toc2078)

[6.14 Difference between ZKFinger10.0 and ZKFinger9.0 and Comparison between Templates 262](#_Toc152)

[6.15 Uploading a Large Capacity of Fingerprints 262](#_Toc16670)

[6.16 Differences between High-speed Upload and Ordinary Upload 262](#_Toc28957)

[6.17 How to Determine Whether the Device Uses ZKFinger10.0 or ZKFinger9.0? 263](#_Toc32286)

[6.18 How to Upload, Download, and Delete ZKFinger10.0 Templates? 263](#_Toc27408)

[6.19 How to Upload, Download, and Delete ZKFinger9.0 Templates? 264](#_Toc26505)

[6.20 How to Download a Face Template? 264](#_Toc27299)

[6.21 A description of the special parameters of GetSysOption 265](#_Toc28788)

# SDK Description

The Standalone communication SDK is an interface for data communication with standalone fingerprint devices, access control devices, and RFID card devices. It can be used to conveniently manage user information and fingerprints, download attendance records, operation records, user information, and fingerprint templates, set devices, and configure access control. The SDK is used to:

1. Download attendance records.

2. Upload and download user information, card information, fingerprints, and face information.

3. Set access control rules of access control devices.

4. Set device time, match thresholds, etc.

5. Trigger various events of devices in real time, for example, fingerprint verification.

6. Directly enroll users online.

7. Set SMS and work code (available only on devices that support this function) of users.

8. Set personalized prompt tones, function keys, etc.

# Quick Start

## Terms

* + - * 1. Real-time event

After the SDK and the device communicate with each other successfully, some operations on the device (for example, connecting to the device, verifying a user, and enrolling a user) trigger corresponding events in real time, and data is transmitted to the PC (host computer). The triggered events are called real-time events. Users can monitor device states and user operations in real time through real-time events.

* + - * 1. FP

Shortened form of "fingerprint".

* + - * 1. Fingerprint algorithm

A fingerprint algorithm refers to the algorithm used to generate and verify fingerprint templates. At present, ZKFinger 10.0 is the latest fingerprint algorithm used by ZKTeco. It is a high-speed algorithm with higher performance. For details, see FAQs.

* + - * 1. High-speed buffer

A high-speed buffer refers to the memory requested by the SDK on a PC during usage. In the data upload or download process, data is first saved in the buffer before being processed.

* + - * 1. Time Zone/Time slot, group, open door combination

These three terms are the most important concepts of access control.

A Time Zone/Time slot is a time range. A Time Zone includes the time information of one week, and a time range is specified for each day of this week. For example, the following expression indicates a time range from 00:00 to 22:11 in each day of one week: 00002211000022110000221100002211000022110000221100002211. Generally, 50 Time Zones can be set in the device.

A group is a collection. When many users have the same access control privileges, these users can be added to the same group and use the group Time Zone. Then, Time Zones can be set for the group.

An open door combination refers to the groups that are required for unlock. If the open door combination contains only one group, it indicates that the door is opened when any of the users in this group passes verification. If the open door combination contains two or more groups, the door is opened only after all groups pass verification. For example, an open door combination contains groups A and B, the door is opened only after a member of group A and a member of group B pass verification.

The following figure shows the relationship of the three concepts:



* + - * 1. Operation record

An operation record, also called management record, is a record generated when users or administrators operate on the device, for example, powering on/off the device and enrolling a user.

## 2.2 Common Processes

For details, see the descriptions of the demo program.

### 2.2.1 Downloading Attendance Records



### 2.2.2 Downloading Operation Records



### 2.2.3 Setting Access Control



### 2.2.4 Downloading User Information, or Fingerprint Templates

The left diagram shows the process of downloading the information of all users. The right diagram shows the process of downloading the information of a specified user.



### 2.2.5 Receiving Real-time Events

Real-time events can be received in two modes. The second mode is recommended.



### 2.2.6 Enrolling Users Online (Uploading Information,and Fingerprint Templates of Users)

There are two online user enrollment modes. The left diagram shows the process in which the device accesses the enrollment interface to enroll a user after being connected. The right diagram shows the process of creating a user on the device and uploading the card number, password, and fingerprint information for the user (that is, enrolling a card user, a password user, and a fingerprint user).



### 

### 

### 2.2.7 Uploading Short Messages



# Related Attributes

## AccGroup

To set or query the group to which a specified user belongs.

**Type:** LONG

**Attention**

If this attribute is set before uploading the user, set the group to which this user belongs when invoking SetUserInfo. Otherwise, the default group 1 takes effect. This attribute is configurable.

**Note**

Applicable to B&W, TFT, and IFACE devices

## AccTimeZones

To set the usage period of a user.

**Type:** LONG

**Attention**

1. If this attribute is set before uploading the user, set the usage period of the user when invoking SetUserInfo.

2.This attribute is of the LONG\* type. It is a LONG-type array with the subscript 3. This attribute is configurable.

**Note**

Applicable to B&W, TFT, and IFACE devices

## BASE64

To set the code type.

**Type:** LONG

**Attention**

If this attribute is set to True, the SDK will output a Base64 code when outputting the character string template. Otherwise, it will output a hexadecimal code.

**Note**

Applicable to B&W, TFT, and IFACE devices

## CardNumber

To set or read the card number of a user.

**Type:** LONG

**Attention**

If this attribute cannot be used, invoke GetStrCardnumber and SetStrCardnumber. This attribute is configurable.

**Note**

Applicable to B&W, TFT, and IFACE devices

## CommPort

To set a serial port number, or a port number used for RS485 connection.

**Type:** LONG

**Attention**

The attribute is of the LONG type and is configurable.

**Note**

Applicable to B&W, TFT, and IFACE devices

## ConvertBIG5

To convert simplified Chinese characters into traditional Chinese characters.

**Type:** LONG

**Attention**

1. If the value of this attribute is set to True, the SDK will automatically convert simplified Chinese characters into traditional Chinese characters for further development. This function is invalid on a multi-language machine. Therefore, do not set this attribute.

2.This attribute is invalid on a multi-language machine and later versions. Therefore, do not set this attribute. You do not need to modify this attribute in versions later than ZEM100 5.22 and ZEM200 5.30.

**Note**

Applicable to B&W, TFT, and IFACE devices

## PINWidth

To label the maximum length of the user code, which is an Arabic numeral.

**Type:** LONG

**Attention**

The attribute is of the LONG type and is read-only.

**Note**

Applicable to B&W, TFT, and IFACE devices

## GetStrCardNumber

VARIANT\_BOOL GetStrCardNumber(BSTR \* ACardNumber)

To query the value of the SDK attribute cardnumber. You can invoke this function to query the card number of a user after obtaining the information about this user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ACardNumber | BSTR\* | [out] | Card number |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

Attention

Note

Applicable to B&W

## SetStrCardNumber

VARIANT\_BOOL SetStrCardNumber(BSTR ACardNumber)

To set the value of the SDK attribute cardnumber. Before setting user information, you are advised to invoke this function to set the card number of the user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ACardNumber | BSTR | [in] | Card number |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

Attention

Note

Applicable to B&W

## IsNewFirmwareMachine

VARIANT\_BOOL IsNewFirmwareMachine(LONG dwMachineNumber)

To identify whether the current device firmware are new architecture firmware.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

Attention

Note

Apply to B&W, TFT, IFACE devices

## GetDeviceFirmwareVersion

VARIANT\_BOOL GetDeviceFirmwareVersion(LONG dwMachineNumber, BSTR \* strVersion)

To query the firmware version.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| strVersion | BSTR\* | [out] | Firmware version of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

# Real-time Event Functions

## Obtaining Real-Time Events

### RegEvent

VARIANT\_BOOL RegEvent(LONG dwMachineNumber, LONG EventMask)

To register a real-time event to be triggered.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| EventMask | LONG | [in] | Event ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The event IDs are described as follows:  
1 OnAttTransaction OnAttTransactionEx  
2(1<<1) OnFinger  
4(1<<2) OnNewUser  
8(1<<3) OnEnrollFinger OnEnrollFingerEx  
16(1<<4) OnKeyPress  
256(1<<7) OnVerify  
512(1<<8) OnFingerFeature  
1024(1<<9) OnDoor OnAlarm  
2048(1<<10) OnHIDNum  
4096(1<<11) OnWriteCard  
8192(1<<12) OnEmptyCard  
16384(1<<13) OnDeleteTemplate  
To register multiple real-time events, perform the OR operation for the binary event IDs. To register all real-time events, set EventMask to 65535.

Note

Applicable to B&W, TFT, and IFACE devices

### ReadRTLog

VARIANT\_BOOL ReadRTLog(LONG dwMachineNumber)

To read real-time events to the buffer of the PC.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

Attention

This function can be used together with GetRTLog to actively obtain real-time events from the machine after the machine is connected successfully.

Note

Applicable to B&W, TFT, and IFACE devices

### GetRTLog

VARIANT\_BOOL GetRTLog(LONG dwMachineNumber)

To get a real-time event from the buffer of the PC and trigger this event.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function can be used together with ReadRTLog to actively obtain real-time events from the machine after the machine is connected successfully.

Note

Applicable to B&W, TFT, and IFACE devices

## Real-Time Events

### OnConnected

OnConnected(LONG MachineNumber)

To trigger this event when the machine is connected successfully.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| MachineNumber | LONG | [in] | Machine ID |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnDisConnected

OnDisConnected(LONG MachineNumber)

To trigger this event when the machine is disconnected.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| MachineNumber | LONG | [in] | Machine ID |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnAlarm

OnAlarm (LONG AlarmType,LONG EnrollNumber,LONG Verified)

To trigger this event when the machine raises an alarm.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| AlarmType | LONG | [in] | Alarm type |
| EnrollNumber | LONG | [in] | User ID |
| Verified | LONG | [in] | Whether to perform verification |

Returns

None

See also

Attention

1. Alarm Type: alarm type. The value 55 indicates a tamper alarm, 58 misoperation alarm, 32 duress alarm, and 34 passback alarm.

2.EnrollNumber: user ID. The value is 0 if the alarm is a tamper alarm, misoperation alarm, or duress key alarm. The value is the user ID if the alarm is another type of duress alarm or a passback alarm.

3.Verified: whether to perform verification. The value is 0 when the alarm is a tamper or misoperation alarm, and is 1 if the alarm is of other types.

Note

Applicable to B&W, TFT, and IFACE devices

### OnDoor

OnDoor (LONG EventType)

To trigger this event when the machine opens the door.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EventType | LONG | [in] | Door opening type |

Returns

None

See also

Attention

EventType: specifies the door opening type. The value 4 indicates that the door is not closed properly or has been opened, 53 indicates an exit button, 5 indicates that the door has been closed, and 1 indicates that the door is opened unexpectedly.

Note

Applicable to B&W, TFT, and IFACE devices

### OnAttTransaction

OnAttTransaction(LONG EnrollNumber, LONG IsInValid, LONG AttState, LONG VerifyMethod, LONG Year, LONG Month, LONG Day, LONG Hour, LONG Minute, LONG Second, LONG WorkCode)

To trigger this event when the verification is passed.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | LONG | [in] | Alarm type |
| IsInValid | LONG | [in] | Whether the record is valid |
| AttState | LONG | [in] | Attendance status |
| VerifyMode | LONG | [in] | Verification mode of the record |
| Year | LONG | [in] | Year |
| Month | LONG | [in] | Month |
| Day | LONG | [in] | Date |
| Hour | LONG | [in] | Hour |
| Minute | LONG | [in] | Minute |

Returns

None

See also

Attention

1. The VerifyMode parameter specifies the verification mode. The values are described as follows:  
Under normal conditions:  
0 indicates password verification, 1 fingerprint verification, and 2 card verification.  
Under multiple verification modes:  
FP\_OR\_PW\_OR\_RF 0  
FP 1  
PIN 2  
PW 3  
RF 4  
FP\_OR\_PW 5  
FP\_OR\_RF 6  
PW\_OR\_RF 7  
PIN\_AND\_FP 8  
FP\_AND\_PW 9  
FP\_AND\_RF 10  
PW\_AND\_RF 11  
FP\_AND\_PW\_AND\_RF 12  
PIN\_AND\_FP\_AND\_PW 13  
FP\_AND\_RF\_OR\_PIN 14

2.The AttState parameter specifies the attendance status. The values are described as follows:  
0-Check-In Default  
1-Check-Out  
2-Break-Out  
3-Break-In  
4-OT-In  
5-OT-Out

Note

Applicable to B&W

### OnAttTransactionEx

OnAttTransactionEx(BSTR EnrollNumber, LONG IsInValid, LONG AttState, LONG VerifyMethod, LONG Year, LONG Month, LONG Day, LONG Hour, LONG Minute, LONG Second, LONG WorkCode)

To trigger this event when the verification is passed.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | BSTR | [in] | Alarm type |
| IsInValid | LONG | [in] | Whether the record is valid |
| AttState | LONG | [in] | Attendance status |
| VerifyMode | LONG | [in] | Verification mode of the record |
| Year | LONG | [in] | Year |
| Month | LONG | [in] | Month |
| Day | LONG | [in] | Date |
| Hour | LONG | [in] | Hour |
| Minute | LONG | [in] | Minute |
| Second | LONG | [in] | Second |
| WorkCode | LONG | [in] | Work code |

Returns

None

See also

Attention

1. The VerifyMode parameter specifies the verification mode. The values are described as follows:  
Under normal conditions:  
0 indicates password verification, 1 fingerprint verification, and 2 card verification.  
Under multiple verification modes:  
FP\_OR\_PW\_OR\_RF 0  
FP 1  
PIN 2  
PW 3  
RF 4  
FP\_OR\_PW 5  
FP\_OR\_RF 6  
PW\_OR\_RF 7  
PIN\_AND\_FP 8  
FP\_AND\_PW 9  
FP\_AND\_RF 10  
PW\_AND\_RF 11  
FP\_AND\_PW\_AND\_RF 12  
PIN\_AND\_FP\_AND\_PW 13  
FP\_AND\_RF\_OR\_PIN 14

2.The AttState parameter specifies the attendance status. The values are described as follows:  
0-Check-In Default  
1-Check-Out  
2-Break-Out  
3-Break-In  
4-OT-In  
5-OT-Out

The WorkCode parameter specifies the work code. If the machine does not support the work code, 0 is returned.

Note

Applicable to B&W, TFT, and IFACE devices

### OnEnrollFinger

OnEnrollFinger(LONG EnrollNumber, LONG FingerIndex, LONG ActionResult, LONG TemplateLength)

To trigger this event when registering a fingerprint.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | LONG | [in] | ID of the user whose fingerprint is to be registered |
| FingerIndex | LONG | [in] | Index of the fingerprint to be registered |
| ActionResult | LONG | [in] | Operation result. The value is 0 if the operation succeeds and is larger than 0 if the operation fails. |
| TemplateLength | LONG | [in] | Length of the fingerprint template |

Returns

None

See also

Attention

Use under certain circumstances of 9-digit job width and 5-digit job width.

Note

Applicable to B&W devices

### OnEnrollFingerEx

OnEnrollFinger(**BSTR** EnrollNumber, LONG FingerIndex, LONG ActionResult, LONG TemplateLength)

To trigger this event when registering a fingerprint.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | BSTR | [in] | ID of the user whose fingerprint is to be registered |
| FingerIndex | LONG | [in] | Index of the fingerprint to be registered |
| ActionResult | LONG | [in] | Operation result. The value is 0 if the operation succeeds and is larger than 0 if the operation fails. |
| TemplateLength | LONG | [in] | Length of the fingerprint template |

Returns

None

See also

Attention

Note

Applicable to TFT, and IFACE devices

### OnDeleteTemplate

OnDeleteTemplate(LONG EnrollNumber, LONG FingerIndex)

To trigger this event when deleting a fingerprint template on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | LONG | [in] | ID of the user to which the fingerprint template to be deleted belongs |
| FingerIndex | LONG | [in] | Index of the fingerprint template to be deleted |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnFinger

OnFinger()

To trigger this message when a fingerprint is scanned by the machine.

Parameters

None

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnFingerFeature

OnFingerFeature(LONG Score)

To trigger this message if a finger is pressed onto the fingerprint reader when registering user fingerprints.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| Score | LONG | [in] | Quality score of the fingerprint |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnHIDNum

OnHIDNum (LONG CardNumber)

To trigger this message when punching a card.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| CardNumber | LONG | [in] | Card number |

Returns

None

See also

Attention

The card can be an ID card or HID card. For an MIFARE card, this event will be triggered only when it is used as an ID card.

Note

Applicable to B&W, TFT, and IFACE devices

### OnKeyPress

OnKeyPress(LONG Key)

To trigger this message when a key is available.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| Key | LONG | [in] | Key value |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnNewUser

OnNewUser (LONG EnrollNumber)

To trigger this message when a new user is registered successfully.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | LONG | [in] | ID of the newly registered user |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnVerify

OnVerify (LONG UserID)

To trigger this message during user verification.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| UserID | LONG | [in] | ID of the user to be verified |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnWriteCard

OnWriteCard (LONG EnrollNumber, LONG ActionResult, LONG Length)

To trigger this event when the machine writes data to a card.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| EnrollNumber | LONG | [in] | User ID of the card to which data is to be written |
| ActionResult | LONG | [in] | Result of the write operation. The value 0 indicates operation success and other values indicate operation failure. |
| Length | LONG | [in] | Total size of the data written to the card |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnEmptyCard

OnEmptyCard (LONG ActionResult)

To trigger this event when the machine writes data to a card.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ActionResult | LONG | [in] | Result of the clear operation. The value 0 indicates operation success and other values indicate operation failure. |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### OnEMData

OnEMData (LONG DataType, LONG DataLen, CHAR\* DataBuffer)

To trigger this event when the machine sends an unknown event to the SDK.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| DataType | LONG | [in] | Event type |
| DataLen | LONG | [in] | Total data length |
| DataBuffer | CHAR\* | [in] | Data |

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

# Common Functions

## Device Connection Functions

### Connect\_Net

VARIANT\_BOOL Connect\_Net(BSTR IPAdd, LONG Portl)

To connect to the machine to set up a network connection with the machine by using an IP address.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| IPAdd | BSTR | [in] | IP address of the machine |
| Portl | LONG | [in] | Port number of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The default port number for connecting to the machine is 4370.

Note

Applicable to B&W, TFT, and IFACE devices

### Connect\_Com

VARIANT\_BOOL Connect\_Com(LONG ComPort, LONG MachineNumber, LONG BaudRate)

To connect to the machine through a serial port, that is, the RS232 RS485 port.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ComPort | LONG | [in] | Serial port of the PC to be connected to the machine |
| MachineNumber | LONG | [in] | Machine ID |
| BaudRate | LONG | [in] | Baud rate |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function is also used when a machine communicates with a PC by using a USB client. The precondition is that the USB client drive is installed beforehand to simulate a virtual serial port, which can be viewed in the device manager on the PC. The program can also search for this serial port. For details, see the description of USBClient in DEMO.

Note

Applicable to B&W, TFT, and IFACE devices

### Connect\_USB

VARIANT\_BOOL Connect\_USB(LONG MachineNumber)

To connect to the machine through a USB port.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| MachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

[Connect\_Com](mk:@MSITStore:C:\\Users\\Hess\\Desktop\\clone\\standalonesdk\\release\\sdk-6.3.1.34-doc-1.0.4-demo-1.1.15\\Doc-Ver1.0.4\\SDKAPI-en.chm::/1_8_terminal_8h.html" \l "a7325d3e70d7e4e25c9c43b08ae063298" \o "To connect to the machine through a serial port, that is, the RS232 RS485 port. )

Attention

This function applies only to H series machines and cannot be used in the communication conducted through a USB client. For details about the communication conducted through a USB client, see the description of Connect\_Com.

Note

Applicable to B&W

### Disconnect

Disconnect()

To disconnect from the machine to release relevant resources.

Parameters

None

Returns

None

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

## Data Management Functions

### Attendance Record Data

#### ReadGeneralLogData

VARIANT\_BOOL ReadGeneralLogData(LONG dwMachineNumber)

To read attendance records to the internal buffer of the PC. The function is the same as ReadAllGLogData.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### ReadAllGLogData

VARIANT\_BOOL ReadAllGLogData(LONG dwMachineNumber)

To read attendance records to the internal buffer of the PC. The function is the same as ReadGeneralLogData.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### ReadTimeGLogData

VARIANT\_BOOL ReadTimeGLogData(LONG dwMachineNumber, BSTR sTime, BSTR eTime)

To download attendance records based on the specified start time and end time, accurate to seconds.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sTime | BSTR | [in] | Start time in the format of YYYY-MM-DD hh:mm:ss |
| eTime | BSTR | [in] | End time in the format of YYYY-MM-DD hh:mm:ss |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### ReadNewGLogData

VARIANT\_BOOL ReadNewGLogData(LONG dwMachineNumber)

To download the new generated attendance records.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### GetGeneralLogData

VARIANT\_BOOL GetGeneralLogData(LONG dwMachineNumber, LONG\*

dwTMachineNumber, LONG\* dwEnrollNumber, LONG\* dwEMachineNumber, LONG\*

dwVerifyMode, LONG\* dwInOutMode, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay,

LONG\* dwHour, LONG\* dwMinute)

To read attendance records from the internal buffer one by one. Before using this function,

execute ReadAllGLogData or ReadGeneralLogData to read the attendance records from the

machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to

the next attendance record. This function is the same as GetAllGLogData. They differ only in the

interface name for compatibility.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwTMachineNumber | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the machine ID of  an attendance  record. |
| dwEnrollNumber | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the user ID of an  attendance record. |
| dwEMachineNumber | LONG\* | [out] | Pointer that points37  to the LONG  variable. Its value  is the machine ID of  an attendance  record. |
| dwVerifyMode | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the verification  mode of an  attendance record. |
| dwInOutMode | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the attendance  status of an  attendance record. |
| dwYear | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the year of an  attendance record. |
| dwMonth | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the month of an  attendance record. |
| dwDay | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the day of an  attendance record. |
| dwHour | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the hour of an  attendance record. |
| dwMinute | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the minute of an  attendance record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The dwVerifyMode parameter specifies the verification mode. The values are described as

follows:

Generally:

0 indicates password verification, 1 indicates fingerprint verification and 2 indicates card

verification.

Under multiple verification modes:

FP\_OR\_PW\_OR\_RF 0

FP 1

PIN 2

PW 3

RF 4

FP\_OR\_PW 5

FP\_OR\_RF 6

PW\_OR\_RF 7

PIN\_AND\_FP 8

FP\_AND\_PW 9

FP\_AND\_RF 10

PW\_AND\_RF 11

FP\_AND\_PW\_AND\_RF 12

PIN\_AND\_FP\_AND\_PW 13

FP\_AND\_RF\_OR\_PIN 14

2. The dwInOutMode parameter specifies the attendance status. The values are described as

follows:

0-Check-In Default

1-Check-Out

2-Break-Out

3-Break-In39

4-OT-In

5-OT-Out

Note

Applicable to B&W, TFT, and IFACE devices

#### **GetAllGLogData**

VARIANT\_BOOL GetAllGLogData(LONGdwMachineNumber , LONG\* dwTMachineNumber,

LONG\* dwEnrollNumber,LONG\* dwEMachineNumber, LONG\* dwVerifyMode, LONG\*

dwInOutMode, LONG\*dwYear, LONG\* dwMonth,LONG\* dwDay, LONG\* dwHour,LONG\*

dwMinute)

To read attendance records from the internal buffer one by one. Before using this function,  
execute ReadAllGLogData or ReadGeneralLogData to read the attendance records from the  
machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to  
the next attendance record. This function is the same as GetGeneralLogData. They differ only in  
the interface name for compatibility.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwTMachineNumber | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the machine ID of  an attendance  record. |
| dwEnrollNumber | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the user ID of an  attendance record. |
| dwEMachineNumber | LONG\* | [out] | Pointer that points37  to the LONG  variable. Its value  is the machine ID of  an attendance  record. |
| dwVerifyMode | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the verification  mode of an  attendance record. |
| dwInOutMode | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the attendance  status of an  attendance record. |
| dwYear | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the year of an  attendance record. |
| dwMonth | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the month of an  attendance record. |
| dwDay | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the day of an  attendance record. |
| dwHour | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the hour of an  attendance record. |
| dwMinute | LONG\* | [out] | Pointer that points  to the LONG  variable. Its value  is the minute of an  attendance record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

GetGeneralLogData

Attention

1. The dwVerifyMode parameter specifies the verification mode. The meanings of the values are

the same as those of the GetGeneralLogData parameter.

2. The dwInOutMode parameter specifies the attendance status. The meanings of the values are

the same as those of the GetGeneralLogData parameter.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetGeneralLogDataStr

VARIANT\_BOOL GetGeneralLogDataStr(LONG dwMachineNumber, LONG\* dwEnrollNumber, LONG \* dwVerifyMode, LONG \* dwInOutMode, BSTR \* TimeStr)

To read attendance records from the internal buffer one by one. Before using this function, execute ReadAllGLogData or ReadGeneralLogData to read the attendance records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next attendance record. This function is the same as GetGeneralLogData. They differ in the format of time in the returned values.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the user ID of an attendance record. |
| dwVerifyMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the verification mode of an attendance record. The meanings of the values are the same as those of the GetGeneralLogData parameter. |
| dwInOutMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the attendance status of an attendance record. The meanings of the values are the same as those of the GetGeneralLogData parameter. |
| TimeStr | BSTR\* | [out] | Pointer that points to the LONG variable. Its value is the attendance time of an attendance record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

[GetGeneralLogData](mk:@MSITStore:C:\\Users\\Hess\\Desktop\\clone\\standalonesdk\\release\\sdk-6.3.1.34-doc-1.0.4-demo-1.1.15\\Doc-Ver1.0.4\\SDKAPI-en.chm::/3_802_8_data-_record_8h.html" \l "a84d2e60405001aaf8de986791f2daca2" \o "To read attendance records from the internal buffer one by one. Before using this function...)

Attention

1. The dwVerifyMode parameter specifies the verification mode. The meanings of the values are the same as those of the GetGeneralLogData parameter.

2. The dwInOutMode parameter specifies the attendance status. The meanings of the values are the same as those of the GetGeneralLogData parameter.

Note

Applicable to B&W

#### GetGeneralExtLogData

VARIANT\_BOOL GetGeneralExtLogData(LONG dwMachineNumber, LONG\* dwEnrollNumber, LONG\* dwVerifyMode, LONG\* dwInOutMode, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay, LONG\* dwHour, LONG\* dwMinute, LONG\* dwSecond, LONG\* dwWorkCode, LONG \* dwReserved)

To read attendance records from the internal buffer one by one. Before using this function, execute ReadAllGLogData or ReadGeneralLogData to read the attendance records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next attendance record. This function is an enhanced version of GetGeneralLogData. They are compatible.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the user ID of an attendance record. |
| dwVerifyMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the verification mode of an attendance record. |
| dwInOutMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the attendance status of an attendance record. |
| dwYear | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the year of an attendance record. |
| dwMonth | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the month of an attendance record. |
| dwDay | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the day of an attendance record. |
| dwHour | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the hour of an attendance record. |
| dwMinute | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the minute of an attendance record. |
| dwSecond | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the second of an attendance record. |
| dwWorkCode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the work code of an attendance record. |
| dwReserved | LONG\* | [out] | Reserved parameter. It is meaningless. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

[GetGeneralLogData](mk:@MSITStore:C:\\Users\\Hess\\Desktop\\clone\\standalonesdk\\release\\sdk-6.3.1.34-doc-1.0.4-demo-1.1.15\\Doc-Ver1.0.4\\SDKAPI-en.chm::/3_802_8_data-_record_8h.html" \l "a84d2e60405001aaf8de986791f2daca2" \o "To read attendance records from the internal buffer one by one. Before using this function...)

Attention

1. The dwVerifyMode parameter specifies the verification mode. The meanings of the values are the same as those of the GetGeneralLogData parameter.

2. The dwInOutMode parameter specifies the attendance status. The meanings of the values are the same as those of the GetGeneralLogData parameter.

Note

Applicable to B&W

#### SSR\_GetGeneralLogData

VARIANT\_BOOL SSR\_GetGeneralLogData(LONG dwMachineNumber, BSTR\* dwEnrollNumber, LONG\* dwVerifyMode, LONG\* dwInOutMode, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay, LONG\* dwHour, LONG\* dwMinute, LONG\* dwSecond, LONG\* dwWorkcode)

To read attendance records from the internal buffer one by one. Before using this function, execute ReadAllGLogData or ReadGeneralLogData to read the attendance records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next attendance record. This function is the same as GetGeneralLogData. The difference is that this function applies to color-screen machines.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR\* | [out] | Pointer that points to the BSTR variable. Its value is the user ID of an attendance record. A user ID contains a maximum of 24 digits. |
| dwVerifyMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the verification mode of an attendance record. |
| dwInOutMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the attendance status of an attendance record. |
| dwYear | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the year of an attendance record. |
| dwMonth | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the month of an attendance record. |
| dwDay | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the day of an attendance record. |
| dwHour | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the hour of an attendance record. |
| dwMinute | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the minute of an attendance record. |
| dwSecond | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the second of an attendance record. |
| dwWorkcode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the work code of an attendance record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The dwVerifyMode parameter specifies the verification mode. The values are described as follows:  
Generally:  
0 indicates password verification, 1 indicates fingerprint verification and 2 indicates card verification.  
Under multiple verification modes:  
FP\_OR\_PW\_OR\_RF 0  
FP 1  
PIN 2  
PW 3  
RF 4  
FP\_OR\_PW 5  
FP\_OR\_RF 6  
PW\_OR\_RF 7  
PIN\_AND\_FP 8  
FP\_AND\_PW 9  
FP\_AND\_RF 10  
PW\_AND\_RF 11  
FP\_AND\_PW\_AND\_RF 12  
PIN\_AND\_FP\_AND\_PW 13  
FP\_AND\_RF\_OR\_PIN 14

2. The dwInOutMode parameter specifies the attendance status. The values are described as follows:  
0-Check-In Default  
1-Check-Out  
2-Break-Out  
3-Break-In  
4-OT-In  
5-OT-Out

Note

Applicable to TFT and IFACE devices

#### ClearGLog

VARIANT\_BOOL ClearGLog(long dwMachineNumber)

To clear all attendance records on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### DeleteAttlogBetweenTheDate

VARIANT\_BOOL DeleteAttlogBetweenTheDate(LONG dwMachineNumber, BSTR sTime, BSTR eTime)

To delete attendance records based on the specified start time and end time, accurate to seconds.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sTime | BSTR | [in] | Start time in the format of YYYY-MM-DD hh:mm:ss |
| eTime | BSTR | [in] | End time in the format of YYYY-MM-DD hh:mm:ss |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### DeleteAttlogByTime

VARIANT\_BOOL DeleteAttlogByTime(LONG dwMachineNumber, BSTR sTime)

To delete all attendance records generated before the specified time point, accurate to seconds.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sTime | BSTR | [in] | Start time point in the format of YYYY-MM-DD hh:mm:ss |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### SSR\_GetGeneralLogDataWithMask

**VARIANT\_BOOL SSR\_GetGeneralLogDataWithMask(**

**LONG dwMachineNumber,**

**BSTR \*dwEnrollNumber,**

**LONG \*dwVerifyMode,**

**LONG \*dwInOutMode,**

**LONG \*dwYear,**

**LONG \*dwMonth,**

**LONG \*dwDay,**

**LONG \*dwHour,**

**LONG \*dwMinute,**

**LONG \*dwSecond,**

**LONG \*dwWorkcode,**

**LONG \*dwMask,**

**BSTR \*dwTemperature)**

To read attendance records from the internal buffer one by one, contain mask status and temperature. Before using this function, execute ReadAllGLogData or ReadGeneralLogData to read the attendance records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next attendance record. This function is the same as GetGeneralLogData. The difference is that this function applies to color-screen machines.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR\* | [out] | Pointer that points to the BSTR variable. Its value is the user ID of an attendance record. A user ID contains a maximum of 24 digits. |
| dwVerifyMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the verification mode of an attendance record. |
| dwInOutMode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the attendance status of an attendance record. |
| dwYear | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the year of an attendance record. |
| dwMonth | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the month of an attendance record. |
| dwDay | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the day of an attendance record. |
| dwHour | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the hour of an attendance record. |
| dwMinute | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the minute of an attendance record. |
| dwSecond | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the second of an attendance record. |
| dwWorkcode | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the work code of an attendance record. |
| dwMask | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the mask status of an attendance record. |
| dwTemperature | LONG\* | [out] | Pointer that points to the BSTR variable. Its value is the temperature of an attendance record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The dwVerifyMode parameter specifies the verification mode. The values are described as follows:  
Generally:  
0 indicates password verification, 1 indicates fingerprint verification and 2 indicates card verification.  
Under multiple verification modes:  
FP\_OR\_PW\_OR\_RF 0  
FP 1  
PIN 2  
PW 3  
RF 4  
FP\_OR\_PW 5  
FP\_OR\_RF 6  
PW\_OR\_RF 7  
PIN\_AND\_FP 8  
FP\_AND\_PW 9  
FP\_AND\_RF 10  
PW\_AND\_RF 11  
FP\_AND\_PW\_AND\_RF 12  
PIN\_AND\_FP\_AND\_PW 13  
FP\_AND\_RF\_OR\_PIN 14

2. The dwInOutMode parameter specifies the attendance status. The values are described as follows:  
0-Check-In Default  
1-Check-Out  
2-Break-Out  
3-Break-In  
4-OT-In  
5-OT-Out

Note

Applicable to the device which support mask status and temperature check.

### Operation Record Data

#### ReadSuperLogData

VARIANT\_BOOL ReadSuperLogData(long dwMachineNumber)

To read operation records to the internal buffer of the PC. The function is the same as ReadAllSLogData.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### ReadAllSLogData

VARIANT\_BOOL ReadAllSLogData(long dwMachineNumber)

To read operation records to the internal buffer of the PC. The function is the same as ReadSuperLogData.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### GetSuperLogData

VARIANT\_BOOL GetSuperLogData(LONG dwMachineNumber, LONG\* dwTMachineNumber, LONG\* dwSEnrollNumber, LONG\* Params4, LONG\* Params1, LONG\* Params2, LONG\* dwManipulation, LONG\* Params3, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay, LONG\* dwHour, LONG\* dwMinute)

To read operation records from the internal buffer one by one. Before using this function, execute ReadAllSLogData or ReadSuperLogData to read the operation records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next operation record. This function differs from GetSuperLogData2 in that the GetSuperLogData2 function can obtain the operation record time accurate to seconds.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwTMachineNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the machine ID of an operation record. |
| dwSEnrollNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the administrator ID of an operation record. |
| Params4 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| Params1 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| Params2 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwManipulation | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the operation type. |
| Params3 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwYear | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the year of an operation record. |
| dwMonth | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the month of an operation record. |
| dwDay | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the day of an operation record. |
| dwHour | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the hour of an operation record. |
| dwMinute | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the minute of an operation record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The meaning of different combinations of the dwManipulation, Params1, Params2, Params3 and Params4 parameters is as follows:

1. dwManipulation=0: dwManipulation indicates starting the machine.

2. dwManipulation=1: dwManipulation indicates shutting down the machine.

3. dwManipulation=3: dwManipulation indicates that an alarm is raised. Params1 specifies the alarm type. The value 58 indicates a misoperation alarm, 54 door sensor alarm, 53 door opening alarm, 55 tamper alarm, and 65535 shutdown alarm.

4. dwManipulation=4: dwManipulation indicates accessing the menu.

5. dwManipulation=5: dwManipulation indicates changing settings. Params1 specifies the number of the option that is set.

6. dwManipulation=6: dwManipulation indicates registering fingerprints. Params1 specifies the ID of the operated user. Params2 specifies the operation result, the value 0 indicates operation success, and other values indicate operation failure. Params3 specifies the registered fingerprint index. Params4 specifies the length of the fingerprint template, and the value 2 indicates duress fingerprint.

7. dwManipulation=7: dwManipulation indicates registering the password. Params1 specifies the ID of the operated user. Params2 specifies the operation result, the value 0 indicates operation success, and other values indicate operation failure.

8. dwManipulation=14: dwManipulation indicates creating an MF card. Params1 specifies the ID of the operated user. Params2 specifies the operation result, the value 0 indicates operation success, and other values indicate operation failure. Params3 specifies the number of fingerprints written to the MF card. Params4 specifies the size of fingerprint data written to the MF card.

9. dwManipulation=20: dwManipulation indicates copying data from the MF card to the machine. Params1 specifies the ID of the operated user. Params2 specifies the operation result, the value 0 indicates operation success, and other values indicate operation failure. Params3 specifies the number of fingerprints read to the MF card.

10. dwManipulation=22: dwManipulation indicates restoring factory settings.

11. dwManipulation=30: dwManipulation indicates registering a new user. Params1 specifies the ID of the operated user. Params2 specifies the operation result, the value 0 indicates operation success, and other values indicate operation failure.

12. dwManipulation=32: dwManipulation indicates duress alarm. Params1 specifies whether the alarm is a verification alarm, the value 0 indicates a key alarm, and 1 indicates a verification alarm. Note: If the alarm is a verification alarm, dwSEnrollNumber will return the duress user ID.

13. dwManipulation=34: dwManipulation indicates blockade. Params1 specifies whether to block.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetAllSLogData

GetAllSLogData(LONG dwMachineNumber, LONG\* dwTMachineNumber, LONG\* dwSEnrollNumber, LONG\* dwSMachineNumber, LONG\* dwGEnrollNumber, LONG\* dwGMachineNumber, LONG\* dwManipulation, LONG\* dwBackupNumber, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay, LONG\* dwHour, LONG\* dwMinute)

To read operation records from the internal buffer one by one. Before using this function, execute ReadAllSLogData or ReadSuperLogData to read the operation records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next operation record. This function is the same as GetSuperLogData. They differ only in the interface name for compatibility.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwTMachineNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the machine ID of an operation record. |
| dwSEnrollNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the administrator ID of an operation record. |
| dwSMachineNumber | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwGEnrollNumber | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwGMachineNumber | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwManipulation | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the operation type. The meanings of the values are as follows: |
| dwBackupNumber | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwYear | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the year of an operation record. |
| dwMonth | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the month of an operation record. |
| dwDay | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the day of an operation record. |
| dwHour | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the hour of an operation record. |
| dwMinute | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the minute of an operation record.\* |

Returns

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

[GetSuperLogData](mk:@MSITStore:C:\\Users\\Hess\\Desktop\\clone\\standalonesdk\\release\\sdk-6.3.1.34-doc-1.0.4-demo-1.1.15\\Doc-Ver1.0.4\\SDKAPI-en.chm::/3_802_8_data-_record_8h.html" \l "afc12de9c37a2a2826790f9472b148718" \o "To read operation records from the internal buffer one by one. Before using this function, execute ReadAllSLogData or ReadSuperLogData to read the operation records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next operation record. This function differs from GetSuperLogData2 in that the GetSuperLogData2 function can obtain the operation record time accurate to seconds. )

Attention

The meanings of the parameters are the same as those of GetSuperLogData.

Note

Applicable to B&W, TFT, and IFACE devices

#### ClearSLog

VARIANT\_BOOL ClearSLog(LONG dwMachineNumber)

To clear all operation records on the machine.

Parameters.

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### GetSuperLogData2

VARIANT\_BOOL GetSuperLogData2(LONG dwMachineNumber, LONG\* dwTMachineNumber, LONG\* dwSEnrollNumber, LONG\* Params4, LONG\* Params1, LONG\* Params2, LONG\* dwManipulation, LONG\* Params3, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay, LONG\* dwHour,LONG\* dwMinute, LONG\* dwSecs)

To read operation records from the internal buffer one by one. Before using this function, execute ReadAllSLogData or ReadSuperLogData to read the operation records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next operation record. GetSuperLogData and GetSuperLogData2 differ in that the GetSuperLogData2 function can obtain the operation record time accurate to seconds.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwTMachineNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the machine ID of an operation record. |
| dwSEnrollNumber | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the administrator ID of an operation record. |
| Params4 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| Params1 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| Params2 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwManipulation | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the operation type. The meanings of the values are as follows: |
| Params3 | LONG\* | [out] | Pointer that points to the LONG variable. The value varies according to that of dwManipulation. |
| dwYear | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the year of an operation record. |
| dwMonth | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the month of an operation record. |
| dwDay | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the day of an operation record. |
| dwHour | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the hour of an operation record. |
| dwMinute | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the minute of an operation record. |
| dwSecs | LONG\* | [out] | Pointer that points to the LONG variable. Its value is the second of an operation record. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

[GetSuperLogData](mk:@MSITStore:C:\\Users\\Hess\\Desktop\\clone\\standalonesdk\\release\\sdk-6.3.1.34-doc-1.0.4-demo-1.1.15\\Doc-Ver1.0.4\\SDKAPI-en.chm::/3_802_8_data-_record_8h.html" \l "afc12de9c37a2a2826790f9472b148718" \o "To read operation records from the internal buffer one by one. Before using this function, execute ReadAllSLogData or ReadSuperLogData to read the operation records from the machine to the internal buffer of the PC. Each time this function is executed, the pointer moves to the next operation record. This function differs from GetSuperLogData2 in that the GetSuperLogData2 function can obtain the operation record time accurate to seconds. )

Attention

dwYear, dwMonth, dwDay, dwHour, dwMinute and dwSecs are all pointers that point to the LONG variable. Their values indicate the date and time of an operation record, accurate to seconds.

Note

Applicable to B&W, TFT, and IFACE devices

### User Information Functions

#### ReadAllUserID

VARIANT\_BOOL ReadAllUserID(LONG dwMachineNumber)

To read all user information to the memory of the PC, including the user ID, password, name, and card number. Fingerprint templates are not read. After this function is executed, invoke the function GetAllUserID to get the user information.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### EnableUser

VARIANT\_BOOL EnableUser(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber, VARIANT\_BOOL bFlag)

To set whether a user account is available.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwEMachineNumber | LONG | [in] | Invalid parameter. It is meaningless. |
| dwBackupNumber | LONG | [in] | Invalid parameter. It is meaningless. |
| bFlag | BOOL | [in] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The bFlag parameter is a flag that indicates whether a user account is enabled. The value True indicates that the user account is enabled and False indicates that the user account is disabled.

Note

Applicable to B&W

#### SSR\_EnableUser

VARIANT\_BOOL SSR\_EnableUser(LONG dwMachineNumber, BSTR dwEnrollNumber, VARIANT\_BOOL bFlag)

To set whether a user account is available.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| bFlag | BOOL | [in] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The bFlag parameter is a flag that indicates whether a user account is enabled. The value True indicates that the user account is enabled and False indicates that the user account is disabled.

Note

Applicable to TFT and IFACE devices

#### SetUserInfoEx

VARIANT\_BOOL SetUserInfoEx(LONG dwMachineNumber, LONG dwEnrollNumber, LONG VerifyStyle, BYTE\* Reserved)

To upload the user verification mode or group verification mode.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| VerifyStyle | LONG | [in] | Verification mode |
| Reserved | BYTE\* | [in] | Reserved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. This function is valid only on machines with multiple verification modes.

2. On a monochrome machine, the VerifyStyle parameter specifies the verification mode, and 15 verification modes are available. For details about the meanings of the values, see the description of the GetGeneralLogData parameter. If a group verification mode is used, the value of VerifyStyle ranges from 129 to 124, indicating group 1 to group 5 respectively.

3. On a color-screen machine, the VerifyStyle parameter specifies the verification mode. On the color-screen access control fingerprint machine, the value 0 indicates group verification. 128(FP/PW/RF), 129(FP), 130(PIN), 131(PW), 132(RF), 133(FP&RF), 134(FP/PW), 135(FP/RF), 136(PW/RF), 137(PIN&FP), 138(FP&PW), 139(PW&RF), 140(FP&PW&RF), 141(PIN&FP&PW), 142(FP&RF/PIN).

Note

Applicable to B&W, TFT, and IFACE devices

#### GetAllUserID

VARIANT\_BOOL GetAllUserID(LONG dwMachineNumber, LONG\* dwEnrollNumber, LONG\* dwEMachineNumber, LONG\* dwBackupNumber, LONG\* dwMachinePrivilege, LONG\*  dwEnable)

To get all user information. Before executing this function, invoke the function ReadAllUserID to read all user information to the memory. Each time GetAllUserID is executed, the pointer moves to the next user information record. After all user information is read, the function returns False. This function differs from GetAllUserInfo in that the GetAllUserInfo function can obtain user names and passwords.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG\* | [out] | User ID |
| dwEMachineNumber | LONG\* | [out] | Invalid parameter |
| dwBackupNumber | LONG\* | [out] | Comment |
| dwMachinePrivilege | LONG\* | [out] | User privilege |
| dwEnable | LONG\* | [out] | Whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The value of dwEMachineNumber is always 0.

2. The dwMachinePrivilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

3. The dwEnable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to B&W

#### GetAllUserInfo

VARIANT\_BOOL GetAllUserInfo(LONG dwMachineNumber, LONG \* dwEnrollNumber, BSTR\* Name, BSTR\* Password, LONG\* Privilege, VARIANT\_BOOL \* Enabled)

To get all user information. Before executing this function, invoke the function ReadAllUserID to read all user information to the memory. Each time GetAllUserInfo is executed, the pointer moves to the next user information record. After all user information is read, the function returns False. The GetAllUserInfo function differs from GetAllUserID in that it can obtain more information.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [out] | Machine ID |
| dwEnrollNumber | LONG\* | [out] | User ID |
| Name | BSTR\* | [out] | User name |
| Name | BSTR\* | [out] | User password |
| Privilege | LONG\* | [out] | User privilege |
| Enabled | BOOL\* | [out] | Whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The dwMachinePrivilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

2. The dwEnable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to B&W

#### GetUserInfoEx

VARIANT\_BOOL GetUserInfoEx(LONG dwMachineNumber, LONG dwEnrollNumber, LONG\* VerifyStyle, BYTE\* Reserved)

To obtain the user verification mode.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| VerifyStyle | LONG\* | [out] | The value is the user verification mode described by dwEnrollNumber. |
| Reserved | BYTE\* | [out] | Reserved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. This function is valid only on machines with multiple verification modes.

2. On a monochrome machine, the VerifyStyle parameter specifies the verification mode, and 15 verification modes are available. For details about the meanings of the values, see the description of the GetGeneralLogData parameter. If a group verification mode is used, the value of VerifyStyle ranges from 129 to 124, indicating group 1 to group 5 respectively.

3. On a color-screen machine, the VerifyStyle parameter specifies the verification mode. On the color-screen access control fingerprint machine, the value 0 indicates group verification. 128(FP/PW/RF), 129(FP), 130(PIN), 131(PW), 132(RF), 133(FP&RF), 134(FP/PW), 135(FP/RF), 136(PW/RF), 137(PIN&FP), 138(FP&PW), 139(PW&RF), 140(FP&PW&RF), 141(PIN&FP&PW), 142(FP&RF/PIN).

Note

Applicable to B&W, TFT, and IFACE devices

#### DeleteUserInfoEx

VARIANT\_BOOL DeleteUserInfoEx(LONG dwMachineNumber, LONG dwEnrollNumber)

To delete the multiple verification modes set by a specified user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function is valid only on machines with multiple verification modes.

Note

Applicable to B&W, TFT, and IFACE devices

#### SSR\_GetAllUserInfo

VARIANT\_BOOL **SSR\_**GetAllUserInfo(LONG dwMachineNumber, **BSTR**\* dwEnrollNumber, BSTR\* Name, BSTR\* Password, LONG\* Privilege, VARIANT\_BOOL\* Enabled)

To get all user information. Before executing this function, invoke the function ReadAllUserID to read all user information to the memory. Each time SSR\_GetAllUserInfo is executed, the pointer moves to the next user information record. After all user information is read, the function returns False.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR\* | [out] | User ID |
| Name | BSTR\* | [out] | User name |
| Name | BSTR\* | [out] | User password |
| Privilege | LONG\* | [out] | User privilege |
| Enabled | BOOL\* | [out] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

2. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to TFT and IFACE devices

#### GetUserInfo

VARIANT\_BOOL GetUserInfo(LONG dwMachineNumber, LONG dwEnrollNumber, BSTR \* Name, BSTR \* Password, LONG \* Privilege, VARIANT\_BOOL \* Enabled)

To get information about a specified user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| Name | BSTR\* | [out] | Returned user name |
| Password | BSTR\* | [out] | User password |
| Privilege | LONG\* | [out] | User privilege |
| Enabled | BOOL\* | [out] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Password indicates the returned user password. If this parameter is left blank, the user does not use a password on the machine.

2. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

3. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to B&W

#### GetUserInfoByPIN2

VARIANT\_BOOL GetUserInfoByPIN2(LONG dwMachineNumber, BSTR\* Name, BSTR\* Password, LONG\* Privilege, VARIANT\_BOOL\* Enabled)

To obtain user information based on the current attribute value pin2.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Name | BSTR\* | [out] | User name |
| Password | BSTR\* | [out] | User password |
| Privilege | LONG\* | [out] | User privilege |
| Enabled | BOOL\* | [out] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

2. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to B&W

#### GetUserInfoByCard

VARIANT\_BOOL GetUserInfoByCard(LONG dwMachineNumber, BSTR\* Name, BSTR\* Password, LONG\* Privilege, VARIANT\_BOOL\* Enabled)

To obtain user information based on the current attribute value CardNumber.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Name | BSTR\* | [out] | User name |
| Password | BSTR\* | [out] | User password |
| Privilege | LONG\* | [out] | User privilege |
| Enabled | BOOL\* | [out] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

2. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to B&W

#### GetUserIDByPIN2

VARIANT\_BOOL GetUserIDByPIN2(LONG PIN2, LONG \* UserID)

To obtain the user ID based on pin2.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| PIN2 | LONG | [in] | Pin2 value |
| UserID | LONG\* | [out] | User ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### GetPIN2

VARIANT\_BOOL GetPIN2(LONG UserID, LONG\* PIN2)

To obtain the pin2 value based on the user ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| UserID | LONG | [in] | User ID |
| PIN2 | LONG\* | [out] | Pin2 value |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### SSR\_GetUserInfo

VARIANT\_BOOL SSR\_GetUserInfo(LONG dwMachineNumber, BSTR dwEnrollNumber, BSTR\* Name, BSTR\* Password, LONG\* Privilege, VARIANT\_BOOL\* Enabled)

To obtain information about a specified user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| Name | BSTR\* | [out] | The value is the user name described by dwEnrollNumber. |
| Password | BSTR\* | [out] | The value is the user password described by dwEnrollNumber. |
| Privilege | LONG\* | [out] | The value is the user privilege described by dwEnrollNumber. |
| Enabled | BOOL\* | [out] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

2. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to TFT and IFACE devices

#### SetUserInfo

VARIANT\_BOOL SetUserInfo(LONG dwMachineNumber, LONG dwEnrollNumber, BSTR Name, BSTR Password, LONG Privilege, VARIANT\_BOOL Enabled)

To set information about a user. If the user does not exist on the machine, the user will be created.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| Name | BSTR | [in] | User name to be set |
| Password | BSTR | [in] | User password to be set. If this parameter is left blank, the password of the user will be cleared on the machine. |
| Privilege | LONG | [in] | User privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator. |
| Enabled | BOOL | [in] | Flag that indicates whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Password parameter specifies the password to be set. If this parameter is left blank, the password of the user will be cleared on the machine.

2. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

3. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to B&W

#### SSR\_SetUserInfo

VARIANT\_BOOL SetUserInfo(LONG dwMachineNumber, LONG dwEnrollNumber, BSTR Name, BSTR Password, LONG Privilege, VARIANT\_BOOL Enabled)

To set information about a user. If the user does not exist on the machine, the user will be created.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| Name | BSTR | [in] | User name |
| Password | BSTR | [in] | User password |
| Privilege | LONG | [in] | User privilege |
| Enabled | BOOL | [in] | Flag that indicates whether a user account is enabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Privilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

2. The Enable parameter specifies whether a user account is enabled. The value 1 indicates that the user account is enabled and 0 indicates that the user account is disabled.

Note

Applicable to TFT and IFACE devices

#### ModifyPrivilege

VARIANT\_BOOL ModifyPrivilege(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber, LONG dwMachinePrivilege)

To modify user privilege.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwEMachineNumber | LONG | [in] | Invalid parameter. It is meaningless. |
| dwBackupNumber | LONG | [in] | Invalid parameter. It is meaningless. |
| dwMachinePrivilege | LONG | [in] | User privilege to be set |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The dwMachinePrivilege parameter specifies the user privilege. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

Note

Applicable to B&W

### Registration Data Functions (Including Both User Information and Fingerprint)

#### GetEnrollData

VARIANT\_BOOL GetEnrollData(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber, LONG\* dwMachinePrivilege, LONG\* dwEnrollData, LONG\* dwPassWord)

To obtain registration data (fingerprint template and part of the user information) based on the user ID and corresponding index.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | Machine ID |
| dwEMachineNumber | LONG | [in] | User ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |
| dwMachinePrivilege | LONG\* | [out] | User privilege |
| dwEnrollData | LONG\* | [out] | Fingerprint template |
| dwPassWord | LONG\* | [out] | Password |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The values of dwMachineNumber and dwEMachineNumber must be the same.

2. The fingerprint index ranges from 0 to 9. If the fingerprint template is obtained successfully, the password also is obtained. The index 10 indicates obtaining only the password.

3. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

Note

Applicable to B&W

#### SetEnrollData

VARIANT\_BOOL **SetEnrollData**(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber, LONG dwMachinePrivilege, LONG\* dwEnrollData, LONG dwPassWord)

To set registration data (fingerprint template and part of the user information)

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | Machine ID |
| dwEMachineNumber | LONG | [in] | User ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |
| dwMachinePrivilege | LONG | [in] | User privilege |
| dwEnrollData | LONG\* | [in] | Fingerprint template to be uploaded |
| dwPassWord | LONG | [in] | User password |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The values of dwMachineNumber and dwEMachineNumber must be the same.

2. The fingerprint index ranges from 0 to 9. The index 10 indicates setting the user password.

3. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

Note

Applicable to B&W

#### DeleteEnrollData

VARIANT\_BOOL DeleteEnrollData(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber)

To delete registration data.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwEMachineNumber | LONG | [in] | Machine ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The values of dwMachineNumber and dwEMachineNumber must be the same.

2. The dwBackupNumber parameter specifies the fingerprint index. The meanings are described as follows:  
The index range is 0-9. The machine will also check whether a user has other fingerprints and passwords. If no, the machine will delete the user. The index 10 indicates deleting the password. The machine will also check whether the user has fingerprint data. If no, the machine will delete the user. The index 11 indicates deleting all fingerprint data of the user, and 12 indicates deleting the user (including the fingerprints, card number and password).

Note

Applicable to B&W

#### SSR\_DeleteEnrollData

VARIANT\_BOOL SSR\_DeleteEnrollData(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwBackupNumber)

To delete registration data.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The dwBackupNumber parameter specifies the fingerprint index. The meanings are described as follows:  
The index range is 0-9. The machine will also check whether a user has other fingerprints and passwords. If no, the machine will delete the user. The index 10 indicates deleting the password. The machine will also check whether the user has fingerprint data. If no, the machine will delete the user. The index 11 indicates deleting all fingerprint data of the user, and 12 indicates deleting the user (including the fingerprints, card number and password).

Note

Applicable to TFT and IFACE devices

#### SSR\_DeleteEnrollDataExt

VARIANT\_BOOL SSR\_DeleteEnrollDataExt(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwBackupNumber)

To delete registration data.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The dwBackupNumber parameter specifies the fingerprint index. The meanings are described as follows:  
The index range is 0-9. The machine will also check whether a user has other fingerprints and passwords. If no, the machine will delete the user. The index 10 indicates deleting the password. The machine will also check whether the user has fingerprint data. If no, the machine will delete the user. The indexes 11 and 13 indicate deleting all fingerprint data of the user. The index 12 indicates deleting the user, including the fingerprints, card number and password.

2. This function differs from SSR\_DeleteEnrollData in that it can delete all fingerprint data by using parameter 13, and therefore this function has higher efficiency.

Note

Applicable to TFT and IFACE devices

#### GetEnrollDataStr

VARIANT\_BOOL GetEnrollDataStr(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber, LONG\* dwMachinePrivilege, BSTR\* dwEnrollData, LONG\* dwPassWord)

To obtain registration data (fingerprint template and part of the user information) based on the user ID and corresponding index. This function differs from GetEnrollData only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | Machine ID |
| dwEMachineNumber | LONG | [in] | User ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |
| dwMachinePrivilege | LONG\* | [out] | User privilege |
| dwEnrollData | BSTR\* | [out] | Fingerprint template to be uploaded |
| dwPassWord | LONG\* | [out] | User password |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The values of dwMachineNumber and dwEMachineNumber must be the same.

2. The fingerprint index ranges from 0 to 9. If the fingerprint template is obtained successfully, the password also is obtained. The index 10 indicates obtaining only the password.

3. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

Note

Applicable to B&W

#### SetEnrollDataStr

VARIANT\_BOOL SetEnrollDataStr(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwEMachineNumber, LONG dwBackupNumber, LONG dwMachinePrivilege, BSTR dwEnrollData, LONG dwPassWord)

To set registration data (fingerprint template and part of the user information)

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID. The two values must be the same. |
| dwEnrollNumber | LONG | [in] | Machine ID. The two values must be the same. |
| dwEMachineNumber | LONG | [in] | User ID |
| dwBackupNumber | LONG | [in] | Fingerprint index |
| dwMachinePrivilege | LONG | [in] | User privilege |
| dwEnrollData | BSTR | [in] | Fingerprint template to be uploaded |
| dwPassWord | LONG | [in] | User password |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The values of dwMachineNumber and dwEMachineNumber must be the same.

2. The fingerprint index ranges from 0 to 9. If the fingerprint template is obtained successfully, the password also is obtained. The index 10 indicates obtaining only the password.

3. The value 0 indicates common user, 1 registrar, 2 administrator, and 3 super administrator.

4. The dwEnrollData parameter specifies the uploaded fingerprint template, which is a character string.

Note

Applicable to B&W

### Fingerprint Template Functions

#### ReadAllTemplate

VARIANT\_BOOL ReadAllTemplate(LONG dwMachineNumber)

To read all fingerprint templates on the machine to the memory of the PC. This function reads all fingerprint data to the memory at a time. Compared with the function that reads data records from the machine one by one, this function has higher efficiency.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### SSR\_GetUserTmp

VARIANT\_BOOL GetUserTmp(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex, BYTE\* TmpData, LONG\* TmpLength)

To obtain a fingerprint template in binary format. This function differs from SSR\_GetUserTmpStr only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index, ranging from 0 to 9 |
| TmpData | BYTE\* | [in] | Fingerprint template data |
| TmpLength | LONG\* | [in] | Length of the fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### SSR\_GetUserTmpStr

VARIANT\_BOOL GetUserTmpStr(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex, BSTR\* TmpData, LONG\* TmpLength)

To obtain a fingerprint template in character string format. This function differs from SSR\_GetUserTmp only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index, ranging from 0 to 9 |
| TmpData | BSTR\* | [out] | Fingerprint template data |
| TmpLength | LONG\* | [out] | Length of the fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### SSR\_SetUserTmp

VARIANT\_BOOL SetUserTmp(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex, BYTE\* TmpData)

To upload a fingerprint template in binary format. This function differs from SSR\_SetUserTmpStr only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index, ranging from 0 to 9 |
| TmpData | BYTE\* | [in] | Fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### SSR\_SetUserTmpStr

VARIANT\_BOOL SetUserTmpStr(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex, BSTR TmpData)

To upload a fingerprint template in character string format. This function differs from SSR\_SetUserTmp only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| LONG dwFingerIndex | LONG | [in] | Fingerprint index, ranging from 0 to 9 |
| TmpData | BSTR | [in] | Fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### DelUserTmp

VARIANT\_BOOL DelUserTmp(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex)

To delete a specified fingerprint template.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### SSR\_DelUserTmp

VARIANT\_BOOL SSR\_DelUserTmp(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex)

To delete a fingerprint template.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index, ranging from 0 to 9 |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT and IFACE devices

#### SSR\_SetUserTmpExt

VARIANT\_BOOL SSR\_SetUserTmpExt(LONG dwMachineNumber, LONG IsDeleted, BSTR dwEnrollNumber, LONG dwFingerIndex, BYTE\* TmpData)

To upload a fingerprint template. This function is an enhanced version of SSR\_SetUserTmp.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| IsDeleted | LONG | [in] | Deletion flag |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index, ranging from 0 to 9 |
| TmpData | BYTE\* | [in] | Fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

IsDeleted is a deletion flag. If a fingerprint with the specified index already exists on the machine when you upload a fingerprint template, this parameter specifies whether to overwrite the original fingerprint. The value 1 indicates overwriting the original fingerprint and 0 indicates not overwriting the original fingerprint.

Note

Applicable to TFT and IFACE devices

#### SSR\_DelUserTmpExt

VARIANT\_BOOL SSR\_DelUserTmpExt(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex)

Deletes the specified fingerprint template for the specified user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint template ID of the specified user |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT devices

#### SetUserTmp

VARIANT\_BOOL SetUserTmp(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex, BYTE\* TmpData)

To upload a fingerprint template in binary format. This function differs from SetUserTmpStr only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| TmpData | BYTE\* | [in] | Fingerprint template data |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The user must already exist on the machine. If the index number of a user has been registered, the fingerprint template will be overwritten.

Note

Applicable to B&W

#### SetUserTmpStr

VARIANT\_BOOL SetUserTmpStr(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex, BSTR TmpData)

To obtain a fingerprint template in character string format. This function differs from SetUserTmp only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| TmpData | BSTR | [in] | Fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The user must already exist on the machine. If the index number of a user has been registered, the fingerprint template will be overwritten.

Note

Applicable to B&W

#### SetUserTmpEx

VARIANT\_BOOL SetUserTmpEx(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex, LONG Flag, BYTE\* TmpData)

To upload fingerprint template ZKFinger 10.0 in binary format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| Flag | LONG | [in] | Flag that indicates whether the fingerprint template is valid or a duress fingerprint |
| TmpData | BYTE\* | [in] | Fingerprint template data |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The machine firmware must support the duress fingerprint function. That is, the internal version of the firmware must be Ver6.60 or later.

2. The Flag parameter specifies whether the fingerprint template is valid or a duress fingerprint. The value 0 indicates that the fingerprint template is invalid, 1 indicates that the fingerprint template is valid, and 3 indicates that the fingerprint template is a duress fingerprint.

Note

Applicable to B&W, TFT, and IFACE devices

#### SetUserTmpExStr

VARIANT\_BOOL SetUserTmpExStr(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex, LONG Flag, BSTR TmpData)

To upload fingerprint template ZKFinger 10.0 in character string format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| Flag | LONG | [in] | Flag that indicates whether the fingerprint template is valid or a duress fingerprint |
| TmpData | BSTR | [in] | Fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The machine firmware must support the duress fingerprint function. That is, the internal version of the firmware must be Ver6.60 or later.

2. The Flag parameter specifies whether the fingerprint template is valid or a duress fingerprint. The value 0 indicates that the fingerprint template is invalid, 1 indicates that the fingerprint template is valid, and 3 indicates that the fingerprint template is a duress fingerprint.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetUserTmp

VARIANT\_BOOL GetUserTmp(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex, BYTE\* TmpData, LONG\* TmpLength )

To obtain a fingerprint template in binary format. This function differs from GetUserTmpStr only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| TmpData | BYTE\* | [out] | Fingerprint template data |
| TmpLength | LONG\* | [out] | Length of the fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### GetUserTmpStr

VARIANT\_BOOL GetUserTmpStr(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex, BSTR\* TmpData, LONG\* TmpLength)

To obtain a fingerprint template in character string format. This function differs from GetUserTmp only in the fingerprint template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| TmpData | BSTR\* | [out] | Fingerprint template data |
| TmpLength | LONG\* | [out] | Length of the fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### GetUserTmpEx

VARIANT\_BOOL GetUserTmpEx(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex, LONG\* Flag, BYTE\* TmpData, LONG\* TmpLength)

To obtain fingerprint template ZKFinger 10.0 in binary format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| Flag | LONG\* | [out] | Flag that indicates whether the fingerprint template is valid or a duress fingerprint |
| TmpData | BYTE\* | [out] | Fingerprint template |
| TmpLength | LONG\* | [out] | Length of the fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The machine firmware must support the duress fingerprint function. That is, the internal version of the firmware must be Ver6.60 or later.

The Flag parameter specifies whether the fingerprint template is valid or a duress fingerprint. The value 0 indicates that the fingerprint template is invalid, 1 indicates that the fingerprint template is valid, and 3 indicates that the fingerprint template is a duress fingerprint.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetUserTmpExStr

**VARIANT\_BOOL GetUserTmpExStr(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFingerIndex, LONG\* Flag, BSTR\* TmpData, LONG\* TmpLength)**

To obtain fingerprint template ZKFinger 10.0 in character string format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| dwFingerIndex | LONG | [in] | Fingerprint index |
| Flag | LONG\* | [out] | Flag that indicates whether the fingerprint template is valid or a duress fingerprint |
| TmpData | BSTR\* | [out] | Fingerprint template |
| TmpLength | LONG\* | [out] | Length of the fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The machine firmware must support the duress fingerprint function. That is, the internal version of the firmware must be Ver6.60 or later.

2. The Flag parameter specifies whether the fingerprint template is valid or a duress fingerprint. The value 0 indicates that the fingerprint template is invalid, 1 indicates that the fingerprint template is valid, and 3 indicates that the fingerprint template is a duress fingerprint.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetFPTempLength

VARIANT\_BOOL GetFPTempLength(BYTE\* dwEnrollData, LONG\* Len)

To calculate length of a specified fingerprint template.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwEnrollData | BYTE\* | [in] | Pointer that points to the fingerprint template |
| Len | LONG\* | [out] | The value is the fingerprint template length described by dwEnrollData. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### GetFPTempLengthStr

VARIANT\_BOOL GetFPTempLengthStr(BSTR dwEnrollData, LONG\* Len)

To calculate length of a specified fingerprint template.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwEnrollData | BSTR | [in] | Fingerprint template in character string format |
| Len | LONG\* | [out] | The value is the fingerprint template length described by dwEnrollData. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### FPTempConvert

VARIANT\_BOOL FPTempConvert(BYTE\* TmpData1, BYTE\* TmpData2, LONG\* Size)

To convert an offline fingerprint template into a BIOKEY fingerprint template. This function differs from FPTempConvertStr only in the data format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| TmpData1 | BYTE\* | [in] | Offline fingerprint template to be converted |
| TmpData2 | BYTE\* | [out] | The value is the BIOKEY fingerprint template after conversion. |
| Size | LONG\* | [out] | The value is the size of the BIOKEY fingerprint template after conversion. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### FPTempConvertStr

VARIANT\_BOOL FPTempConvertStr(BSTR TmpData1, BSTR\* TmpData2, LONG\* Size)

To convert an offline fingerprint template into a BIOKEY fingerprint template in character string format. This function differs from FPTempConvert only in the data format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| TmpData1 | BSTR | [in] | Offline fingerprint template to be converted |
| TmpData2 | BSTR\* | [out] | The value is the BIOKEY fingerprint template after conversion. |
| Size | LONG\* | [out] | The value is the size of the BIOKEY fingerprint template after conversion. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### FPTempConvertNew

VARIANT\_BOOL FPTempConvertNew(BYTE\* TmpData1, BYTE\* TmpData2, LONG\* Size)

To convert a BIOKEY fingerprint template into an offline fingerprint template. This function differs from FPTempConvertNewStr only in the data format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| TmpData1 | BYTE\* | [in] | Offline fingerprint template to be converted |
| TmpData2 | BYTE\* | [out] | The value is the offline fingerprint template after conversion. |
| Size | LONG\* | [out] | The value is the size of the offline fingerprint template after conversion. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### FPTempConvertNewStr

VARIANT\_BOOL FPTempConvertNewStr(BSTR TmpData1, BSTR\* TmpData2, LONG\* Size)

To convert a BIOKEY fingerprint template into an offline fingerprint template in character string format. This function differs from FPTempConvertNew only in the data format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| TmpData1 | BSTR | [in] | Offline fingerprint template to be converted |
| TmpData2 | BSTR\* | [out] | The value is the offline fingerprint template after conversion. |
| Size | LONG\* | [out] | The value is the size of the offline fingerprint template after conversion. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### Face Template Functions

#### SetUserFace

VARIANT\_BOOL SetUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BYTE\* TmpData, LONG TmpLength)

To upload a face template. This function differs from SetUserFaceStr only in the face template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID (not more than 24 digits) |
| dwFaceIndex | LONG | [in] | Face index |
| TmpData | BYTE\* | [in] | Face template |
| TmpLength | LONG | [in] | Length of the face template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of dwFaceIndex is always 50, which indicates uploading all face templates of a user.

Note

Applicable to IFACE

#### GetUserFace

VARIANT\_BOOL GetUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BYTE\* TmpData, LONG\* TmpLength)

To download a face template. This function differs from GetUserFaceStr only in the face template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID (not more than 24 digits) |
| dwFaceIndex | LONG | [in] | Face index |
| TmpData | BYTE\* | [out] | Face template |
| TmpLength | LONG\* | [out] | Length of the face template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of dwFaceIndex is always 50, which indicates downloading all face templates of a user.

Note

Applicable to IFACE

#### DelUserFace

VARIANT\_BOOL DelUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex)

To delete a face template.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID (not more than 24 digits) |
| dwFaceIndex | LONG | [in] | Face index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of dwFaceIndex is always 50, which indicates downloading all face templates of a user.

Note

Applicable to IFACE

#### GetUserFaceStr

VARIANT\_BOOL GetUserFaceStr(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BSTR\* TmpData, LONG\* TmpLength)

To download a face template. This function differs from GetUserFace in that it returns a face template in character string format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID (not more than 24 digits) |
| dwFaceIndex | LONG | [in] | Face index |
| TmpData | BSTR\* | [out] | Face template |
| TmpLength | LONG\* | [out] | Length of the face template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of dwFaceIndex is always 50, which indicates downloading all face templates of a user.

Note

Applicable to IFACE

#### SetUserFaceStr

VARIANT\_BOOL SetUserFaceStr(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BSTR TmpData, LONG TmpLength)

To upload a face template. This function differs from SetUserFace only in the face template format.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID (not more than 24 digits) |
| dwFaceIndex | LONG | [in] | Face index |
| TmpData | BSTR | [in] | Face template |
| TmpLength | LONG | [in] | Length of the face template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of dwFaceIndex is always 50, which indicates uploading all face templates of a user.

Note

Applicable to IFACE

### User Verify Functions

#### SetUserVerifyStyle

VARIANT\_BOOL SetUserVerifyStyle(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG VerifyStyle, BYTE\* Reserved)

To set the user verification mode.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| VerifyStyle | LONG | [in] | Verification mode |
| Reserved | BSTR\* | [in] | Reserved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The VerifyStyle parameter specifies the verification mode. The values are described as follows:  
Group Verify = 0   
FP/PW/RF = 128   
FP = 129   
PIN = 130   
PW = 131   
RF = 132   
FP/PW = 133   
FP/RF = 134   
PW/RF = 135   
PIN&FP = 136   
FP&PW = 137   
FP&RF = 138   
PW&RF = 139   
FP&PW&RF = 140   
PIN&FP&PW = 141   
FP&RF/PIN = 142

2. The Reserved parameter is reserved and not used at present.

Note

This interface is applicable to the new architecture firmware.

#### GetUserVerifyStyle

VARIANT\_BOOL GetUserVerifyStyle(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG\* VerifyStyle, BYTE\* Reserved)

To obtain the user verification mode.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| VerifyStyle | LONG\* | [out] | Verification mode |
| Reserved | BSTR\* | [out] | Reserved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The VerifyStyle parameter specifies the verification mode. The values are described as follows:  
Group Verify = 0   
FP/PW/RF = 128   
FP = 129   
PIN = 130   
PW = 131   
RF = 132   
FP/PW = 133   
FP/RF = 134   
PW/RF = 135   
PIN&FP = 136   
FP&PW = 137   
FP&RF = 138   
PW&RF = 139   
FP&PW&RF = 140   
PIN&FP&PW = 141   
FP&RF/PIN = 142

2. The Reserved parameter is reserved and not used at present.

Note

This interface is applicable to the new architecture firmware.

### Shortcut Keys Functions

#### SSR\_SetShortkey

VARIANT\_BOOL SSR\_SetShortkey(LONG ShortKeyID, LONG ShortKeyFun, LONG StateCode, BSTR StateName, LONG StateAutoChange, BSTR StateAutoChangeTime)

To set a functional key. It is similar to the functional key definition function on the color-screen machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ShortKeyID | LONG | [in] | Key ID |
| ShortKeyFun | LONG | [in] | Function of the key |
| StateCode | LONG | [in] | State code of the status key |
| StateName | BSTR | [in] | Name of the status key |
| StateAutoChange | LONG | [in] | Auxiliary |
| StateAutoChangeTime | BSTR | [in] | Auxiliary |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. ShortKeyID: specifies the ID of the key. The mapping is as follows: F1 ? 1, F2 ? 2, F3 ? 3 ...

2. ShortKeyFun: function of the specified key. The value 0 indicates that the function of the key is not defined, 1 indicates that the specified key is a status key, 2 indicates the work code of the key, and 3 indicates viewing the short message. Note: the value of ShortKeyFun will influence settings of the following four parameters. Please refer to the below instructions.

3. StateCode: To set state code of specified status key. If the specified key is not a status key (namely, the value of ShortKeyFun is not 1), the value of StateCode will be ignored. If the specified key is a status key (ShortKeyFun=1), the state code of the specified status key is determined by the StateCode value, which ranges from 0 to 255.The state code of status key cannot be repeatedly, if the state code of different status key is set as the same, invoking of StateCode will fail. Such as F2 is a status key and the state code is 2; while you invoke StateCode Fun to set F3 as a status key and set its state code as 2, the invoking will fail.

4. StateName: To set name of status key. If the specified key is not a status key (namely, the value of ShortKeyFun is not 1), the value of StateName will be ignored. If the specified key is a status key (ShortKeyFun=1), the name of specified status key will be determined by the StateName value. At most, 18 characters are supported.

5. StateAutoChange: If the specified key is not a status key (namely, the value of ShortKeyFun is not 1), the value of StateAutoChange will be ignored. If the specified key is a status key (ShortKeyFun=1), the value of StateAutoChange indicates whether the status key automatically changes. 0: disable, 1: enable.

6. StateAutoChangeTime: If the specified key is not a status key (namely, the value of ShortKeyFun is not 1), the value of StateAutoChangeTime will be ignored. If the specified key is a status key (ShortKeyFun=1), the automatic change time of the status key is set by the return value of StateAutoChangeTime. Requirements as below:

1) "08:30;09:00;08:00;12:00;11:12;00:00;00:00;".

2) Hour and minute are separated with ":",Dates are separated with ";",space is not allowed between them.

3) Everyday's automatic change time shall be specified(one whole week is a cycle), reaches which the attendance state will change to specified state automatically (the name of automatic change state is decided by StateName, value is defined by the return value of StateCode). If someday needs not to change attendance state automatically, set hour and minute of the StateAutoChangeTime as zero.

Note

Applicable to TFT

#### SSR\_GetShortkey

VARIANT\_BOOL SSR\_GetShortkey(LONG ShortKeyID, LONG\* ShortKeyFun, LONG\* StateCode, BSTR\* StateName, LONG\* AutoChange, BSTR\* AutoChangeTime)

To query the settings of a functional key.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ShortKeyID | LONG | [in] | Key ID |
| ShortKeyFun | LONG\* | [out] | Function of the key |
| StateCode | LONG\* | [out] | State code of the status key |
| StateName | BSTR\* | [out] | Name of the status key |
| AutoChange | LONG\* | [out] | Whether the status key automatically changes |
| AutoChangeTime | BSTR\* | [out] | Automatic change time of the status key |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. ShortKeyID: specifies the ID of the key. The mapping is as follows: F1 ? 1, F2 ? 2, F3 ? 3 ...

2. ShortKeyFun: function of the specified key. The value 0 indicates that the function of the key is not defined, 1 indicates that the specified key is a status key, 2 indicates the work code of the key, and 3 indicates viewing the short message.

3. StateCode: If the specified key is a status key (ShortKeyFun=1), the state code of the status key is returned. Otherwise, 0 is returned.

4. StateName: If the specified key is a status key (ShortKeyFun=1), the name of the status key is returned. Otherwise, a blank character string is returned.

5. AutoChange: If the specified key is a status key (ShortKeyFun=1), the value of this parameter indicates whether the status key automatically changes. Otherwise, 0 is returned.

6. AutoChangeTime: If the specified key is a status key (ShortKeyFun=1), the automatic change time of the status key is returned, in the format of a character string. Otherwise, a blank character string is returned.

Note

Applicable to TFT

#### EnableCustomizeAttState

VARIANT\_BOOL EnableCustomizeAttState(LONG dwMachineNumber, LONG StateID, LONG Enable)

To specify whether to enable a customized attendance status value.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| StateID | LONG | [in] | Attendance status value to be defined |
| Enable | LONG | [in] | Whether to enable the attendance status definition function for the attendance status value specified by StateID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. This function is a customization function. To use this function, the extension function must be enabled for the machine and the machine must support the attendance status definition function.

2. Attendance status definition function: The mapping between the attendance status values and the states are as follows:  
If this function is enabled, you can invoke SetCustomizeAttState to change the status value of a state: 0-Check-In 1-Check-Out 2-Break-Out 3-Break-In 4-OT-In 5-OT-Out  
For example, EnableCustomizeAttState(1,0,1)// indicates enabling the customized state of 0 (check-in)  
SetCustomizeAttState(1,0,8)// indicates setting the customized status value to 8 for the status value 0 (check-in)  
If a user chooses check-in on the machine and passes verification, the saved attendance status value is 8.

Note

Applicable to B&W

#### SetCustomizeAttState

VARIANT\_BOOL SetCustomizeAttState(LONG dwMachineNumber, LONG StateID, LONG NewStatel)

To set a customized attendance status value based on the original attendance status value.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| StateID | LONG | [in] | Original status value |
| NewStatel | LONG | [in] | New status value |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function is a customization function. To use this function, the extension function must be enabled for the machine and the machine must support the attendance status definition function. For details, see the description of EnableCustomizeAttState.

Note

Applicable to B&W

#### DelCustomizeAttState

VARIANT\_BOOL DelCustomizeAttState(LONG dwMachineNumber, LONG StateID)

To delete the customized attendance status value of an original attendance status value.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| StateID | LONG | [in] | Original attendance status value of which the customized attendance value is to be deleted |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function is a customization function. To use this function, the extension function must be enabled for the machine and the machine must support the attendance status definition function. For details, see the description of EnableCustomizeAttState.

Note

Applicable to B&W

#### GetAllSFIDName

VARIANT\_BOOL GetAllSFIDName(LONG dwMachineNumber, BSTR\* ShortkeyIDName, LONG BufferSize1, BSTR\* FunctionIDName, LONG BufferSize2)

To query the shortcut key name and function lists.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| ShortkeyIDName | BSTR\* | [out] | List of shortcut key IDs and names |
| BufferSize1 | LONG | [in] | Size of the buffer for storing the list of shortcut key IDs and names |
| FunctionIDName | BSTR\* | [out] | List of function IDs and names |
| BufferSize2 | LONG | [in] | Size of the buffer for storing the list of function IDs and names |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. ShortkeyIDName: specifies the list of shortcut key IDs and names, in the format of "ID,Key\_Name\r\n1,F1\r\n2,F2...".

2. FunctionIDName: specifies the list of function IDs and names, in the format of "ID,Func\_Name\r\n1,adduser\r\n2,userlist...".

3. The values of BufferSize1 and BufferSize2 are est to 4 kB.

Note

This interface is applicable to the new architecture firmware.

#### GetShortkey

VARIANT\_BOOL GetShortkey(LONG dwMachineNumber, LONG ShortKeyID, BSTR\* ShortKeyName, BSTR\* FunctionName, LONG\* ShortKeyFun, LONG\* StateCode, BSTR\* StateName, BSTR\* Description, LONG\* AutoChange, BSTR\* AutoChangeTime)

To query the settings of a functional key.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| ShortKeyID | LONG | [in] | Key ID |
| ShortKeyName | BSTR\* | [out] | Key name |
| ShortKeyName | BSTR\* | [out] | Function name of the key |
| ShortKeyFun | LONG\* | [out] | Key type |
| StateCode | LONG\* | [out] | State code of the status key |
| StateName | BSTR\* | [out] | Name of the status key |
| Description | BSTR\* | [out] | Description of the status key |
| AutoChange | LONG\* | [out] | Whether the status key automatically changes |
| AutoChangeTime | BSTR\* | [out] | Automatic change time of the status key |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The mapping between ShortKeyName and ShortKeyID is as follows: F1-1, F2 -2, F3 -3 ...

2. ShortKeyFun: function of the specified key. The value 0 indicates a functional key and 1 indicates a status key.

3. ShortKeyName: name of the key specified by ShortKeyID.

4. FunctionName: If the specified key is a functional key (ShortKeyFun=0), the function name is returned.

5. FunctionName: If the specified key is a status key (ShortKeyFun=1), the name of the key is returned. In this case, the value of FunctionName is the same as that of StateName.

6. StateCode: If the specified key is a status key (ShortKeyFun=1), the state code of the status key is returned. Otherwise, an invalid value is returned.

7. StateName: If the specified key is a status key (ShortKeyFun=1), the name of the status key is returned. Otherwise, an invalid value is returned.

8. AutoChange: If the specified key is a status key (ShortKeyFun=1), the value of this parameter indicates whether the status key automatically changes. Otherwise, an invalid value is returned.

9. Description: If the specified key is a status key (ShortKeyFun=1), the description of the status key is returned. Otherwise, an invalid value is returned.

10. AutoChangeTime: If the specified key is a status key (ShortKeyFun=1), the automatic change time of the status key is returned, in the format of a character string. Otherwise, an invalid value is returned.

Note

This interface is applicable to the new architecture firmware.

#### SetShortkey

VARIANT\_BOOL SetShortkey(LONG dwMachineNumber, LONG ShortKeyID, BSTR ShortKeyName, BSTR FunctionName, LONG ShortKeyFun, LONG StateCode, BSTR StateName, BSTR Description, LONG StateAutoChange, BSTR StateAutoChangeTime)

To set a functional key. It is similar to the functional key definition function on the color-screen machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| ShortKeyID | LONG | [in] | Key ID |
| ShortKeyName | BSTR | [in] | Key name |
| FunctionName | BSTR | [in] | Function name of the key |
| ShortKeyFun | LONG | [in] | Key type |
| StateCode | LONG | [in] | State code of the status key |
| StateName | BSTR | [in] | Name of the status key |
| Description | BSTR | [in] | Description of the status key |
| AutoChange | LONG | [in] | Whether the status key automatically changes |
| AutoChangeTime | BSTR | [in] | Automatic change time of the status key |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. When setting ShortKeyName and ShortKeyID, ensure that the mapping between ShortKeyName and ShortKeyID is as follows: F1-1, F2 -2, F3-3 ...

2. ShortKeyFun: function of the specified key. The value 0 indicates a functional key and 1 indicates a status key. Note that the value of this parameter will affect the settings of other four parameters.

3. StateCode: state code of the status key.  
If the specified key is not a functional key, that is, the value of ShortKeyFun is not 1, the value of this parameter will be ignored.  
If the specified key is a status key, that is, the value of ShortKeyFun is 1, the value of this parameter is the state code of the status key, and ranges from 0 to 255. The state values of status keys cannot be duplicate. Otherwise, function invocation will fail. For example, F2 is a status key and its state code is 2. If you invoke this function to set F3 as a status key and set its state code to 2, the function invocation will fail.

4. StateName: name of the status key.  
If the specified key is not a functional key, that is, the value of ShortKeyFun is not 1, the value of this parameter will be ignored.  
If the specified key is a status key, that is, the value of ShortKeyFun is 1, the value of this parameter is the name of the status key, and contains at most 18 characters.

5. Description: description of the status key.  
If the specified key is not a functional key, that is, the value of ShortKeyFun is not 1, the value of this parameter will be ignored.  
If the specified key is a status key, that is, the value of ShortKeyFun is 1, the value of this parameter is the description of the status key.

6. StateAutoChange: whether the status key automatically changes:  
If the specified key is not a functional key, that is, the value of ShortKeyFun is not 1, the value of this parameter will be ignored.  
If the specified key is a status key, that is, the value of ShortKeyFun is 1, the value of this parameter indicates whether the status key automatically changes. The value 0 indicates that the status key automatically changes and 1 indicates that the status key does not automatically change.

7. StateAutoChangeTime:  
If the specified key is not a functional key, that is, the value of ShortKeyFun is not 1, the value of this parameter will be ignored.  
If the specified key is a status key, the value of this parameter is the automatic change time of the status key. The details are as follows: 1. 08:30;09:00;08:00;12:00;11:12;00:00;00:00; 2. The hour is separated from the minute with a colon (:), and days are separated with a semicolon (;), free from spaces. 3. You must specify the automatic change time of each day, with a week as a cycle. After the specified time arrives, the attendance status will change to the state specified by StateName and StateCode. If automatic change is disabled on a day, the hour and minute are both set to 0.

8. If the return value of issued status key is -15001 repeatedly, the return value of description will be -15002 repeatedly.

Note

This interface is applicable to the new architecture firmware.

### Work Code Functions

#### SetWorkCode

VARIANT\_BOOL SetWorkCode(LONG WorkCodeID, LONG AWorkCode)

To define a work code with a specified ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| WorkCodeID | LONG | [in] | Work code ID |
| AWorkCode | LONG | [in] | Value of work code described by WorkCodeID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

A work code value within any range can be input on a monochrome machine. After the work code is defined by using this function, the user can input only the defined work code. For example, if the work code is defined as SetWorkCode(1,345) and SetWorkCode(2,567), the user can input only the work code with the values 345 and 567.

Note

Applicable to B&W

#### GetWorkCode

VARIANT\_BOOL GetWorkCode(LONG WorkCodeID, LONG\* AWorkCode)

To obtain the name of a specified work code ID. For details, see the description of SetWorkCode.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| WorkCodeID | LONG | [in] | Work code ID |
| AWorkCode | LONG\* | [out] | Obtained value of work code described by WorkCodeID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### SSR\_GetWorkCode

VARIANT\_BOOL SSR\_GetWorkCode(LONG AWorkCode, BSTR\* Name)

To obtain the name of a specified work code ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| AWorkCode | LONG | [in] | Work code ID |
| Name | BSTR\* | [out] | Obtained value of work code described by WorkCodeID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### SSR\_SetWorkCode

VARIANT\_BOOL SSR\_SetWorkCode(LONG AWorkCode, BSTR Name)

To set a work code with a specified ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| AWorkCode | LONG | [in] | Work code ID |
| Name | BSTR | [in] | Value of work code described by WorkCodeID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### SSR\_DeleteWorkCode

VARIANT\_BOOL SSR\_DeleteWorkCode(LONG AWorkCode)

To delete a work code with a specified ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| AWorkCode | LONG | [in] | Work code ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### SSR\_ClearWorkCode

VARIANT\_BOOL SSR\_ClearWorkCode()

To delete all user-defined work codes.

Parameters

None

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### DeleteWorkCode

VARIANT\_BOOL DeleteWorkCode(LONG WorkCodeID)

To delete a work code with a specified work code ID. For details, see the description of SetWorkCode.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| WorkCodeID | LONG | [in] | Work code ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### ClearWorkCode

VARIANT\_BOOL ClearWorkCode()

To clear all defined work codes on the machine. For details, see the description of SetWorkCode.

Parameters

None

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### SSR\_GetWorkCodeIDByName

VARIANT\_BOOL SSR\_GetWorkCodeIDByName(LONG dwMachineNumber, BSTR WorkCodeName, LONG\* WorkCodeId)

To query the interface of work code id based on the work code name.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| WorkCodeName | BSTR | [in] | Work code name |
| WorkCodeId | LONG\* | [out] | Work code ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Together with the interface of SSR\_SetWorkCode, to judge whether the issued workname is repeatedly (the same WorkCodeName cannot be issued). When return value of WorkCodeID is greater than zero, the issued workname has existed.

Note

This interface is applicable to the new architecture firmware.

### SMS Functions

#### SetSMS

VARIANT\_BOOL SetSMS(LONG dwMachineNumber, LONG ID, LONG Tag, LONG ValidMinutes, BSTR StartTime, BSTR Content)

To add a short message to the machine. To set the short message for a user, invoke SetUserSMS to assign the short message to the user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| ID | LONG | [in] | Short message ID |
| Tag | LONG | [in] | Short message type |
| ValidMinutes | LONG | [in] | Validity period of the short message |
| StartTime | BSTR | [in] | Effective time of the short message |
| Content | BSTR | [in] | Content of the short message |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. Tag specifies the short message type. The value 253 indicates public short message, 254 personal short message, and 255 reserved short message.

2. ValidMinutes specifies the validity period of the short message. The value ranges from 0 to 65535. The short message will take effect at the time specified by StartTime and will last for a period of time specified by ValidMinutes.

3. StartTime specifies the effective time of the short message, in the format of yyyy-mm-dd hh:mm:ss.

Note

Applicable to B&W and TFT devices

#### SetUserSMS

VARIANT\_BOOL SetUserSMS(LONG dwMachineNumber, LONG dwEnrollNumber, LONG SMSID)

To set the short message of a user. Specifically, this function is used to assign a short message with a specific ID to a user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| SMSID | LONG | [in] | Short message ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### SSR\_SetUserSMS

VARIANT\_BOOL SSR\_SetUserSMS(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG SMSID)

To set the short message of a user. Specifically, this function is used to assign a short message with a specific ID to a user.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| SMSID | LONG | [in] | Short message ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### GetSMS

VARIANT\_BOOL GetSMS(LONG dwMachineNumber, LONG ID, LONG\* Tag, LONG\* ValidMinutes, BSTR\* StartTime, BSTR\* Content)

To obtain details about a short message from the machine based on the short message ID, including the content, effective time, message type, and validity period.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| ID | LONG | [in] | Short message ID |
| Tag | LONG\* | [out] | Short message type |
| ValidMinutes | LONG\* | [out] | Validity period of the short message |
| StartTime | BSTR\* | [out] | Effective time of the short message |
| Content | BSTR\* | [out] | Content of the short message |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. Tag specifies the short message type. The value 253 indicates public short message, 254 personal short message, and 255 reserved short message.

2. ValidMinutes specifies the validity period of the short message. The value ranges from 0 to 65535. The short message will take effect at the time specified by StartTime and will last for a period of time specified by ValidMinutes.

3. StartTime specifies the effective time of the short message, in the format of yyyy-mm-dd hh:mm:ss.

Note

Applicable to B&W and TFT devices

#### DeleteSMS.

VARIANT\_BOOL DeleteSMS(LONG dwMachineNumber, LONG ID)

To delete a short message with a specified ID from the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| ID | LONG | [in] | Short message ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

#### DeleteUserSMS

VARIANT\_BOOL DeleteUserSMS(LONG dwMachineNumber, LONG dwEnrollNumber, LONG SMSID)

To delete the short message with a specified ID for a specified user. Only the mapping relationship between the user and the short message is deleted, and the short message is not deleted.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| SMSID | LONG | [in] | Short message ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### SSR\_DeleteUserSMS

VARIANT\_BOOL SSR\_DeleteUserSMS(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG SMSID)

To delete the short message with a specified ID for a specified user. Only the mapping relationship between the user and the short message is deleted, and the short message is not deleted.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| SMSID | LONG | [in] | Short message ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### ClearUserSMS

VARIANT\_BOOL ClearUserSMS(LONG dwMachineNumber)

To clear all mapping relationships between short messages and users. The short messages are not deleted.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

#### ClearSMS

VARIANT\_BOOL ClearSMS(LONG dwMachineNumber)

To clear all short messages on the machine. All short messages will be deleted from the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

### Holiday Functions

#### SetHoliday

VARIANT\_BOOL SetHoliday(LONG dwMachineNumber, BSTR Holiday)

To set holidays.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Holiday | BSTR | [in] | Holiday to be set |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The Holiday parameter specifies holidays in the format of mmddmmdd. For example, 04140511 indicates a holiday that lasts from April 14 to May 11.

Note

Applicable to B&W

#### GetHoliday

VARIANT\_BOOL GetHoliday(LONG dwMachineNumber, BSTR\* Holiday)

To query the holiday that is set on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Holiday | BSTR\* | [out] | Holiday specified on the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The Holiday parameter specifies holidays in the format of mmddmmdd. For example, 04140511 indicates a holiday that lasts from April 14 to May 11.

Note

Applicable to B&W

#### SSR\_GetHoliday

VARIANT\_BOOL SSR\_GetHoliday(LONG dwMachineNumber, LONG HolidayID, LONG\* BeginMonth, LONG\* BeginDay, LONG\* EndMonth, LONG\* EndDay, LONG\* TimeZoneID)

To query holiday settings on the machine based on the holiday ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| HolidayID | LONG | [in] | Holiday ID |
| BeginMonth | LONG\* | [out] | Start date of the holiday |
| BeginDay | LONG\* | [out] | Start date of the holiday |
| EndMonth | LONG\* | [out] | End date of the holiday |
| EndDay | LONG\* | [out] | End date of the holiday |
| TimeZoneID | LONG\* | [out] | Index of the time segment of the holiday |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT and IFACE devices

#### SSR\_SetHoliday

VARIANT\_BOOL SSR\_SetHoliday(LONG dwMachineNumber, LONG HolidayID, LONG BeginMonth, LONG BeginDay, LONG EndMonth, LONG EndDay, LONG TimeZoneID )

To set holidays.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| HolidayID | LONG | [in] | Holiday ID |
| BeginMonth | LONG | [in] | Start date of the holiday |
| BeginDay | LONG | [in] | Start date of the holiday |
| EndMonth | LONG | [in] | End date of the holiday |
| EndDay | LONG | [in] | Start date of the holiday |
| TimeZoneID | LONG | [in] | Index of the time segment of the holiday |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT and IFACE devices

### DST Functions

#### SetDaylight

VARIANT\_BOOL SetDaylight(LONG dwMachineNumber, LONG Support, BSTR BeginTime, BSTR EndTime)

To set whether to enable DST, and the DST start time and end time.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Support | LONG | [in] | Whether to enable DST |
| BeginTime | BSTR | [in] | DST start time |
| EndTime | BSTR | [in] | DST end time |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Support parameter specifies whether to enable DST. The value 1 indicates enabling DST and 0 indicates disabling DST.

2. The values of BeginTime and EndTime are in the format of mm-dd hh:mm.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetDaylight

VARIANT\_BOOL GetDaylight(LONG dwMachineNumber, LONG\* Support, BSTR\* BeginTime, BSTR\* EndTime)

To query the DST settings on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Support | LONG\* | [out] | Whether to enable DST |
| BeginTime | BSTR\* | [out] | DST start time |
| EndTime | BSTR\* | [out] | DST end time |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The Support parameter specifies whether to enable DST. The value 1 indicates enabling DST and 0 indicates disabling DST.

2. The values of BeginTime and EndTime are in the format of mm-dd hh:mm.

Note

Applicable to B&W, TFT, and IFACE devices

### System Data Management Functions

#### ClearKeeperData

VARIANT\_BOOL ClearKeeperData(LONG dwMachineNumber)

To clear all data on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### ClearData

VARIANT\_BOOL ClearData(LONG dwMachineNumber, LONG DataFlag)

To clear the records specified by DataFlag on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| DataFlag | LONG | [in] | This parameter specifies the type of records to be cleared. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The DataFlag parameter specifies the type of records to be cleared. The value range is 1-5. The meanings are as follows:  
1 Attendance records  
2 Fingerprint template data  
3 None  
4 Operation records  
5 User information  
If the value of this parameter is 5, all users on the machine will be deleted. Note: All fingerprint templates will also be deleted.

Note

Applicable to B&W, TFT, and IFACE devices

#### GetDataFile

VARIANT\_BOOL GetDataFile(LONG dwMachineNumber, LONG DataFlag, BSTR FileName)

To obtain a specified data file from the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| DataFlag | LONG | [in] | Type of the data file to be obtained |
| FileName | BSTR | [in] | Name of the data file that is stored |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The DataFlag parameter specifies the type of records to be cleared. The values are described as follows:  
1 Attendance records  
2 Fingerprint template data  
3 None  
4 Operation records  
5 User information  
6 Short message data file  
7 Data file of short messages and user relationships  
8 Extended user information data file  
9 Work code information data file

Note

Applicable to B&W, TFT, and IFACE devices

#### SendFile

VARIANT\_BOOL SendFile(LONG dwMachineNumber, BSTR FileName)

To send a file to the machine, usually to the directory /mnt/mtdblock/.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Name of the file to be sent |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

If the color-screen machine transfers user pictures or advertisement pictures, name the pictures properly and then the pictures will be automatically saved to the corresponding directory.  
Name format of advertisement pictures: ad\_Number.jpg. The value of number ranges from 1 to 20. For example, ad\_4.jpg.  
Name format of user pictures: User ID.jpg. For example, 1.jpg.

Note

Applicable to B&W, TFT, and IFACE devices

#### ReadFile

VARIANT\_BOOL ReadFile(LONG dwMachineNumber, BSTR FileName, BSTR FilePath)

To read a specified file from the machine, usually under the directory /mnt/mtdblock/.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Name of the file to be read |
| FilePath | BSTR | [in] | Path for saving the file |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### RefreshData

VARIANT\_BOOL RefreshData(LONG dwMachineNumber)

To refresh the data on the machine. This function is typically invoked after user information or a fingerprint is uploaded, so that the modification takes effect immediately.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### User Photo & Attendance Photo

#### UploadUserPhoto

VARIANT\_BOOL UploadUserPhoto(LONG dwMachineNumber, BSTR FileName)

To upload a user picture to the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | User picture name and the absolute path where the picture resides |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The FileName parameter specifies the absolute path where the user picture to be uploaded resides, for example, C:\Users\HP\Desktop\11.jpg. The user picture is named in the format of User ID.jpg.

2. This function can also be implemented by using the SendFile function.

3. You can invoke this function to upload multiple user pictures by traversing through all file names under a specified directory.

Note

This interface is applicable to the new architecture firmware.

#### DownloadUserPhoto

VARIANT\_BOOL DownloadUserPhoto(LONG dwMachineNumber, BSTR FileName, BSTR FilePath)

To download a user picture from the machine to the software.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Name of the user picture, with the file name extension .jpg |
| FilePath | BSTR | [in] | Absolute path where the downloaded user picture is to be saved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The FileName parameter specifies the name of the user picture to be downloaded. The name format is User ID.jpg, for example, 11.jag. The downloaded picture will be saved in JPG format to the specified path.

2. The FilePath parameter specifies the absolute path where the downloaded picture is to be saved, for example, C:\Users\HP\Desktop\.

Note

This interface is applicable to the new architecture firmware.

#### DeleteUserPhoto

VARIANT\_BOOL DeleteUserPhoto(LONG dwMachineNumber, BSTR FileName)

To delete a single user picture or all user pictures on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Name of the user picture to be deleted |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The FileName parameter is used to set to delete one or all user pictures. To delete a single user picture, specify the name of the picture, which is named in the format of User ID.jpg, for example, 11.jpg. To delete all user pictures at a time, set this parameter to ALL, which is case-sensitive.

2. The FilePath parameter specifies the absolute path where the downloaded picture is to be saved, for example, C:\Users\HP\Desktop\.

Note

This interface is applicable to the new architecture firmware.

#### GetAllUserPhoto

VARIANT\_BOOL GetAllUserPhoto(LONG dwMachineNumber, BSTR dlDir)

To download all user pictures from the machine to the software.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dlDir | BSTR | [in] | Absolute path where the downloaded user pictures are to be saved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. The dlDir parameter specifies the absolute path where the downloaded pictures are to be saved, for example, C:\Users\HP\Desktop\.

2. All user pictures downloaded from the machine will be automatically saved in JPG format to the specified path one by one.

Note

This interface is applicable to the new architecture firmware.

#### GetPhotoNamesByTime

VARIANT\_BOOL GetPhotoNamesByTime(LONG dwMachineNumber, LONG iFlag, BSTR sTime, BSTR eTime, BSTR\* AllPhotoName)

To download attendance pictures from the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| iFlag | LONG | [in] | Flag bit |
| sTime | BSTR | [in] | Start time |
| eTime | BSTR | [in] | End time |
| AllPhotoName | BSTR\* | [out] | Names of attendance pictures |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. If the value of iFlag is 0, all attendance pictures on the machine will be downloaded. If the value is 1, attendance pictures between sTime and eTime will be downloaded.

2. sTime: specifies the start time, in the format of YYYY-MM-DD hh:mm:ss.

3. eTime: specifies the end time, in the format of YYYY-MM-DD hh:mm:ss.

4. AllPhotoName: specifies the names of attendance pictures, in the format of verification success pictures (separated with \t)+\n+verification failure pictures (separated with \t).

Note

Applicable to TFT

#### GetPhotoByName

VARIANT\_BOOL GetPhotoByName(LONG dwMachineNumber, BSTR PhotoName, BYTE\* PhotoData, LONG\* PhotoLength)

To download an attendance picture from the machine based on the picture name.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| PhotoName | BSTR | [in] | Picture name |
| PhotoData | BYTE\* | [out] | Picture data in binary format |
| PhotoLength | LONG\* | [out] | Picture size |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to TFT

#### GetPhotoCount.

VARIANT\_BOOL GetPhotoCount(LONG dwMachineNumber, LONG\* Count, LONG iFlag)

To query the number of attendance pictures on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Count | LONG\* | [in] | Picture quantity |
| iFlag | LONG | [in] | Flag bit |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

If the value of iFlag is 0, the total number of attendance pictures is returned. If the value is 1, the number of pictures that pass verification is returned. If the value is 2, the number of pictures that fail verification is returned.

Note

Applicable to TFT

#### ClearPhotoByTime

VARIANT\_BOOL ClearPhotoByTime(LONG dwMachineNumber, LONG iFlag, BSTR sTime, BSTR eTime)

To clear attendance pictures on the machine according to specified conditions.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| iFlag | LONG | [in] | Flag bit |
| sTime | BSTR | [in] | Start time |
| eTime | BSTR | [in] | End time |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. If the value of iFlag is 0, all attendance pictures on the machine will be downloaded. If the value is 1, attendance pictures between sTime and eTime will be downloaded.

2. sTime: specifies the start time, in the format of YYYY-MM-DD hh:mm:ss.

3. eTime: specifies the end time, in the format of YYYY-MM-DD hh:mm:ss.

Note

Applicable to TFT

#### GetPhotoByNameToFile

VARIANT\_BOOL GetPhotoByNameToFile(LONG dwMachineNumber, BSTR PhotoName, BSTR  LocalFileName);

To download an attendance picture from the machine based on the picture name.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| PhotoName | BSTR | [in] | File Name |
| LocalFileName | BSTR | [in] | File path name to save image |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

### Bell Functions

#### GetBellSchDataEx

VARIANT\_BOOL GetBellSchDataEx(LONG dwMachineNumber, LONG weekDay, LONG Index, LONG\* Enable, LONG\* hour, LONG\* min, LONG\* voice, LONG\* way, LONG\* InerBellDelay, LONG\* ExtBellDelay)

To query bell settings based on the specified weekday and bell index.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| weekDay | LONG | [in] | Weekday |
| Index | LONG | [in] | Bell index |
| Enable | LONG\* | [out] | Whether to enable the bell |
| hour | LONG\* | [out] | Hour |
| min | LONG\* | [out] | Minute |
| voice | LONG\* | [out] | Ringtone |
| way | LONG\* | [out] | Ringing mode |
| InerBellDelay | LONG\* | [out] | Internal ringing duration |
| ExtBellDelay | LONG\* | [out] | External ringing duration |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

NMS\_GetBellSchData

Attention

1. weekDay: specifies the weekday. The value 0 indicates Monday, 1 Tuesday, 2 Wednesday, 3 Thursday, 4 Friday, 5 Saturday, and 6 Sunday.

2. Index: specifies the bell index. The value range is 1-65535.

3. Enable: specifies whether to enable the bell. The value 0 indicates disabling the bell and 1 indicates enabling the bell.

4. voice: specifies the ringtone. The value ranges from 1 to 10, representing bell01.wav to bell10.wav respectively.

5. way: specifies the ringing mode. The value 0 indicates internal ringing, 1 external ringing, and 2 internal and external ringing.

6. InerBellDelay: specifies the internal ringing duration. The value ranges from 1 to 999, in seconds.

7. ExtBellDelay: specifies the external ringing duration. The value ranges from 1 to 999, in seconds.

Note

This interface is applicable to the new architecture firmware.

#### SetBellSchDataEx

VARIANT\_BOOL SetBellSchDataEx(LONG dwMachineNumber, LONG weekDay, LONG Index, LONG Enable, LONG hour, LONG min, LONG voice, LONG way, LONG InerBellDelay, LONG ExtBellDelay)

To set bell information.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| weekDay | LONG | [in] | Weekday |
| Index | LONG | [in] | Bell index |
| Enable | LONG | [in] | Whether to enable the bell |
| hour | LONG | [in] | Hour |
| min | LONG | [in] | Minute |
| voice | LONG | [in] | Ringtone |
| way | LONG | [in] | Ringing mode |
| InerBellDelay | LONG | [in] | Internal ringing duration |
| ExtBellDelay | LONG | [in] | External ringing duration |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. weekDay: specifies the weekday. At most 7 days in a whole week can be set. Specify the weekday by using the least significant 7 bits of a byte. You can set multiple weekdays.  
The first bit indicates Monday (0x00000001), the second bit Tuesday (0x00000002), third bit Wednesday (0x00000004), fourth bit Thursday (0x00000008), fifth bit Friday (0x00000010), sixth bit Saturday (0x00000020), and seventh bit Sunday (0x00000040).  
To set multiple days, add the corresponding values. For example, to set Monday and Thursday, add 1 and 8, making 9. To set Tuesday, Friday and Saturday, add 2, 16 and 32, making 50.

2. Index: specifies the bell index. The value range is 1-65535.

3. Enable: specifies whether to enable the bell. The value 0 indicates disabling the bell and 1 indicates enabling the bell.

4. voice: specifies the ringtone. The value ranges from 1 to 10, representing bell01.wav to bell10.wav respectively.

5. way: specifies the ringing mode. The value 0 indicates internal ringing, 1 external ringing, and 2 internal and external ringing.

6. InerBellDelay: specifies the internal ringing duration. The value ranges from 1 to 999, in seconds.

7. ExtBellDelay: specifies the external ringing duration. The value ranges from 1 to 999, in seconds.

Note

This interface is applicable to the new architecture firmware.

#### GetDayBellSchCount

VARIANT\_BOOL GetDayBellSchCount(LONG dwMachineNumber, LONG\* DayBellCnt)

To query the number of bells that are set on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| DayBellCnt | LONG\* | [out] | Bell quantity |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### GetMaxBellIDInBellSchData

VARIANT\_BOOL GetMaxBellIDInBellSchData(LONG dwMachineNumber, LONG\* MaxBellID)

To obtain the maximum bell index among all bells.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| MaxBellID | LONG\* | [out] | Weekday |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

NMS\_GetMaxBellIDInBellSchData

Attention

Note

This interface is applicable to the new architecture firmware.

#### ReadAllBellSchData

VARIANT\_BOOL ReadAllBellSchData(LONG dwMachineNumber)

To read the information about all bells to the SDK memory.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function needs to be used together with GetEachBellInfo to obtain bell information.

Note

This interface is applicable to the new architecture firmware.

#### GetEachBellInfo

VARIANT\_BOOL GetEachBellInfo(LONG dwMachineNumber, LONG\* weekDay, LONG\* Index, LONG\* Enable, LONG\* hour, LONG\* min, LONG\* voice, LONG\* way, LONG\* InerBellDelay, LONG\* ExtBellDelay)

To read bell information records from the SDK memory one by one.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| weekDay | LONG\* | [out] | Weekday |
| Index | LONG\* | [out] | Bell index |
| Enable | LONG\* | [out] | Whether to enable the bell |
| hour | LONG\* | [out] | Hour |
| min | LONG\* | [out] | Minute |
| voice | LONG\* | [out] | Ringtone |
| way | LONG\* | [out] | Ringing mode |
| InerBellDelay | LONG\* | [out] | Internal ringing duration |
| ExtBellDelay | LONG\* | [out] | External ringing duration |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. This function needs to be used together with ReadAllBellSchData, which is used to read all bell information to the memory.

2. weekDay: specifies the weekday. At most 7 days in a whole week can be set. Specify the weekday by using the least significant 7 bits of a byte. You can set multiple weekdays.  
The first bit indicates Monday (0x00000001), the second bit Tuesday (0x00000002), third bit Wednesday (0x00000004), fourth bit Thursday (0x00000008), fifth bit Friday (0x00000010), sixth bit Saturday (0x00000020), and seventh bit Sunday (0x00000040).  
You can learn the weekday settings by checking whether the corresponding bit is set. For example, if the returned value is 50, which is the sum of 0x00000002, 0x00000010 and 0x00000020, the bell is set to ring on Tuesday, Friday and Saturday.

3. Index: specifies the bell index. The value range is 1-65535.

4. Enable: specifies whether to enable the bell. The value 0 indicates disabling the bell and 1 indicates enabling the bell.

5. voice: specifies the ringtone. The value ranges from 1 to 10, representing bell01.wav to bell10.wav respectively.

6. way: specifies the ringing mode. The value 0 indicates internal ringing, 1 external ringing, and 2 internal and external ringing.

7. InerBellDelay: specifies the internal ringing duration. The value ranges from 1 to 999, in seconds.

8. ExtBellDelay: specifies the external ringing duration. The value ranges from 1 to 999, in seconds.

Note

This interface is applicable to the new architecture firmware.

### UserValidDate Functions

#### SetUserValidDate

VARIANT\_BOOL SetUserValidDate(LONG dwMachineNumber, BSTR UserID, LONG Expires, LONG ValidCount, BSTR StartDate, BSTR EndDate)

To set the validity period of a user account.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UserID | BSTR | [in] | User ID |
| Expires | LONG | [in] | Validity period type |
| ValidCount | LONG | [in] | Number of valid usage times |
| StartDate | BSTR | [in] | Start time of the validity period, in the format of YYYY-MM-DD hh:mm:ss |
| EndDate | BSTR | [in] | End time of the validity period, in the format of YYYY-MM-DD hh:mm:ss |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. UserID: specifies the ID. The user ID is a string of English letters.

2. Expires: specifies the validity period type. The value ranges from 0 to 3. The value 0 indicates not limiting the the validity period of the user account, 1 indicates limiting the validity period by setting the start time and end time, 2 indicates limiting the validity period by setting the number of usage times, and 3 indicates limiting the validity period by setting the start time, end time, and number of usage times.

3. ValidCount: specifies the number of usage times of the user account. The value is larger than or equal to 0.

4. StartDate: specifies the start time of the validity period. Only the YYYY-MM-DD part is kept because time is accurate only to date on the firmware.

5. EndDate: specifies the end time of the validity period. Only the YYYY-MM-DD part is kept because time is accurate only to date on the firmware.

Note

This interface is applicable to the new architecture firmware.

#### GetUserValidDate

VARIANT\_BOOL GetUserValidDate(LONG dwMachineNumber, BSTR UserID, LONG\* Expires, LONG\* ValidCount, BSTR\* StartDate, BSTR\* EndDate)

To query the validity period of a user account.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UserID | BSTR | [in] | User ID |
| Expires | LONG\* | [out] | Validity period type |
| ValidCount | LONG\* | [out] | Number of valid usage times |
| StartDate | BSTR\* | [out] | Start time of the validity period, in the format of YYYY-MM-DD hh:mm:ss |
| EndDate | BSTR\* | [out] | End time of the validity period, in the format of YYYY-MM-DD hh:mm:ss |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. UserID: specifies the ID. The user ID is a string of English letters.

2. Expires: specifies the validity period type. The value ranges from 0 to 3. The value 0 indicates not limiting the the validity period of the user account, 1 indicates limiting the validity period by setting the start time and end time, 2 indicates limiting the validity period by setting the number of usage times, and 3 indicates limiting the validity period by setting the start time, end time, and number of usage times.

3. ValidCount: specifies the number of usage times of the user account. The value is larger than or equal to 0.

4. StartDate: specifies the start time of the validity period. Only the YYYY-MM-DD part is kept because time is accurate only to date on the firmware.

5. EndDate: specifies the end time of the validity period. Only the YYYY-MM-DD part is kept because time is accurate only to date on the firmware.

Note

This interface is applicable to the new architecture firmware.

### Personalise Functions

#### UploadTheme

VARIANT\_BOOL UploadTheme(LONG dwMachineNumber, BSTR FileName, BSTR InDevName)

To upload a theme picture to the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Save path + File name |
| InDevName | BSTR | [in] | File name in the device |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The FileName parameter specifies the file name and save path of the theme picture to be uploaded.

Note

This interface is applicable to the new architecture firmware.

#### UploadPicture

VARIANT\_BOOL UploadPicture(LONG dwMachineNumber, BSTR FileName, BSTR InDevName)

To upload a background picture to the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Save path + File name |
| InDevName | BSTR | [in] | File name in the device |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The FileName parameter specifies the file name and save path of the background picture to be uploaded.

Note

This interface is applicable to the new architecture firmware.

#### DownloadPicture

VARIANT\_BOOL DownloadPicture(LONG dwMachineNumber, BSTR FileName, BSTR FilePath)

To download the background picture from the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | Name of the picture file |
| FilePath | BSTR | [in] | Path where the downloaded picture is to be saved |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

### APP Info Functions

#### GetAllAppFun

VARIANT\_BOOL GetAllAppFun(LONG dwMachineNumber, BSTR\* AppName, BSTR\* FunofAppName)

To query all App names, all App and corresponding Fun names.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| AppName | BSTR | [out] | App name list.The data is in .txt format and format name is [App name]. Records are separated with \r\n; |
| FunofAppName | BSTR | [out] | App and App functions lsit.The data is in .txt format and format name is [Fun name, Function name]. Records are separated with \r\n; |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

It is applicable to the new architecture machine

#### GetAllRole

VARIANT\_BOOL GetAllRole(LONG dwMachineNumber, BSTR\* RoleName)

To query all role names and the corresponding permission names.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| RoleName | BSTR | [out] | To query all role names and the corresponding permission names. The data is in .txt format and format name is [Role name, Permission name]. Records are separated with \r\n; |

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

It is applicable to the new architecture machine

#### GetAppOfRole

VARIANT\_BOOL GetAppOfRole(LONG dwMachineNumber, LONG Permission, BSTR\* AppName)

To query all App names with specified role permission.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Permission | LONG | [in] | Permisssion of specified role |
| AppName | BSTR | [out] | AppName with the specified role permission will be returned. The data is in .txt format and format name is [App name]. Records are separated with \r\n; |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

It is applicable to the new architecture machine

#### GetFunOfRole

VARIANT\_BOOL GetFunOfRole(LONG dwMachineNumber, LONG Permission, BSTR\* FunName)

To query all function names with the specified role permission.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Permission | LONG | [in] | Permisssion of specified role |
| FunName | BSTR | [out] | FunName with the specified role permission will be returned. The data is in .txt format and format name is [Function name]. Records are separated with \r\n; |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

It is applicable to the new architecture machine

#### SetPermOfAppFun

VARIANT\_BOOL SetPermOfAppFun(LONG dwMachineNumber, LONG Permission, BSTR AppName, BSTR FunName)

To set the corresponding permission of function and App meanwhile.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Permission | LONG | [in] | Permission value to be set |
| AppName | BSTR | [in] | App name |
| FunName | BSTR | [in] | Fun name |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Function of FunName must be included in the functions of AppName

Note

It is applicable to the new architecture machine

#### DeletePermOfAppFun

VARIANT\_BOOL DeletePermOfAppFun(LONG dwMachineNumber, LONG Permission, BSTR AppName, BSTR FunName)

To delete the corresponding permission of function and App meanwhile.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Permission | LONG | [in] | Permission value to be deleted |
| AppName | BSTR | [in] | App name |
| FunName | BSTR | [in] | Fun name |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Function of FunName must be included in the functions of AppName

Note

It is applicable to the new architecture machine

#### IsUserDefRoleEnable

VARIANT\_BOOL IsUserDefRoleEnable(LONG dwMachineNumber, LONG Permission, VARIANT\_BOOL\* Enable)

To judge whether the user defined role is enabled.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Permission | LONG | [in] | Permission value to be deleted |
| Enable | BOOL\* | [out] | Whether the user defined role is enabled: 1 means enable and 0 means disable |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

It is applicable to the new architecture machine

### Template Integration Functions

#### SSR\_SetDeviceData

VARIANT\_BOOL SSR\_SetDeviceData(LONG dwMachineNumber, BSTR TableName, BSTR Datas, BSTR Options)

This function is applicable to an attendance machine on which the new firmware supports the PULL protocol. This function is used to set data, including the time segment, user information, and holiday settings. The data can be one or more records. If the primary key of an inserted record already exists, the original record will be overwritten.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| TableName | BSTR | [in] | Data table name. For details about the available tables, see attachment 1 PULL Data Dictionary for the New Firmware. |
| Datas | BSTR | [in] | Data records. The data is in .txt format. Records are separated with \r\n, and fields with values are separated with \t. |
| Options | BSTR | [in] | It is left blank by default and for extension. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### SSR\_GetDeviceData

VARIANT\_BOOL SSR\_GetDeviceData(LONG dwMachineNumber, BSTR\* Buffer, LONG BufferSize, BSTR TableName, BSTR FiledNames, BSTR Filter, BSTR Options)

This function is applicable to an attendance machine on which the new firmware supports the PULL protocol. This function is used to read data from the machine, including the punch records, time segments, user information, and holiday settings. The data can be one or more records.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Buffer | BSR\* | [out] | Buffer for storing returned data. The returned data is in .txt format and may contain multiple records, which are separated with \r\n. |
| BufferSize | LONG | [in] | Size of the buffer for storing returned data |
| TableName | BSTR | [in] | Data table name. For details about the available tables, see attachment 1 PULL Data Dictionary for the New Firmware. |
| FiledNames | BSTR | [in] | Field name list. The fields are separated with \t. \* indicates all fields. The field names are on the first line in the returned data. |
| Filter | BSTR | [in] | Filter criteria for reading data. If the data is a character string in the format of "field name operator value", multiple filter criteria separated with a comma (,) are supported. The details are as follows: |
| Options | BSTR | [in] | Options |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

#### SSR\_GetDeviceDataCount

VARIANT\_BOOL SSR\_GetDeviceDataCount(BSTR TableName, BSTR Filter, BSTR Options)

Query the number of The unification of multiple Biometric Templates within the device.

Parameters

Parameters are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| TableName | BSTR | [in] | Name of Table |
| Filter | BSTR | [in] | Filter condition |
| Options | BSTR | [in] | parameter |

Returns

returned value specification:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Success |
| False | BOOL | Fault |

See also

Attention

Note

This interface is applicable to B&W,TFT devices,For The unification of multiple Biometric Template Tablestructure, see the SSR\_SetDeviceData interface Attention

#### SSR\_DeleteDeviceData

VARIANT\_BOOL SSR\_DeleteDeviceData(LONG dwMachineNumber, BSTR TableName, BSTR Datas, BSTR Options)

Delete The unification of multiple Biometric Templates within the device.

Parameters

Parameters are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| TableName | BSTR | [in] | Name of Table |
| Datas | BSTR | [in] | Filter condition |
| Options | BSTR | [in] | parameter |

Returns

returned value specification:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Success |
| False | BOOL | Fault |

See also

Attention

Note

This interface is applicable to B&W,TFT devices,For The unification of multiple Biometric Template Tablestructure, see the SSR\_SetDeviceData interface Attention

#### Variable description : BiometricType

BiometricType.

Parameters

BiometricType Get the supported biometric type by retrieving the parameter and return 8-bit numeric string. Each represents a type of biometric type. (0 not support ; 1 support;e.g.:BiometricType=01100000 ,device support FP and Face) Character index 0 1 2 3 4 5 6 7 8  
Related Type General FP Face Voice Iris Retina Palmprint FingerVein Palmvein

See also

Attention

Note

Applicable to interface of unification of multiple Biometric Template

#### Variable description : BiometricVersion

BiometricVersion.

Parameters

BiometricVersion Get the version of the supported biometric type Get the version of the supported biometric type of the device by retrieving the parameter and return the version of the supported biometric type seperated by ":" (e.g.:BiometricVersion=0:10.0:7.0:::::device support FP10.0 and Face7.0) Character index 0 1 2 3 4 5 6 7 8  
Related Type General FP Face Voice Iris Retina Palmprint FingerVein Palmvein

See also

Attention

TableName

Note

Applicable to interface of unification of multiple Biometric Template

#### Variable description : BiometricMaxCount

BiometricMaxCount.

Parameters

BiometricMaxCount Get the supported biometric type data capacity by retrieving the parameter and return the supported biometric type data capacity seperated by ":" (e.g.:BiometricMaxCount=0:3000:1000:::::,device support 3000 FP and 1000 Face templates) Character index 0 1 2 3 4 5 6 7 8  
Related Type General FP Face Voice Iris Retina Palmprint FingerVein Palmvein The new device is directly returned by the firmware, and the old firmware is SDK compatible.(supported)

See also

Attention

TableName

Note

Applicable to interface of unification of multiple Biometric Template

#### Variable description : BiometricUsedCount

BiometricUsedCount.

Parameters

BiometricUsedCount Get the number of supported biometric type data by retrieving the padameter and return the number of supported biometric type data seperated by ":" (e.g.:BiometricUsedCount=0:100:10:::::,indicate there are 100 FP templates, 10 faces) Character index 0 1 2 3 4 5 6 7 8  
Related Type General FP Face Voice Iris Retina Palmprint FingerVein Palmvein

See also

Attention

TableName

Note

Applicable to interface of unification of multiple Biometric Template

### LiveFace Integration Functions

#### GetUserFacePhotoNames

**VARIANT\_BOOL GetUserFacePhotoNames(LONG dwMachineNumber, BSTR\* AllPhotoName)**

To download all face photos file name for device.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| AllPhotoName | BSTR\* | [out] | Names of face pictures |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

AllPhotoName: specifies the names of attendance pictures, in the format of verification success pictures (separated with \t)+\n+verification failure pictures (separated with \t).

**Note**

Applicable to the device which support visible face recognition.

#### GetUserFacePhotoByName

VARIANT\_BOOL GetUserFacePhotoByName(LONG dwMachineNumber, BSTR PhotoName, BYTE\* PhotoData, LONG\* PhotoLength)

To download an user face photo from the machine based on the picture name.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| PhotoName | BSTR | [in] | Picture name |
| PhotoData | BYTE\* | [out] | Picture data in binary format |
| PhotoLength | LONG\* | [out] | Picture size |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to the device which support visible face recognition.

#### **GetUserFacePhotoCount**

VARIANT\_BOOL **GetUserFacePhotoCount**(LONG dwMachineNumber, LONG\* Count)

To query the number of face pictures on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Count | LONG\* | [in] | Picture quantity |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to the device which support visible face recognition.

#### SendUserFacePhoto

VARIANT\_BOOL SendUserFacePhoto(LONG dwMachineNumber, BSTR FileName)

To upload user face picture to device.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| FileName | BSTR | [in] | User face picture file path name. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to the device which support visible face recognition.

## Access Control Functions(Time Zone, Group, Open Door Combination)

### GetUserGroup

VARIANT\_BOOL GetUserGroup(LONG dwMachineNumber, LONG dwEnrollNumber, LONG\* UserGrp)

To obtain the ID of the group to which a specified user belongs.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| UserGrp | LONG\* | [out] | Returned group ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of UserGrp is the ID of the group to which the user specified by dwEnrollNumber belongs. The value range is 1-5.

Note

Applicable to B&W, TFT, and IFACE devices

### SetUserGroup

VARIANT\_BOOL SetUserGroup(LONG dwMachineNumber, LONG dwEnrollNumber, LONG UserGrp)

To set the group to which a specified user belongs.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| UserGrpl | LONG | [in] | Group ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of UserGrp is the ID of the group to which the user specified by dwEnrollNumber belongs. The value range is 1-5.

Note

Applicable to B&W, TFT, and IFACE devices

### GetTZInfo

VARIANT\_BOOL GetTZInfo(LONG dwMachineNumber, LONG TZIndex, BSTR\* TZ)

To obtain the information about a time segment with a specified index.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| TZIndex | LONG | [in] | Time segment index |
| TZ | BSTR\* | [out] | The value is the information about the time segment with the index specified by TZIndex. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of the TZ parameter is the information about the time segment with the index specified by TZIndex. Every eight digits indicate a time segment in the format of hhmmhhmm. For example, 10111223000023590000235900002359000023590000235900002359 indicates the time segment from 10:11 to 12:23 on Sunday and the whole day from Monday to Saturday.

Note

Applicable to B&W, TFT, and IFACE devices

### SetTZInfo

VARIANT\_BOOL SetTZInfo(LONG dwMachineNumber, LONG TZIndex, BSTR TZ)

To set the information about a time segment with a specified index.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| TZIndex | LONG | [in] | Time segment index |
| TZ | BSTR | [in] | Time segment information to be set |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZ parameter specifies the time segment information to be set. Every eight digits indicate a time segment in the format of hhmmhhmm.For example, 10111223000023590000235900002359

000023590000235900002359 indicates the time segment from 10:11 to 12:23 on Sunday and the whole day from Monday to Saturday.

Note

Applicable to B&W, TFT, and IFACE devices

### GetUnlockGroups

VARIANT\_BOOL GetUnlockGroups(LONG dwMachineNumber, BSTR\* Grps)

To obtain unlock combinations of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Grps | BSTR\* | [out] | Current unlock combination of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of the Grps parameter is an unlock combination. There are totally 10 unlock combinations, separated with a colon (:). For example, 12:23:14:15 represents four valid combinations, which are combination 1 (12 represents groups 1 and 2), combination 2 (23 represents groups 2 and 3), combination 3 (14 represents groups 1 and 4), and combination 4 (15 represents groups 1 and 5).

Note

Applicable to B&W

### SetUnlockGroups

VARIANT\_BOOL SetUnlockGroups(LONG dwMachineNumber, BSTR Grps)

To set unlock combinations.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Grps | BSTR | [in] | Unlock combination |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Totally 10 unlock combinations need to be set, and separate these combinations with a colon (:). For example, 12:23:14:15 represents four valid combinations, which are combination 1 (12 represents groups 1 and 2), combination 2 (23 represents groups 2 and 3), combination 3 (14 represents groups 1 and 4), and combination 4 (15 represents groups 1 and 5).

Note

Applicable to B&W

### SSR\_SetUnLockGroup

VARIANT\_BOOL SSR\_SetUnLockGroup(LONG dwMachineNumber, LONG CombNo, LONG Group1, LONG Group2, LONG Group3，LONG Group4，LONG Group5)

To set unlock combinations.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| CombNo | LONG | [in] | Unlock combination ID ranging from 1 to 10. The machine supports at most 10 unlock combinations. |
| Group1 | LONG | [in] | Unlock combination group 1 |
| Group2 | LONG | [in] | Unlock combination group 2 |
| Group3 | LONG | [in] | Unlock combination group 3 |
| Group4 | LONG | [in] | Unlock combination group 4 |
| Group5 | LONG | [in] | Unlock combination group 5 |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Group1, Group2, Group3, Group4 and Group5 are group IDs of an unlock combination. Each unlock combination contains five group IDs. The group ID ranges from 1 to 99. For example, SSR\_SetUnLockGroup(1,1,2,23,14,0,56) indicates that the personnel of groups 2, 23, 14 and 56 need to verify together to open the door.

Note

Applicable to TFT and IFACE devices

### SSR\_GetUnLockGroup

VARIANT\_BOOL SSR\_GetUnLockGroup(LONG dwMachineNumber, LONG CombNo, LONG\* Group1, LONG\* Group2, LONG\* Group3, LONG\* Group4, LONG\* Group5)

To obtain unlock combination information based on the group ID.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| CombNo | LONG | [in] | Unlock combination ID ranging from 1 to 10 |
| Group1 | LONG\* | [out] | Unlock combination group 1 |
| Group2 | LONG\* | [out] | Unlock combination group 2 |
| Group3 | LONG\* | [out] | Unlock combination group 3 |
| Group4 | LONG\* | [out] | Unlock combination group 4 |
| Group5 | LONG\* | [out] | Unlock combination group 5 |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The values of Group1, Group2, Group3, Group4, and Group5 are group IDs of the specified unlock combination. The group IDs are returned. Each unlock combination contains a maximum of five groups. The group ID ranges from 1 to 99.

Note

Applicable to TFT and IFACE devices

### GetGroupTZs

VARIANT\_BOOL GetGroupTZs(LONG dwMachineNumber, LONG GroupIndex, LONG\* TZs)

To obtain the time segments of a specified group. This function differs from GetGroupTZStr in the format of the returned values.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| GroupIndex | LONG | [in] | Group index, ranging from 1 to 5 |
| TZs | LONG\* | [out] | Time segment index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs parameter is a LONG pointer. Its value is the indexes of the three time segments used by the group specified by GroupIndex, indicated by TZs[0], TZs[1] and TZs[2] respectively.

Note

Applicable to B&W

### SetGroupTZs

VARIANT\_BOOL SetGroupTZs(LONG dwMachineNumber, LONG GroupIndex, LONG\* TZs)

To set the three time segments of a specified group. This function differs from SetGroupTZStr in the format of the specified values.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| GroupIndex | LONG | [in] | Group index, ranging from 1 to 5 |
| TZs | LONG\* | [in] | Time segment index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs parameter is a LONG pointer. The three time segments are specified by TZs[0], TZs[1], and TZs[2] respectively.

Note

Applicable to B&W

### GetGroupTZStr

VARIANT\_BOOL GetGroupTZStr(LONG dwMachineNumber, LONG GroupIndex,BSTR\* TZs)

To obtain the time segments of a specified group. This function differs from GetGroupTZs in the format of the returned values.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| GroupIndex | LONG | [in] | Group index, ranging from 1 to 5 |
| TZs | BSTR\* | [out] | Time segment index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The value of TZs is the indexes of the time segments of the group specified by GroupIndex. Each group contains three time segments separated with a colon (:). For example, if the returned value is 1:23:13, the indexes of the three time segments are 1, 23 and 13 respectively.

Note

Applicable to B&W

### SetGroupTZStr

VARIANT\_BOOL SetGroupTZStr(LONG dwMachineNumber, LONG GroupIndex, BSTR TZs)

To set the time segments of a specified group. This function differs from SetGroupTZs in the format of the specified values.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| GroupIndex | LONG | [in] | Group index, ranging from 1 to 5 |
| TZs | BSTR | [in] | Time segment index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs parameter specifies the indexes of the time segments of the group described by GroupIndex. Each group contains three time segments separated with a colon (:). For example, if the value is set to 1:23:13, the indexes of the three time segments are 1, 23 and 13 respectively.

Note

Applicable to B&W

### SSR\_SetGroupTZ

VARIANT\_BOOL SSR\_SetGroupTZ(LONG dwMachineNumber, LONG GroupNo, LONG Tz1, LONG Tz2, LONG Tz3, LONG ValidHoliday, LONG VerifyStyle)

To set the time segments of a group.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| GroupNo | LONG | [in] | Group ID, ranging from 1 to 99 |
| Tz1 | LONG | [in] | Time segment 1 |
| Tz2 | LONG | [in] | Time segment 2 |
| Tz3 | LONG | [in] | Time segment 3 |
| ValidHoliday | LONG | [in] | Whether the time segments are valid on holidays |
| VerifyStyle | LONG | [in] | Group verification mode |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The VerifyStyle parameter specifies the group verification mode. 0(FP/PW/RF), 1(FP), 2(PIN), 3(PW), 4(RF), 5(FP&RF), 6(FP/PW), 7(FP/RF), 8(PW/RF), 9(PIN&FP), 10(FP&PW), 11(PW&RF), 12(FP&PW&RF), 13(PIN&FP&PW), 14(FP&RF/PIN).

Note

Applicable to TFT and IFACE devices

### SSR\_GetGroupTZ

VARIANT\_BOOL SSR\_GetGroupTZ(LONG dwMachineNumber, LONG GroupNo, LONG Tz1, LONG Tz2, LONG Tz3, LONG ValidHoliday, LONG VerifyStyle)

To obtain the time segments of a group.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Fingerprint machine ID |
| GroupNo | LONG | [in] | Group ID, ranging from 1 to 99 |
| Tz1 | LONG\* | [out] | The indexes of the three time segments of the specified group are returned. The index ranges from 1 to 50. |
| Tz2 | LONG\* | [out] | The indexes of the three time segments of the specified group are returned. The index ranges from 1 to 50. |
| Tz3 | LONG\* | [out] | The indexes of the three time segments of the specified group are returned. The index ranges from 1 to 50. |
| ValidHoliday | LONG\* | [out] | Whether the time segments are valid on holidays. The value 1 indicates that the time segments are valid on holidays and 0 indicates that the time segments are invalid on holidays. |
| VerifyStyle | LONG\* | [out] | Verification mode of the fingerprint machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The VerifyStyle parameter specifies the group verification mode. 0(FP/PW/RF), 1(FP), 2(PIN), 3(PW), 4(RF), 5(FP&RF), 6(FP/PW), 7(FP/RF), 8(PW/RF), 9(PIN&FP), 10(FP&PW), 11(PW&RF), 12(FP&PW&RF), 13(PIN&FP&PW), 14(FP&RF/PIN).

Note

Applicable to TFT and IFACE devices

### GetUserTZs

VARIANT\_BOOL GetUserTZs(LONG dwMachineNumber, LONG UserID, LONG TZs)

To obtain the time segments of a user. Each user has three time segments. This function differs from GetUserTZStr in the format of the returned time segments.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UserID | LONG | [in] | User ID |
| TZs | LONG\* | [out] | Time segments during which the user can open the door |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs pointer has three values that store three time segment indexes, which can be read from TZs[0], TZs[1] and TZs[2] respectively.

Note

Applicable to B&W, TFT, and IFACE devices

### SetUserTZs

VARIANT\_BOOL SetUserTZs(LONG dwMachineNumber, LONG UserID, LONG\* TZs)

To set the time segments of a user. At most three time segments can be set for a user. This function differs from SetUserTZStr in the format of specified time segments.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UserID | LONG | [in] | User ID |
| TZs | LONG\* | [in] | Time segment index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs parameter specifies the indexes of the time segments of the user described by UserID. The three indexes can be specified by TZs[0], TZs[1] and TZs[2] respectively. When TZs[0] is 0, the group setting takes effect. When it is 1, a user-defined setting takes effect.

Note

Applicable to B&W, TFT, and IFACE devices

### GetUserTZStr

VARIANT\_BOOL GetUserTZStr(LONG dwMachineNumber, LONG UserID, BSTR\* TZs)

To obtain the time segments of a user. This function differs from GetUserTZs in the format of returned time segments.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UserID | LONG | [in] | User ID |
| TZs | BSTR\* | [out] | Unlock time segments of the user |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs parameter specifies the unlock time segments of the user. The details are as follows:  
Monochrome access control panel: X1:X2:X3. X1, X2 and X3 are indexes of user-defined time segments. If X1 is 0, the group time segments are used. To query whether a user uses the group time segments, invoke the UseGroupTimeZone function and check the returned value. If user A uses user-defined time segments 1 and 2, the fingerprint machine will return 1:2:0. If user B uses the group time segments, the fingerprint machine will return 0:0:0.

Color-screen access control panel: X1:X2:X3:X4. X4 specifies whether to use the group time segments. The value 0 indicates using the group time segments and 1 indicates using user-defined time segments. X1, X2 and X3 are indexes of used time segments. For example, if user A uses user-defined time segments 1 and 2, the fingerprint machine will return 1:2:0:1. If user B uses the group time segments defined as 1:1:1:0, the fingerprint machine will return 0:0:0:0.

Note

Applicable to B&W, TFT, and IFACE devices

### SetUserTZStr

VARIANT\_BOOL SetUserTZStr(LONG dwMachineNumber, LONG UserID, BSTR TZs)

To set the time segments of a user. The time segments are separated with a colon (:). This function differs from SetUserTZs in the format of specified time segments.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UserID | LONG | [in] | User ID |
| TZs | BSTR | [in] | Unlock time segments of the user |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The TZs parameter specifies the unlock time segments of the user. The details are as follows:  
Monochrome access control panel: X1:X2:X3. X1, X2 and X3 are indexes of user-defined time segments. If X1 is 0, the group time segments are used. To query whether a user uses the group time segments, invoke the UseGroupTimeZone function and check the returned value. If user A uses user-defined time segments 1 and 2, the fingerprint machine will return 1:2:0. If user B uses the group time segments, the fingerprint machine will return 0:0:0.Color-screen access control panel: X1:X2:X3:X4. X4 specifies whether to use the group time segments. The value 0 indicates using the group time segments and 1 indicates using user-defined time segments. X1, X2 and X3 are indexes of used time segments. For example, if user A uses user-defined time segments 1 and 2, the fingerprint machine will return 1:2:0:1. If user B uses the group time segments defined as 1:1:1:0, the fingerprint machine will return 0:0:0:0.

Note

Applicable to B&W, TFT, and IFACE devices

### ACUnlock

VARIANT\_BOOL ACUnlock(LONG dwMachineNumber, LONG Delay)

To enable the access control panel to output door opening level and close the door after Delay/10 seconds.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Delay | LONG | [in] | Delay in closing the door |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetACFun

VARIANT\_BOOL GetACFun(LONG\* ACFun)

To check whether the machine has the access control function.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| ACFun | LONG\* | [out] | Flag of the access control function |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The ACFun parameter is an access control flag. The value 0 indicates that access control is unavailable, 1 indicates simple access control, 2 indicates medium-level access control, 6 indicates advanced access control, and 14 indicates advanced access control+normal open.

Note

Applicable to B&W, TFT, and IFACE devices

### GetDoorState

VARIANT\_BOOL GetDoorState(LONG MachineNumber, LONG\* State)

To query the current door status. The value 1 indicates that the door is opened and 0 indicates that the door is closed.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| MachineNumber | LONG | [in] | Machine ID |
| State | LONG\* | [out] | Door status |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### UseGroupTimeZone

VARIANT\_BOOL UseGroupTimeZone()

To query whether a user uses the group time segments.

Parameters

none

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

This function must be used together with GetUserTZs or GetUserTZStr. Specifically, invoke the GetUserTZs or GetUserTZStr function to obtain the time segment information about a user, and then invoke UseGroupTimeZone to check whether the user uses the group time segments.

Note

Applicable to B&W, TFT, and IFACE devices

### TurnOffAlarm

VARIANT\_BOOL TurnOffAlarm(LONG MachineNumber)

To turn off alarms.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| MachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

This interface is applicable to the new architecture firmware.

## Device Management Functions

### IsTFTMachine

VARIANT\_BOOL IsTFTMachine(LONG dwMachineNumber)

To check whether the machine is a color-screen one.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetDeviceStatus

VARIANT\_BOOL GetDeviceStatus(LONG dwMachineNumber, LONG dwStatus, LONG\* dwValue)

To query the data storage status on the machine, such as the number of administrators and number of users.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwStatus | LONG | [in] | Data to be obtained |
| dwValue | LONG\* | [out] | Content of the data specified by dwStatus |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The dwStatus parameter specifies the data to be obtained. The value range is 1-22. The values are described as follows:  
1 Number of administrators  
2 Number of registered users  
3 Number of fingerprint templates on the machine  
4 Number of passwords  
5 Number of operation records  
6 Number of attendance records  
7 Fingerprint template capacity  
8 User capacity  
9 Attendance record capacity  
10 Remaining fingerprint template capacity  
11 Remaining user capacity  
12 Remaining attendance record capacity  
21 Number of faces  
22 Face capacity  
0 Other conditions

Note

Applicable to B&W, TFT, and IFACE devices

### GetDeviceInfo

VARIANT\_BOOL GetDeviceInfo(LONG dwMachineNumber, LONG dwInfo, LONG\* dwValue)

To obtain machine information, such as the language and baud rate.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwInfo | LONG | [in] | Information type |
| dwValue | LONG\* | [out] | Informatin of the type specified by dwInfo |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

dwInfo: specifies the type of information to be obtained. The value ranges from 1 to 68, and cannot be 65. The values are described as follows:

1. Maximum number of administrators, which is fixed at 500.
2. Machine ID.
3. Language: If the value of dwValue is 0, the language is English. The value 1 indicates other conditions. The value 2 indicates Traditional Chinese, and 3 indicates Thai.
4. Idle duration (in minutes): After the specified idle duration elapses, the machine will enter the standby state or be shut down.
5. Lock control duration, that is, the lock drive duration.
6. Attendance record quantity alarm: When the specified attendance record quantity is reached, the machine will raise an alarm to remind the user.
7. Operation record quantity alarm: When the specified operation record quantity is reached, the machine will raise an alarm to remind the user.
8. Duplicate record time: minimum interval for a user to record the same attendance state.
9. Baud rate for RS232/485 communication: The value 0 indicates the baud rate of 1200 bps, 1 indicates 2400 bps, 2 indicates 4800 bps, 3 indicates 9600 bps, 4 indicates 19200 bps, 5 indicates 38400 bps, 6 indicates 57600 bps, and Others indicates 115200 bps.
10. Parity check bit, return value of which is fixed at 0.
11. Stop bit, return value of which is fixed at 0.
12. Date separator, return value of which is fixed at 1.
13. Whether to enable network functions: The value 1 indicates enabling network functions and 0 indicates disabling network functions.
14. Whether to enable RS232.
15. Whether to enable RS485.
16. Whether to enable announcements.
17. Whether to perform high-speed comparison.
18. Idle mode, that is, the state of the machine during idle hours. The value 87 indicates shutdown and 88 indicates hibernation.
19. Automatic shutdown time: The default value is 255, which means that the machine will not automatically shut down.
20. Automatic startup time: The default value is 55, which means that the machine will not automatically start.
21. Automatic hibernation time: The default value is 255, which means that the machine will not automatically enter the hibernation state.
22. Automatic ringing time 1: The default value is 65535, which means that the bell will not automatically ring.
23. 1:N comparison threshold.
24. Registration threshold.
25. 1:1 comparison threshold.
26. Whether to display the matching score during verification.
27. Number of people that unlock the door concurrently.
28. Verify the card number only.
29. Network speed: The value 1 indicates 100M-H, 4 indicates 10M-F, 5 indicates 100M-F, 8 indicates AUTO, and Others indicates 10M-H.
30. Whether a card must be registered.
31. Waiting time before the machine automatically returns to the initial state if no operation is performed.
32. Waiting time before the machine automatically returns to the initial state if no response is returned after the PIN is input.
33. Waiting time before the machine automatically returns to the initial state if no operation is performed after entering the menu.
34. Time format.
35. Whether 1:1 comparison is mandatory.  
    36 40. Automatic ringing time 2, 3, 4, 5, and 6: The default value is 65535, which means that the bell will not automatically ring.  
    41 56. Automatic status change time 1 16: The default values are all -1, which means that the status will not change automatically.
36. Wiegand failure ID.
37. Wiegand duress ID.
38. Wiegand zone bit.
39. Pulse width of Wiegand outputs.
40. Pulse interval of Wiegand outputs.
41. ID of the start sector on the MIFARE card where fingerprints are stored.
42. Total number of sectors on the MIFARE card where fingerprints are stored.
43. Number of fingerprints stored on the MIFARE card.
44. Whether to display the attendance status.  
    67-68. Meaningless.
45. TCP Comm Port.
46. UDP port.
47. Fingerprint algorithm version.
48. Face algorithm version.
49. Finger vein version
50. FaceFunOn.
51. PIN2Width.
52. IsSupportABCPin.
53. IMEFunOn.
54. IsSupportAlarmExt.
55. ~DCTZ.
56. ~DOTZ.
57. dwValue serves the input and output parameter. The input indicates the name of another option to be obtained and the output is the value of this option. This function is similar to GetSysOption in this case.  
    [Note] The values of the preceding time points are all digits, which can be converted into actual time points. Specifically, convert a value into binary format, the least significant eight bits indicate the minute and the most significant eight bits indicate the hour. For example, if the value is 2860, it is 101100101100 in binary, the least significant eight bits are 00101100, which is 44 in decimal, and the most significant eight bits are 00001011, which is 11 in decimal. That is, the actual time point is 11:44.

Note

Applicable to B&W, TFT, and IFACE devices

### SetDeviceInfo

VARIANT\_BOOL SetDeviceInfo(LONG dwMachineNumber, LONG dwInfo, LONG dwValue)

To set machine information, such as the language and duplicate record time.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwInfo | LONG | [in] | Information type, ranging from 1-20, 80, 81. For details about the meanings of values, see the description of GetDeviceInfo |
| dwValue | LONG | [in] | Information of the type specified by dwInfo |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

dwInfo specifies the information type, ranging from 1-20, 80, 81. For details about the meanings of values, see the description of GetDeviceInfo.

Note

Applicable to B&W, TFT, and IFACE devices

### SetDeviceTime

VARIANT\_BOOL SetDeviceTime(LONG dwMachineNumber)

To set the time of the machine to be the same as that of the local computer. To set specified time, see the description of SetDeviceTime2.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### SetDeviceTime2

VARIANT\_BOOL SetDeviceTime2(LONG dwMachineNumber, LONG dwYear, LONG dwMonth, LONG dwDay, LONG dwHour, LONG dwMinute, LONG dwSecond)

To set the time of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwYear | LONG | [in] | Year |
| dwMonth | LONG | [in] | Month |
| dwDay | LONG | [in] | Date |
| dwHour | LONG | [in] | Hour |
| dwMinute | LONG | [in] | Minute |
| dwSecond | LONG | [in] | Second |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetDeviceTime

VARIANT\_BOOL GetDeviceTime(LONG dwMachineNumber, LONG\* dwYear, LONG\* dwMonth, LONG\* dwDay, LONG\* dwHour, LONG\* dwMinute, LONG\* dwSecond)

To query the time of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwYear | LONG\* | [out] | Year |
| dwMonth | LONG\* | [out] | Month |
| dwDay | LONG\* | [out] | Date |
| dwHour | LONG\* | [out] | Hour |
| dwMinute | LONG\* | [out] | Minute |
| dwSecond | LONG\* | [out] | Second |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetSerialNumber

VARIANT\_BOOL GetSerialNumber(LONG dwMachineNumber, BSTR\* dwSerialNumber)

To query the serial number of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwSerialNumber | BSTR\* | [out] | Serial number |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetProductCode

VARIANT\_BOOL GetProductCode(LONG dwMachineNumber, BSTR\* lpszProductCode)

To query the product code of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| lpszProductCode | BSTR\* | [out] | Product code |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetFirmwareVersion

VARIANT\_BOOL GetFirmwareVersion(LONG dwMachineNumber, BSTR\* strVersion)

To query the firmware version of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| strVersion | BSTR\* | [out] | Firmware version of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetSDKVersion

VARIANT\_BOOL GetSDKVersion(BSTR\* strVersion)

To query the SDK version.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| strVersion | BSTR\* | [out] | SDK version |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetDeviceIP

VARIANT\_BOOL GetDeviceIP(LONG dwMachineNumber, BSTR\* IPAddr)

To query the IP address of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| IPAddr | BSTR\* | [out] | IP address |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### SetDeviceIP

VARIANT\_BOOL SetDeviceIP(LONG dwMachineNumber, BSTR IPAddr)

To set the IP address of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| IPAddr | BSTR | [in] | IP address |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetDeviceMAC

VARIANT\_BOOL GetDeviceMAC(LONG dwMachineNumber, BSTR\* sMAC)

To query the MAC address of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sMAC | BSTR\* | [out] | MAC address |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### SetDeviceMAC

VARIANT\_BOOL SetDeviceMAC(LONG dwMachineNumber, BSTR sMAC)

To set the MAC address of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sMAC | BSTR | [in] | MAC address |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetWiegandFmt

VARIANT\_BOOL GetWiegandFmt(LONG dwMachineNumber, BSTR\* sWiegandFmt)

To query the Wiegand format of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sWiegandFmt | BSTR\* | [out] | Wiegand format |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### SetWiegandFmt

VARIANT\_BOOL SetWiegandFmt(LONG dwMachineNumber, BSTR sWiegandFmt)

To set the Wiegand format of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| sWiegandFmt | BSTR | [in] | Wiegand format |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetCardFun

VARIANT\_BOOL GetCardFun(LONG dwMachineNumber, LONG\* CardFun)

To query whether the machine supports the RF card.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| CardFun | LONG\* | [in] | Whether the RF card is supported |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

If the value of CardFun is 1, the machine supports only the RF card. If the value is 2, both the RF card and fingerprints are supported. If the value is 0, the RF card is not supported.

Note

Applicable to B&W, TFT, and IFACE devices

### SetDeviceCommPwd

VARIANT\_BOOL SetDeviceCommPwd(LONG dwMachineNumber, LONG CommKey)

To set the communication password of the machine, which will be saved on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| CommKey | LONG | [in] | Communication password |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### SetCommPassword

VARIANT\_BOOL SetCommPassword(LONG CommKey)

To set the communication password of the PC. A connection can be set up between the machine and the PC only if their communication passwords are the same.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| CommKey | LONG | [in] | Communication password |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### QueryState

VARIANT\_BOOL QueryState(LONG\* State)

To query the current status of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| State | LONG\* | [out] | Current status of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

State specifies the current status of the machine. The meanings of the values are as follows:  
0 Waiting state  
1 Fingerprint registration state  
2 Fingerprint identification state  
3 Menu access state  
4 Busy state (handling other work)  
5 State of waiting for card writing

Note

Applicable to B&W, TFT, and IFACE devices

### GetVendor

VARIANT\_BOOL GetVendor(BSTR\* strVendor)

To query the manufacturer of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| strVendor | BSTR\* | [in] | Manufacturer of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### GetDeviceStrInfo

VARIANT\_BOOL GetDeviceStrInfo(LONG dwMachineNumber, LONG dwInfo, BSTR\* Value)

To query the delivery time of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwInfo | LONG | [in] | It can be set only to 1 |
| Value | BSTR\* | [out] | Delivery time of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

dwInfo can be set only to 1.

Note

Applicable to B&W, TFT, and IFACE devices

### GetPlatform

VARIANT\_BOOL GetPlatform(LONG dwMachineNumber, BSTR\* Platform)

To query the platform of the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Platform | BSTR\* | [out] | Platform name |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### ReadAOptions

VARIANT\_BOOL ReadAOptions (LONG dwMachineNumber, BSTR Option, BSTR Value)

Read the values of specified configuration parameters from the device. The parameters beginning with "~" are skipped.

Parameter

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine equipment number |
| Option | BSTR | [in] | parameter name |
| Value | BSTR | [in] | Parameter value |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT devices

### GetSysOption

VARIANT\_BOOL GetSysOption (LONG dwMachineNumber, BSTR Option, BSTR Value)

Get parameter values

Parameter

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine equipment number |
| Option | BSTR | [in] | parameter name |
| Value | BSTR | [in] | Parameter value |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Parameter name. When the parameter is the character string "~ZKFPVersion", if the returned Value is 10, the current device uses ZKFinger10.0; if the returned Value is null or 9, the current device uses ZKFinger9.0.

Note

Applicable to B&W, TFT devices

### SetSysOption

VARIANT\_BOOL SetSysOption(LONG dwMachineNumber, BSTR Option, BSTR Value)

To set parameter value.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| Option | BSTR | [in] | Parameter name |
| Value | BSTR | [in] | Parameter value |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

### GetDeviceStatusEx

LONG GetDeviceStatusEx(LONG dwMachineNumber)

To get the status of the P2P devices.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| 1 | LONG | Machine is reading data |
| 2 | LONG | Machine is writing data |
| 3 | LONG | Machine is working |
| greater than 3 | LONG | The machine is free and the software can operate the machine |

See also

Attention

P2P devices could not do multi-threaded work. Must get the device status and when the device isfree the software can operate the machine.

Note

Applicable to some P2P devices, such as the K Pro series attendace machine

## Others

### Device Control Functions

#### ClearAdministrators

VARIANT\_BOOL ClearAdministrators(LONG dwMachineNumber)

To clear the operation rights of all administrators.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### EnableDevice

VARIANT\_BOOL EnableDevice(LONG dwMachineNumber, VARIANT\_BOOL bFlag)

To enable or disable the machine. After the machine is disabled, the fingerprint, keyboard, and card modules are unavailable.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| bFlag | BOOL | [in] | Flag that indicates whether the machine is enabled. The value 1 indicates that the machine is enabled and 0 indicates that the machine is disabled. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### EnableClock

VARIANT\_BOOL EnableClock(LONG Enabled)

To enable or disable the display of the colon (:) in the time of the machine. The colon is displayed on the main screen of the machine if enabled, and not displayed if disabled.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| Enabled | LONG | [in] | Whether to enable the display of the colon. The value 1 indicates enabling the display of the colon and 0 indicates disabling the display of the colon. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### DisableDeviceWithTimeOut

VARIANT\_BOOL DisableDeviceWithTimeOut(LONG dwMachineNumber, LONG TimeOutSec)

To disable the machine for a period of time.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| TimeOutSec | LONG | [in] | Time period during which the machine is disabled |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### PowerOffDevice

VARIANT\_BOOL PowerOffDevice(LONG dwMachineNumber)

To shut down the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### RestartDevice

VARIANT\_BOOL RestartDevice(LONG dwMachineNumber)

To restart the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### Online Registration Functions

#### StartEnroll

VARIANT\_BOOL StartEnroll(LONG UserID, LONG FingerID)

To register a user. The machine will then enter the user registration state and waits for the user to scan fingerprints.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| UserID | LONG | [in] | ID of the user to be registered |
| FingerID | LONG | [in] | Index of the fingerprint of the user, ranging from 0 to 9 |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

After this function is invoked and the user scans fingerprints for three times, the machine may not respond when the user scans fingerprints again. In this case, invoke StartIdentify to enable the machine to enter the waiting state.

Note

Applicable to B&W

#### StartEnrollEx

VARIANT\_BOOL StartEnrollEx(BSTR UserID, LONG FingerID, LONG Flag)

To register a user. The machine will then enter the user registration state and waits for the user to scan fingerprints.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| UserID | BSTR | [in] | ID of the user to be registered |
| FingerID | LONG | [in] | Index of the fingerprint of the user, ranging from 0 to 9 |
| Flag | LONG | [in] | Flag that indicates whether the fingerprint template is valid or a duress fingerprint |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

1. After this function is invoked and the user scans fingerprints for three times, the machine may not respond when the user scans fingerprints again. In this case, invoke StartIdentify to enable the machine to enter the waiting state.

2. The Flag parameter indicates whether the fingerprint template is valid, or the fingerprint template is a duress fingerprint. The value 0 indicates that the fingerprint template is invalid, 1 indicates that the fingerprint template is valid, and 3 indicates that the fingerprint template is a duress fingerprint.

Note

Applicable to TFT

#### StartVerify

VARIANT\_BOOL StartVerify(LONG UserID, LONG FingerID)

To start 1:1 comparison.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| UserID | LONG | [in] | ID of the user to be verified |
| FingerID | LONG | [in] | Index of the fingerprint of the user, ranging from 0 to 9 |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

#### StartIdentify

VARIANT\_BOOL StartIdentify()

To start 1:N comparison. Then the machine will enter the 1:N verification state.

Parameters

None

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

#### CancelOperation

VARIANT\_BOOL CancelOperation()

To cancel the current fingerprint registration state of the machine.

Parameters

None

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W and TFT devices

### Card Operation Functions

#### WriteLCD

VARIANT\_BOOL WriteLCD(LONG Row, LONG Col, BSTR Text)

To write data. Specifically, this function is used to write a character string to any row of any column on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| Row | LONG | [in] | Start row |
| Col | LONG | [in] | Start column |
| Text | BSTR | [in] | Content to be written on the screen of the machine |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### ClearLCD

VARIANT\_BOOL ClearLCD()

To clear the screen of the machine. Specifically, this function is used to clear all information displayed on the LCD screen of the machine.

Parameters

None

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W

#### WriteCard

VARIANT\_BOOL WriteCard(LONG dwMachineNumber, LONG dwEnrollNumber, LONG dwFingerIndex1, BYTE\* TmpData1, LONG dwFingerIndex2, BYTE\* TmpData2, LONG dwFingerIndex3, BYTE\* TmpData3, LONG dwFingerIndex4, BYTE\* TmpData4)

To write a specified user and the fingerprint template of the user to the MF card. After this function is invoked, the user needs to punch the MF card on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | LONG | [in] | User ID |
| dwFingerIndex1 | LONG | [in] | Fingerprint index, ranging from 0 to 3 |
| TmpData1 | BYTE\* | [in] | Fingerprint template corresponding to the fingerprint index |
| dwFingerIndex2 | LONG | [in] | Fingerprint index, ranging from 0 to 3 |
| TmpData2 | BYTE\* | [in] | Fingerprint template corresponding to the fingerprint index |
| dwFingerIndex3 | LONG | [in] | Fingerprint index, ranging from 0 to 3 |
| TmpData3 | BYTE\* | [in] | Fingerprint template corresponding to the fingerprint index |
| dwFingerIndex4 | LONG | [in] | Fingerprint index, ranging from 0 to 3 |
| TmpData4 | BYTE\* | [in] | Fingerprint template corresponding to the fingerprint index |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

TmpData1 cannot be left blank

Note

Applicable to B&W, TFT, and IFACE devices

#### EmptyCard

VARIANT\_BOOL EmptyCard(LONG dwMachineNumber)

To clear the MF card.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

### Others

#### GetLastError

GetLastError(LONG\* dwErrorCode)

To query information about the last error.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwErrorCode | LONG\* | [out] | Error code |

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The dwErrorCode parameter specifies the error code. The values are described as follows:

1. During connection, the following error codes may be returned:  
0 Connected successfully  
-1 Failed to invoke the interface  
-2 Failed to initialize  
-3 Failed to initialize parameters  
-5 Data mode read error  
-6 Wrong password  
-7 Reply error  
-8 Receive timeout  
-307 Connection timeout  
In invoking other interfaces, the following error codes may be returned:  
-201 Device is busy   
-199 New Mode   
-103 device send back error of face version error   
-102 face template version error, like 8.0 face template send to 7.0 device   
-101 malloc memory failed   
-100 Not supported or the data does not exist  
-10 The length of transmitted data is incorrect  
-5 Data already exists  
-4 Insufficient space  
-3 Wrong size  
-2 File read/write error  
-1 The SDK is not initialized and needs to be reconnected  
0 Data not found or duplicate data  
1 Correct operation  
4 Parameter error  
101 Buffer allocation error  
102 repeat invoking

2. Underlying error codes:  
-12001 Socket creation timeout (connection timeout)  
-12002 Insufficient memory  
-12003 Wrong Socket version  
-12004 Not TCP protocol  
-12005 Waiting timeout  
-12006 Data transmission timeout  
-12007 Data reading timeout  
-12008 Failed to read Socket  
-13009 Waiting event error  
-13010 Exceeded retry attempts  
-13011 Wrong reply ID  
-13012 Checksum error  
-13013 Waiting event timeout  
-13014 DIRTY\_DATA  
-13015 Buffer size too small  
-13016 Wrong data length  
-13017 Invalid data read1  
-13018 Invalid data read2  
-13019 Invalid data read3  
-13020 Data loss  
-13021 Memory initialization error  
-15001 Invoking return value of status key issued by SetShortkey interface repeatedly  
-15002 Invoking return value of description issued by SetShortkey interface repeatedly  
-15003 The two level menu is not opened in the device, and the data need not be issued

3. getdevicedata and setdevicedata invocation error codes  
-15100 Error occurs in obtaining table structure  
-15101 The condition field does not exist in the table structure  
-15102 Inconsistency in the total number of fields  
-15103 Inconsistency in sorting fields  
-15104 Memory allocation error  
-15105 Data parsing error  
-15106 Data overflow as the transmitted data exceeds 4M  
-15108 Invalid options  
-15113 Data parsing error: table ID not found  
-15114 A data exception is returned as the number of fields is smaller than or equal to 0  
-15115 A data exception is returned as the total number of table fields is inconsistent with the total number of fields of the data

4. Firmware error codes:  
2000 Return OK to execute  
-2001 Return Fail to execute command  
-2002 Return Data  
-2003 Regstered event occorred  
-2004 Return REPEAT Command  
-2005 Return UNAUTH Command  
0xffff Return Unknown Command  
-4999 Device parameter read error  
-4998 Device parameter write error  
-4997 The length of the data sent by the software to the device is incorrect  
-4996 A parameter error exists in the data sent by the software to the device  
-4995 Failed to add data to the database  
-4994 Failed to update the database  
-4993 Failed to read data from the database  
-4992 Failed to delete data in the database  
-4991 Data not found in the database  
-4990 The data amount in the database reaches the limit  
-4989 Failed to allocate memory to a session  
-4988 Insufficient space in the memory allocated to a session  
-4987 The memory allocated to a session overflows  
-4986 File does not exist  
-4985 File read failure  
-4984 File write failure  
-4983 Failed to calculate the hash value  
-4982 Failed to allocate memory

Note

This interface is applicable to the new architecture firmware.

#### GetHIDEventCardNumAsStr

VARIANT\_BOOL GetHIDEventCardNumAsStr(BSTR\* strHIDEventCardNum)

To query the number of the card that is punched most recently.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| strHIDEventCardNum | BSTR\* | [out] | Number of the punched card |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### CaptureImage

VARIANT\_BOOL CaptureImage(VARIANT\_BOOL FullImage, LONG\* Width, LONG\* Height, BYTE\* Image, BSTR ImageFile)

To capture an image of the finger of which the fingerprint is being scanned.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| FullImage | BOOL | [in] | Whether to capture an image of the entire finger. The value True indicates capturing an image of the entire finger and False indicates capturing an image of only the fingerprint. |
| Width | LONG\* | [in] | Width of the captured image |
| Height | LONG\* | [in] | Height of the captured image |
| Image | BYTE\* | [in] | Fingerprint image in binary format |
| ImageFile | BSTR | [in] | Name of the file of captured image to be saved (including the path) |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### UpdateFirmware

VARIANT\_BOOL UpdateFirmware(BSTR FirmwareFile)

To upgrade the firmware. To use this function, obtain the firmware from the technical support engineers of our company beforehand.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| FirmwareFile | BSTR | [in] | Name of the firmware file to be upgraded (including the path) |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### BeginBatchUpdate

VARIANT\_BOOL BeginBatchUpdate(LONG dwMachineNumber, LONG UpdateFlagl)

To get ready for uploading data in batches. For example, if you invoke this function before uploading data such as fingerprint templates or user information, the SDK will temporarily store the data in the buffer. Then you can invoke BatchUpdate to upload the data to the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| UpdateFlagl | LONG | [in] | Whether to overwrite the original fingerprint template |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The UpdateFlag1 parameter specifies whether to overwrite the original fingerprint template if a fingerprint with the same index already exists when you upload a fingerprint template. The value 1 indicates overwriting the original fingerprint template and 0 indicates not overwriting the original fingerprint template.

Note

Applicable to B&W, TFT, and IFACE devices

#### BatchUpdate

VARIANT\_BOOL BatchUpdate(LONG dwMachineNumber)

To upload data in batches. Generally you are advised to invoke BeginBatchUpdate to upload the data to the buffer before invoking this function.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### CancelBatchUpdate

VARIANT\_BOOL CancelBatchUpdate(LONG dwMachineNumber)

To cancel bulk data uploading. You can invoke this function after invoking BeginBatchUpdate but before invoking BatchUpdate. This function aims to release the buffer allocated for bulk data uploading.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### PlayVoice

VARIANT\_BOOL PlayVoice(LONG Position, LONG Length)

To play announcements with specified consecutive indexes. The indexes depend on the machine. You can view the indexes, which range from 0 to 11, in voice testing on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| Position | LONG | [in] | Index of the start announcement |
| Length | LONG | [in] | Index of the end announcement |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### PlayVoiceByIndex

VARIANT\_BOOL PlayVoiceByIndex(LONG Index)

To play an announcement with the specified index. The index depends on the machine. You can view the index, which ranges from 0 to 11, in voice testing on the machine.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| Index | LONG | [in] | Index of the announcement to be played |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W, TFT, and IFACE devices

#### ReadAttRule

VARIANT\_BOOL ReadAttRule(LONG dwMachineNumber)

To read the attendance rule of the machine. This function supports OP1000.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W devices

#### SaveTheDataToFile

VARIANT\_BOOL SaveTheDataToFile(LONG dwMachineNumber, BSTR TheFilePath, LONG FileFlag)

To save the data in the buffer as a file. This function supports OP1000.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| TheFilePath | BSTR | [in] | Path for saving the file |
| FileFlag | LONG | [in] | File type flag |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The FileFlag parameter specifies the file type. The values are described as follows:  
1 Attendance records  
2 User  
3 Attendance rule  
4 Department list  
5 Shift

Note

Applicable to B&W devices

#### ReadTurnInfo

VARIANT\_BOOL ReadTurnInfo(LONG dwMachineNumber)

To read the shift information on the machine. This function supports OP1000.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

Applicable to B&W devices

#### SSR\_OutPutHTMLRep

VARIANT\_BOOL SSR\_OutPutHTMLRep(LONG dwMachineNumber, BSTR dwEnrollNumber, BSTR AttFile, BSTR UserFile, BSTR DeptFile, BSTR TimeClassFile, BSTR AttruleFile, LONG BYear, LONG BMonth, LONG BDay, LONG BHour, LONG BMinute, LONG BSecond, LONG EYear, LONG EMonth, LONG EDay, LONG EHour, LONG EMinute, LONG ESecond, BSTR TempPath, BSTR OutFileName, LONG HTMLFlag, LONG resv1, BSTR resv2)

To generate the attendance report of a user within the specified time period in HTML format. This function supports OP1000.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine ID |
| dwEnrollNumber | BSTR | [in] | User ID |
| AttFile | BSTR | [in] | Name of the attendance record file |
| UserFile | BSTR | [in] | Name of the user information file |
| DeptFile | BSTR | [in] | Name of the department information file |
| TimeClassFile | BSTR | [in] | Name of the shift information file |
| AttruleFile | BSTR | [in] | Name of the attendance rule file |
| BYear | LONG | [in] | Start time of the time period |
| BMonth | LONG | [in] | End time of the time period |
| BDay | LONG | [in] | Start time of the time period |
| BHour | LONG | [in] | Start time of the time period |
| BMinute | LONG | [in] | Start time of the time period |
| BSecond | LONG | [in] | Start time of the time period |
| EYear | LONG | [in] | End time of the time period |
| EMonth | LONG | [in] | End time of the time period |
| EDay | LONG | [in] | End time of the time period |
| EHour | LONG | [in] | End time of the time period |
| EMinute | LONG | [in] | End time of the time period |
| ESecond | LONG | [in] | End time of the time period |
| TempPath | BSTR | [in] | Path where other exception files are saved |
| OutFileName | BSTR | [in] | Attendance report file name (including the path) |
| HTMLFlag | LONG | [in] | HTML report format |
| resv1 | LONG | [in] | Reserved |
| resv2 | BSTR | [in] | Flow report name |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

The HTMLFlag parameter specifies the HTML report type. The values are described as follows:  
1 Flow report  
2 Exception report  
3 Statistics report

Note

Applicable to B&W devices

#### SetCommuTimeOut

VARIANT\_BOOL SetCommuTimeOut(LONG timeout)

Set the timeout for communication.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| timeout | LONG | [in] | Timeout of seconds. |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Note

No dependency with device.

#### SendFileByType

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine equipment number |
| FileName | BSTR | [in] | File full path name |
| iType | LONG | [in] | File type |

#### SetCommProType

VARIANT\_BOOL SetCommProType(LONG, proType)

Set the priority to use PULL or Standlone SDK to connect the device.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| proType | LONG | [in] | Priority protocol type |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

proType=1 Priority to use the Standlone SDK protocol proType=2 Priority to use the PULL SDK protocol Default to use Standlone SDK protocol to connect devices. If the customer does not develop PULL devices, this interface is not recommended by the developer.

Note

No dependency with device.

#### SetCommProType

VARIANT\_BOOL SetCommProType(LONG proType)

Set the priority to use PULL or Standlone SDK to connect the device.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| proType | LONG | [in] | Priority protocol type |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

proType=1 Priority to use the Standlone SDK protocol proType=2 Priority to use the PULL SDK protocol Default to use Standlone SDK protocol to connect devices. If the customer does not develop PULL devices, this interface is not recommended by the developer.

Note

No dependency with device.

#### GetConnectStatus

VARIANT\_BOOL GetConnectStatus(LONG\* dwErrorCode)

Get the error code return from the function **[BatchUpdate()](mk:@MSITStore:C:\\Users\\Hess\\AppData\\Local\\kingsoft\\WPS Cloud Files\\userdata\\qing\\filecache\\.347219938\\cachedata\\873AD516F4394C9897BB284E7BA00A2A\\SDKAPI-en.chm::/6_8_other_8h.html" \l "a2378e2b9425960af13c79655ee1ae9d2" \o "To upload data in batches. Generally you are advised to invoke BeginBatchUpdate to upload the data to...)**

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwErrorCode | LONG\* | [out] | Error code |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

When (dwErrorCode >= -11004 && dwErrorCode <= -10004), the software will disconnect and re-connect the device.

Note

Only used to get the error code return from the function **[BatchUpdate()](mk:@MSITStore:C:\\Users\\Hess\\AppData\\Local\\kingsoft\\WPS Cloud Files\\userdata\\qing\\filecache\\.347219938\\cachedata\\873AD516F4394C9897BB284E7BA00A2A\\SDKAPI-en.chm::/6_8_other_8h.html" \l "a2378e2b9425960af13c79655ee1ae9d2" \o "To upload data in batches. Generally you are advised to invoke BeginBatchUpdate to upload the data to...)**, without the limitation of devices.

#### SetDeviceTableData

VARIANT\_BOOL SendFileByType(LONG dwMachineNumber, BSTR TableName, BSTR Datas, BSTR OptionsLONG\* Count)

This function applies only to the time and attendance applications that comply with the PULL protocol in the new firmware. Used to set and insert data (such as time segments, user information, and leaves settings) into a device. The data can be one or more records. If the primary key of an inserted record already exists in the device, the original record is overwritten.

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| dwMachineNumber | LONG | [in] | Machine equipment number |
| TableName | BSTR | [in] | Data table name |
| Datas | BSTR | [in] | Data record representation |
| Options | BSTR | [in] | Default empty, use for expansion |
| Count | LONG\* | [out] | The number of records which were issued successfully |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

Indication of data records. Data is in the text format. Multiple records are separated by \r\n. Various pairs of "field=value" are separated by \t.

Note

#### SearchDevice

VARIANT\_BOOL SearchDevice(BSTR commType, BSTR address, BSTR\* DevBuffer, LONG DevBufferSize)

Use web search devices.

Parameters

Parameter description:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **param direction** | **description of param** |
| commType | BSTR | [in] | Communication type |
| address | BSTR | [in] | Communication data |
| DevBuffer | BSTR\* | [out] | Device list |
| DevBufferSize | LONG | [in] | Size of device list |

Returns

Value description:

|  |  |  |
| --- | --- | --- |
| **name** | **type** | **description of value** |
| True | BOOL | Function execution success |
| False | BOOL | Function execution failure |

See also

Attention

commType=UDP  
address=255.255.255.255 The return data include the IP address, serial number, firmware version, etc.

Note

Applicable to some customized devices

# FAQs

## How to Download Attendance Records?

First, use ReadGeneralLogData to read all attendance records and write them into the memory. Then, use GetGeneralLogData repeatedly to obtain attendance records. When GetGeneralLogData returns False, it means that all attendance records are obtained. Then, you can write the obtained records into database or display them in other forms to finish downloading. You can follow the same steps to down operation records.

## How to Create a User Online?

First, use SetuserInfo to write user information (such as enrollment number, password, and name) into the device. Then, use SetUserTmpStr/SetUerTmp/SetEnrollDataStr/SetEnrollData to set fingerprint templates for the user. This method improves enrollment efficiency and is suitable when user information has been collected and stored in media such as database.

To upload user information and corresponding fingerprint templates in batches, use BeginBatchUpdata or SetUserInfo together with SetUserInfo, BarchUpdata, EnableDevice, or RefreshData. For details, see demo program.

## How to Import or Download Data from USB Disk?

Many existing offline products support data download from USB disks. Many customers are concerned about data formats of USB disks. As the downloaded data formats are complex, ZKTeco provides a tool for importing data from USB disks to database. Database is open for customers to download data. In addition, ZKTeco also provides examples on how to process data files (\*.dat, etc.) collected from USB disks and how to write the data into specified data files. All structs adopt 1 byte alignment mode.

Data in USB disks include user information, fingerprint templates, face templates, attendance records, and short messages. Detailed data structures are used in demo program. They are described briefly below:

User data structure:

typedef struct \_User\_{

U16 PIN; //Internal number of a user

U8 Privilege;

char Password[8];

char Name[24];

U8 Card[4];

U8 Group;

U16 TimeZones[4];

char PIN2[24]; //User ID

}

Data structure of 9.0 fingerprint template:

typedef struct \_Template\_{

U16 Size; //fingerprint template length

U16 PIN; //internal user ID, which corresponds to PIN2 in user table

BYTE FingerID; // fingerprint backup index

BYTE Valid;

BYTE Template[MAXTEMPLATESIZE]; //maximize template length

} //MAXTEMPLATESIZE 602 Bytes

Data structure of template.fp10:

typedef struct \_Template\_{

U16 Size; //entire structure data size

U16 PIN; //user ID

BYTE FingerID; //fingerprint index

BYTE Valid; //flag

BYTE \*Template; //template

}

Attendance record,

struct \_AttLog\_{

U16 PIN;

U8 PIN2[24];

BYTE verified;

time\_t time\_second;

BYTE status;

U32 workcode;

BYTE reserved[4];

}

Exported as a text file:

attlog.dat format explanation:

segment:

BadgeNumber(employee number),

checktime,

DeviceID,

checktype(check status),

VerifyCode(verification ways:password or fingerprint or other)

Workcode

There is an Ascii code #9(Tab) between each segment. When development, move to the segment value you want to choose by "Tab".

If the device has the output data protection function, the serial number of the current device is displayed in first line and the hash value in the last line of the file to which attendance records are exported from USB disk.

Data structure of SMS

typedef struct \_SMS\_{

BYTE Tag; //category

U16 ID; //data content flag. 0 indicates that the record is invalid.

U16 ValidMinutes; //valid minutes. 0 indicates that the record is permanently valid.

U16 Reserved;

U32 StartTime; //start time

BYTE Content[MAX\_SMS\_CONTENT\_SIZE+1]; //short message content

} // MAX\_SMS\_CONTENT\_SIZE 60 Bytes

Data structure between SMS and user pin//user->sms,udata.dat

typedef struct \_UData\_{

U16 PIN; //0 indicates that the record is invalid

U16 SmsID;

}GCC\_PACKED TUData, \*PUData; //4Bytes

Data structure of face template:

typedef struct \_FaceTmp\_{

U16 Size;//face template size

U16 PIN;//user ID

BYTE FaceID;//Face ID

BYTE Valid;//flag

U16 Reserve;//reserve

U32 ActiveTime;

U32 VfCount;//Verify Count

BYTE FaceTmp[FACE\_TMP\_MAXSIZE]

} //FACE\_TMP\_MAXSIZE=1024\*2+512

## How to Use Biokey to Write the Collected Fingerprint Templates Offline?

When a fingerprint is collected, Biokey usually obtains the fingerprint template during enrollment. For example, the currently enrolled fingerprint template can be obtained via OnEnroll. After obtaining the fingerprint template, Biokey converts it into an offline fingerprint template. Then, the template can be written into the device.

## How to Obtain All Information of All Users?

Use ReadAllUserID to read IDs of all users and write them into memory. Then, use GetAllUserID repeatedly to obtain EnrollNumber of users, and use GetUserInfo to obtain user information. If necessary, you can also use GetUserTmpStr to obtain the fingerprint templates in string form.

## How to Connect to the Device?

During connection, the device can be regarded as an independent PC. However, the IP address of the device must match the IP address to be connected. Some devices, for example F4, support serial port connection and network connection. Therefore, during different connections, you need to set the device differently, modify communication mode, and set the controller switch to TCP/IP or RS232/485. Otherwise, the connection may fail. Sometimes, if the device fails to be connected due to busy serial ports, you can restart the program. If the application keeps connecting to the device without being manually disconnected, you can use DisableDeviceWithTimeOut to set the automatic disconnection time of the device. If some connections are used to download or modify data via serial ports or network, you can use EnableDevice to keep the device working and restore the connections after communication finishes, so as to maintain data consistency and avoid unknown errors.

## Password Is Invalid After SetUserInfo Is Used.

After SetUserInfo is called, Password may be set to null. If so, password verification will fail. To keep the password unchanged when writing user information, use GetUserInfo to obtain user password and transmit the password value to the Password parameter of SetUserInfo before using SetUserInfo.

## How to Convert an Online Template into an Offline Template?

Use FPTempConvertNew to convert the collected templates into offline fingerprint templates. See related descriptions of Biokey SDK for how to obtain the templates collected by Biokey. FPTempConvertNew is used to convert binary fingerprint templates. Parameters temp1 and temp2 are binary parameters. You can also use FPTempConvertNewStr to convert Biokey fingerprint templates of string type into offline fingerprint templates.

## Demo Program Fails to Connect to the Device

After the attendance management program is installed, users can connect to the device by using the attendance management program, but cannot connect to the device by using demo program. The reason is that DLL is copied to the directory of the attendance management program but registered in the installation directory during program installation. Generally, SDK loads controls from the system directory. Therefore, if the SDK version in the system directory is different from that in the attendance software directory, conflicts occur. (DLL function addresses of different versions are different, but OCX functions are the same in programming. Therefore, the problem is found only during runtime.)

**Caution: The common procedure for registering the SDK in the system is as follows:**

1. If an SDK has been already registered in the system, run **regsvr32 /u zkemkeeper.dll** to unregister the original SDK.

2. Copy all DLLs to the system directory, for example, win2000 is located in winnt\system32.

3. Run regsvr32 "registration path\zkemkeeper.dll" to register the SDK.

4. Correctly load controls in development environment (learn the usage of development tool by yourself. Relevant details are omitted here).

5. Try to use the SDK of the same version in development or running environment.

## Offline Fingerprint Device Keeps Working After Being Connected

After connecting the SDK to the offline fingerprint device, use EnableDevice to keep the offline fingerprint device working (see EnableDevice), so as to maintain data consistency and avoid unknown errors. After the offline fingerprint device is working, the keypad and fingerprint sensor stop working. After communication finishes, disconnect the SDK from the device or use EnableDevice again to restore the offline fingerprint device to normal state.

DisableDeviceWithTimeOut is recommended.

## Illegal Characters Are Displayed or Screen Display Is Abnormal After Non-English Names or Short Messages Are Uploaded to the Device

First, check whether the device supports the specified language. For example, if the current language of the device is English, but an Arabic name is uploaded to the device, the name cannot be displayed normally. If the device already supports the language, but the name still cannot be displayed, use related functions to convert the user name into UTF-8 format (for example, use AnsiToUTF8() in Dephi), and then use SetUserInfo to upload the user name.

## Card Management Problems

How to register a card in the device? How to obtain the user card?

The SDK has the cardnumber parameter. If this parameter is invisible in development environment, use GetStrCardNumber and SetStrCardNumber instead.

For a user enrolled in the device, the card number is a kind of user information. When SetUserInfo is used to set user information, cardnumber is automatically used as the card number and set for the user.

The procedure for registering a card is as follows:

Set cardnumber -> Upload user information

The procedure for obtaining the card number of a user is as follows:

Obtain information of the specified user -> Obtain cardnumber

Note: The card number is internally defined as four unsigned bytes of long type. If VB does not support four unsigned bytes, verification can be started after the last three bytes of the card number are input (if the last three bytes are different from each other).

## Firewall or Router Traversal

In most cases, the device to be connected needs to traverse firewalls or routers, and UDP socket and port 4370 are used for SDK communication. Therefore, UDP and port 4370 must be enabled on firewalls or routers. If the device traverses gateways via port mapping, the device can be accessed via port number and IP address of routers. Generally, if UDP and port 4370 are enabled and can be pinged, the device can be connected. Certainly, in the case of data download, network connection must be considered. In addition, some devices that support SOAP ports can be accessed via embedded Web Server and SOAP.

Caution: The zem100 series products can traverse internet via port mapping. For zem200 products, as the devices run on Linux, they can be accessed after the gateway is configured if the local network environment supports gateway communication. Certainly, there are still some other methods for accessing the device, for example, VPN and IP mapping. The connection scheme should be selected according to specific network environments.

## Difference between ZKFinger10.0 and ZKFinger9.0 and Comparison between Templates

**Algorithm performance:** Compared with ZKFinger9.0, ZKFinger10.0 achieves better false acceptance rate (FAR), false rejection rate (FRR), and enrollment rejection rate (ERR), better image processing effect of low-quality fingerprints (for example, fingerprints are too dry or too wet, or users have worn or injury), and 10 times faster comparison.

**Template size:** The size of a ZKFinger10.0 fingerprint template is about 1.6 KB. The size of a ZKFinger9.0 fingerprint template is about 512B. When ZKFinger10.0 is used, a Mifare card with at least 2 KB capacity should be used for data storage.

**Template compatibility:** The ZKFinger10.0 and ZKFinger9.0 fingerprint templates are incompatible with each other. If a user who have already registered ZKFinger9.0 fingerprint templates wants to use ZKFinger10.0, the user has to register fingerprint templates again, and vice versa.

## Uploading a Large Capacity of Fingerprints

Large capacity usually means over 1500 fingerprints. Some devices can hold 8000 fingerprints or more. Fingerprint templates must be uploaded in batches. In this mode, the upload is much faster. For how to upload fingerprint templates in batches, see descriptions of batch process function.

## Differences between High-speed Upload and Ordinary Upload

In an ordinary upload, each time upload functions (such as SetUserinfo and SetUserTmpStr) are used, the SDK communicates with the device and uploads related data to the device.

In a high-speed upload, BeginBatchUpdata is used to create a temporary buffer to store the data to be uploaded in subsequent operations. Then, BatchUpdata can be used to upload all the data in the buffer to the device at a time. This mode greatly reduces communications between the SDK and the device, and raises the speed of large-capacity upload in particular.

## How to Determine Whether the Device Uses ZKFinger10.0 or ZKFinger9.0?

Use the following function:

VARIANT\_BOOL GetSysOption([in] LONG dwMachineNumber, [in] BSTR Option, [out] BSTR\* Value)

The Option parameter constantly "~ZKFPVersion". If the returned Value is 10, the device uses ZKFinger10.0. If the returned Value is 9 or null (null is returned as old TFT devices do not have this value), the device uses ZKFinger9.0.

For example:

zkem.GetSysOption(EmManth.EmMan.Dev.MachineNumber,'~ZKFPVersion',verSionFp);

If verSionFp='10', the device uses ZKFinger10.0

If verSionFp=‘9’ or verSionFp=‘’, the device uses ZKFinger9.0

## How to Upload, Download, and Delete ZKFinger10.0 Templates?

ZKFinger10.0 provides faster comparison, but the template size and storage mode are different from those of older versions:

When ZKFinger10.0 is used, the size of a fingerprint template is about 1.6 KB. When an older version is used, the size of a fingerprint template is smaller than 608B.

You can use follow function to upload,download and delete ZKFinger10.0 Templates:

VARIANT\_BOOL **SetUserTmpEx**([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFingerIndex,[in] LONG Flag, [in] BYTE\* TmpData)

VARIANT\_BOOL **GetUserTmpEx**([in] LONG dwMachineNumber, [in] BSTR

dwEnrollNumber, [in] LONG dwFingerIndex,[out] LONG \* Flag, [out] BYTE\* TmpData, [out] LONG\* TmpLength)

VARIANT\_BOOL **SetUserTmpExStr**([in] LONG dwMachineNumber, [in] BSTR

dwEnrollNumber, [in] LONG dwFingerIndex,[in] LONG Flag, [in] BSTR TmpData)

VARIANT\_BOOL **GetUserTmpExStr**([in] LONG dwMachineNumber, [in] BSTR

dwEnrollNumber, [in] LONG dwFingerIndex,[out] LONG \* Flag, [out] BSTR\* TmpData, [out] LONG\* TmpLength)

The above four functions can upload and download both ordinary fingerprint templates (Flag=1) and threatened fingerprint templates (Flag=3). Additionally, they can be used for both ZKFinger10.0 templates and ZKFinger9.0 templates.

To delete ZKFinger10.0 templates, you can use the following functions (these two functions are used on TFT devices to delete fingerprint templates):

VARIANT\_BOOL **SSR\_DelUserTmp**([in] LONG dwMachineNumber, [in] BSTR EnrollNumber, [in]LONG dwFingerIndex)

VARIANT\_BOOL **SSR\_DelUserTmpExt**([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in]LONG dwFingerIndex)

Note:

Ver6.60 is the internal version of device firmware. You can obtain it by using GetFirmwareVersion after SDK connects to the device (or by using the attendance software). Caution: The internal version obtained is different from the firmware version that you view on the device.

## How to Upload, Download, and Delete ZKFinger9.0 Templates?

The functions used to upload and download ordinary ZKFinger9.0 templates (that is, SSR\_GetUserTmp, SSR\_GetUserTmpStr, SSR\_SetUserTmp, and SSR\_SetUserTmpStr) in earlier SDK versions are continuously used in the present SDK version.

New functions (that is, SetUserTmpEx, GetUserTmpEx, SetUserTmpExStr, and GetUserTmpExStr) are also added in the present SDK version. These functions can upload and download both ordinary fingerprint templates (Flag=1) and threatened fingerprint templates (Flag=3). But they are used only on the TFT devices with firmware Ver6.60 or later. Additionally, they can be used for both ZKFinger10.0 templates and ZKFinger9.0 templates.

To delete ZKFinger9.0 templates, you can use SSR\_DelUserTmp or SSR\_DelUserTmpExt.

## How to Download a Face Template?

1. The transmission mode of face templates is the same as that of ZKFinger10.0 fingerprint  
templates.  
2. One user has about 15 face templates in different angles. Each template consists of 2576 bytes.  
The third and fourth bytes of each template indicate the ID, corresponding to the first and second  
bytes of user structure. For devices that support face identification, the last 24 bytes of user  
structure indicate the user ID. Therefore, the total size of face templates of each user is about 37  
KB. You are not recommended to upload or download data via serial ports. When the value of  
dwFaceIndex is 50, all face templates of a user are uploaded or downloaded.  
3. To upload and download ZKFace templates, use the following functions:  
SetUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BYTE\*  
TmpData, LONG TmpLength, VARIANT\_BOOL\* pVal);  
GetUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex,  
BYTE\* TmpData, LONG \* TmpLength, VARIANT\_BOOL\* pVal);  
DelUserFace (LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex,  
VARIANT\_BOOL\* pVal);  
For details, see description of the functions.

## A description of the special parameters of GetSysOption

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1,BiometricType Get support for biometric types** | | | | | | | | | |
| Each represents a type of biometric type by retrieving the 8-bit numeric string（0 representative no support;1 representative Support;For example, BiometricType= 01100,000, indicating that the device supports fingerprints and faces） | | | | | | | | | |
| Character index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| On behalf of the type | General purpose | Fingerprint | Face | Voiceprint | Iris | The retina | Palmprint | Refers to the vein | Palm vein | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2,BiometricVersion** Gets a version of the supported biometric type | | | | | | | | | |
| A version number of a biometric type supported by the current device, which is separated by ":", is returned by obtaining this parameter（For example,:BiometricVersion=0:10.0:7.0:::::,The device supports fingerprint 10.0 and face 7.0） | | | | | | | | | |
| Character index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| On behalf of the type | General purpose | Fingerprint | Face | Voiceprint | Iris | The retina | Palmprint | Refers to the vein | Palm vein | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3,BiometricMaxCount** Obtain supported biometric type data capacity | | | | | | | | | |
| The data capacity of the biometric type supported by the current device, separated by ":", is returned by obtaining this parameter（For example,:BiometricMaxCount=0:3000:1000:::::,The equipment supports 3000 fingerprints and 1000 faces） | | | | | | | | | |
| Character index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| On behalf of the type | General purpose | Fingerprint | Face | Voiceprint | Iris | The retina | Palmprint | Refers to the vein | Palm vein | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4,BiometricUsedCount** Gets the number of supported biometric type data | | | | | | | | | |
| Returns the number of biometric type data supported by the current device separated by this parameter（For example,:BiometricUsedCount=0:100:10:::::,The device has 100 fingerprints and 10 faces） | | | | | | | | | |
| Character index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| On behalf of the type | General purpose | Fingerprint | Face | Voiceprint | Iris | The retina | Palmprint | Refers to the vein | Palm vein | |