ReaderSDK DLL reference.

Questo SDK permette l'utilizzo delle principali funzionalità del software ScreenReader o FutuReader, e dei relativi hardware di lettura per i test rapidi Screenitalia e Futurlab, all'interno dei propri programmi o come mezzo di integrazione con gli stessi. ReaderSDK è scaricabile al seguente link http://www.screenreader.it/downloads/screenreader/ReaderSDK.zip

bool ReaderSDK::InitEngineSDK(string optional_priority_name)
Inizializza il motore SDK, se è tutto ok ritorna true.
Il valore passato 'optional_priority_name' è opzionale e può essere omesso nel caso nel computer esista una sola installazione del FutuReader o dello ScreenReader, ma nel caso siano presenti entrambi i software nella stessa macchina occorrerà specificare quale dei 2 pilotare popolando la stringa optional_priority_name con la parola " FutuReader" o "ScreenReader" a secondo delle proprie esigenze.
Anche nel caso che il software sia stato installato a mano in una differente directory rispetto quella di default, bisognerà impostare la variabile optional_priority_name, questa volta con il percorso completo del software da pilotare (es: "C:\Users\Public\ScreenReader\ScreenReader.exe" oppure "C:\Users\Public\FutuReader\FutuReader.exe").

string ReaderSDK::GetReaderFileName()
Torna il nome del file eseguibile del Reader (ScreenReader.exe o FutuReader.exe).

string ReaderSDK::GetReaderFilePath()
Torna il percorso dell'eseguibile del software Reader.

bool ReaderSDK::ExecuteReaderSoftware(bool show_reader, bool bool wait_main_msg_ready, bool hook_running_reader)
Esegue il software Reader (FutuReader.exe o ScreenReader.exe) di lettura dei test rapidi.
Una volta eseguito e verificato che sia pronto torna true se tutto ok, la funzione è sincrona.
Se il valore show_reader è omesso oppure è impostato a true il software Reader sarà visibile, nel caso invece sia impostato a false appena eseguiti i controlli di licenza e aggiornamento, l'interfaccia del software sarà nascosta.
Il valore wait_main_msg_ready omesso o impostato a true fa si che la funzione attenda prima di ritornare che la main window del software Reader sia in grado di processare altri comandi remoti.
Il valore hook_running_reader viene utilizzato per abilitare le funzioni SDK in un software Reader già precedentemente in esecuzione, ciò permette di agganciarsi al software Reader anche se lo stesso fosse stato eseguito manualmente o comunque già in esecuzione prima dell'avvio del SDK.
Per default hook_running_reader è impostato a true, cosi in automatico, nel caso ci fosse in esecuzione un Reader lo aggancia, altrimenti ne esegue una nuova istanza.

bool ReaderSDK::CloseReaderSoftware(int max_sec_timeout)
Chiude il software Reader di lettura, una volta eseguito torna true se l'applicazione è stata chiusa con successo. Nel caso il software remoto sia stato lanciato con interfaccia nascosta, è importante non dimenticare di usare questa funzione altrimenti rimarrebbe aperto in background ed impedirebbe il suo utilizzo fino alla sua chiusura attraverso il task manager o un riavvio della macchina. Potrebbe essere utile inserire questa funzione nell'evento di chiusura dell'applicazione che utilizza l'SDK.
La variabile max_sec_timeout imposta il numero massimo di secondi che sdk attenderà affinché avvenga una chiusa standard del software Reader, dopo di che forzerà la chiusura (**NB**: la chiusura forzata non garantisce il flush del DB delle anagrafiche nel caso in cui nella stessa sessione di lavoro si fossere apportate modifiche alle anagrafiche stesse) .
Tale funzione può essere utilizzata anche per chiudere un instanza del Reader precedentemente in esecuzione prima dell'esecuzione del SDK.

int ReaderSDK::GetAvailableTestCount()
Torna il numero di tutti i test che il lettore può leggere. Servirà per fare un loop insieme alla funzione "GetTestName(int test_index)" per reperire il nome di ogni singolo test leggibile dal lettore.

string ReaderSDK::GetTestName(int test_index)
Torna il nome del test alla posizione test_index, test_index è un valore compreso da 0 (per il primo elemento) a GetAvailableTestCount()-1 (per l'ultimo).
Se si imposta test_index uguale a -1 la funzione ritorna la lista completa di tutti i test rapidi supportati con separatore '\n'.

string ReaderSDK::GetActiveScanner()
Torna il nome della periferica utilizzata come lettore ottico (ScreenBox, OpticSlim 500,OpticSlim 500. Se non reperibile (come ad esempio ScreenBox spento) torna stringa vuota.

bool ReaderSDK::SetActiveScanner(string optical_scanner)
Imposta il lettore ottico da utilizzare per la lettura dei tests rapidi.

bool ReaderSDK::CalibrateBox(string scanner_name)
Esegue la procedura di calibrazione del lettore denominato in *scanner_name.*
Valori possibili per i lettori di tipo Box sono "ScreenBox L", "ScreenBox S", "ScreenBox V" mentre per i lettori di tipo scanner piano sono "Plustek OpticSlim 500", "Plustek OpticSlim 500+", "Plustek OpticSlim 2610-TWAIN" e "AVA5 Plus 11.11" ecc...).
NB: Omettendo il parametro *scanner_name : CalibrateBox()* selezionerà in automatico l'ultimo lettore usato ed attualmente collegato (scelta consigliata).

string ReaderSDK::GetActiveTest()
Torna il nome completo del test attualmente impostato e che sarà letto durante l'esecuzione di StartScanTest().

bool ReaderSDK::SetActiveTest(string test_name)
Imposta il test che sarà utilizzato al momento della lettura attraverso l'esecuzione di StartScanTest().
Torna true se è stato correttamente impostato. Per avere un elenco e completo e preciso nella denominazione dei tests usare le funzioni GetAvailableTestCount() e GetTestName(int test_index).

string ReaderSDK::GetListOperatorsNames()
Torna la lista dei nomi di tutti gli operatori presenti nel database.
Come separatore dei nomi è usato '\n' .

bool ReaderSDK::AddNewOperatorRecord(string _data)
Con questa funzione si può aggiungere al database un nuovo operatore con alcuni parametri di base.
Il campo _data va popolato con una testo nello stile XML come ad esempio nel seguente modo:
<operatore nome="Dr Luigi Balestrin" luogo="Roma" tester="Screen 7+AD (DUA-174)" lotto="21547" />
NB: Attualmente i campi sopra indicati (**nome, luogo, tester, lotto**) sono i soli a poter essere impostati tramite SDK.
Gli altri campi (*scanner, path_verb,frontespizio_verb,reduce_font_size,creatinina_verb,firma_verb,firma_responsabile,firma_da_immagine, show_quantita_campione_verb,quantita_campione_verb,show_note_alcol,note_alcol,show_campi_personalizzati, show_temperatura_campione_verb,temperatura_campione_verb,show_farmaci_assunti_verb,farmaci_assunti_verb,codice_paziente_verb, qualifica_soggetto_verb,azienda_soggetto_verb,data_nascita_verb,doc_identita_verb,foto_test_verb,stampa_dichiarazione,stampa_note, dichiarazione_paziente_verb,note_verb*) vengono creati automaticamente con i valori di default.

bool ReaderSDK::RemoveOperatorByName(string operator_name)
Rimuove un paziente e la sua anagrafe dal database attraverso l'identificazione del nominativo.
ATTENZIONE: Insieme all'operatore verranno cancellati anche tutti i pazienti/individui (e i relativi dati anagrafici) associati ad esso.

bool ReaderSDK::SetActiveOperatorByName(string operator_name)
Imposta come attivo il record dell'operatore che corrisponde al nominativo *operator_name* caricando le relative informazioni.

string ReaderSDK::GetListIndividualsNames()
Torna la lista dei nomi di tutti i pazienti dell'operatore attivo.
Come separatore dei nomi è usato '\n' .

string ReaderSDK::GetActiveIndividualName()
Torna il nome del paziente che è impostato per la lettura attraverso l'esecuzione di StartScanTest().

string ReaderSDK::GetActiveIndividualDocument()
Torna il documento di identità del paziente che è impostato per la lettura attraverso l'esecuzione di StartScanTest().

bool ReaderSDK::SetActiveIndividualByName(string individual_name)
Imposta come attivo il record del paziente che corrisponde al nominativo individual_name caricando le relative informazioni anagrafiche che serviranno per la redazione del verbale in seguito alla lettura del test attraverso l'esecuzione di StartScanTest().

bool ReaderSDK::SetActiveIndividualByDocument(string _document)
Imposta come attivo il record del paziente che corrisponde al documento di identità _document caricando le relative informazioni anagrafiche che serviranno per la redazione del verbale in seguito alla lettura del test attraverso l'esecuzione di StartScanTest().

bool ReaderSDK::StartScanTest()
Avvia la procedura di lettura del test, che se va a buon fine torna true e si potrà poi procedere al reperimento dei risultati attraverso l'utilizzo di GetFullPathPdfFileResult() , GetFullResultText(), GetImageResultHBITMAP() (o in alternativa a quest'ultima GetImageResultData() e GetImageResultDataSize()). Si rimanda alla descrizione di queste funzioni per il loro utilizzo nel dettaglio.
Nota che il valore di ritorno di questa funzione non è relazionato alla corretta lettura del test, ma alla corretta esecuzione della procedura che avvia la lettura. Nel caso ci siano problemi di lettura del test (es: test inserito errato o mancante o altro) lo si potrà facilmente dedurre dalla mancanza dei risultati di ritorno nelle funzioni sopra citate.

string ReaderSDK::GetFullPathPdfFileResult()
Torna il percorso completo del nome del file del risultato (verbale pdf) del test precedentemente letto attraverso la funzione StartScanTest().

string ReaderSDK::GetFullResultText()
Torna una stringa che contiene sostanzialmente il risultato (verbale in pdf) del test precedentemente letto attraverso la funzione StartScanTest() in forma di puro testo.

int ReaderSDK::GetCountResultImages()
Nel caso in cui il test appena letto abbia più facciate, torna il numero delle immagini relative, tale valore può essere usato in associazione con le funzioni GetImageResultHBITMAP(int index_image), GetImageResultData(int index_image) e GetImageResultDataSize(int index_image).

HBITMAP ReaderSDK::GetImageResultHBITMAP(int index_image)
Torna un handle alla bitmap dell'immagine del test che è stata utilizzata per elaborare il risultato del test precedentemente letto attraverso la funzione StartScanTest().
Nel caso in cui il test abbia più facciate e quindi più immagini index_image viene utilizzato per specificare quale, index_image è opzionale nel caso ci sia una sola immagine. GetCountResultImages() invece può essere utilizzata per tornare quante immagini contiene il test appena letto.
Vedi anche GetImageResultData() e GetImageResultDataSize() che possono essere usati come alternativa.

unsigned char *ReaderSDK::GetImageResultData(int index_image)
Torna un puntatore di tipo unsigned char dell'immagine bitmap del test che è stata utilizzata per elaborare il risultato del test precedentemente letto attraverso la funzione StartScanTest().
Nel caso in cui il test abbia più facciate e quindi più immagini index_image viene utilizzato per specificare quale, index_image è opzionale nel caso ci sia una sola immagine. GetImageResultDataSize() può essere utilizzata per tornare la dimensione della bitmap.
Es: *BYTE *img=ReaderSDK::GetImageResultData();*
unsigned int img_size=ReaderSDK::GetImageResultDataSize();
SaveBufferToFile("ResultImage.bmp",img,img_size);

unsigned int ReaderSDK::GetImageResultDataSize(int index_image)
Torna le dimensione in byte della bitmap dell'immagine del test che è stata utilizzata per elaborare il risultato del test precedentemente letto attraverso la funzione StartScanTest().
Nel caso in cui il test abbia più facciate e quindi più immagini index_image viene utilizzato per specificare quale, index_image è opzionale nel caso ci sia una sola immagine. Per utilizzo con GetImageResultData() vedi esempio nella relativa descrizione della funzione.

long ReaderSDK::AddNewIndividualRecord(string _data)

Con questa funzione si può aggiungere un nuovo paziente al database completo della sua anagrafica ed eventuali campi aggiuntivi.
Il campo _data va popolato con una testo nello stile XML come ad esempio nel seguente modo:
<paziente nome="Mario Rossi" qualifica="Manager" azienda="FCA Italy S.p.A" data_nascita="12/10/1985" doc_identita="RSSMRA85T10A562S" codice_paziente="" campi_personalizzati="email|mario.rossi@esempio.com||tel|3215487||Indirizzo|Via del Monte 32||sesso|M" />
L'ordine non è obbligatorio ma i campi *nome, qualifica, azienda, data_nascita, doc_identita,codice_paziente e campi_personalizzati* devono comunque essere impostati, nel caso in cui non vengo utilizzati possono essere settati come vuoti ad eccezione del nome che contiene il nome e cognome del paziente quindi si può aggiungere un paziente con il minimo dei dati (cioè il solo nominativo) ad esempio nel seguente modo:
<paziente nome="Mario Rossi" qualifica="" azienda="" data_nascita="" doc_identita="" codice_paziente="" campi_personalizzati="" />
Si consiglia, ad ogni modo, di inserire oltre al nominativo anche un documento di identità univoco attraverso il campo *doc_identita* questo permette al software di selezionare e salvare correttamente pazienti che abbiano lo stesso nome e cognome, la necessità di questo campo non è comunque obbligatoria finché non si inseriscono pazienti con omonimia.
La stessa funzione permette di inserire più nuovi paziente contemporaneamente basta popolare la variabile _data con una stringa ad esempio nel seguente modo:
<paziente nome="Mario Rossi" qualifica="Manager" azienda="FCA Italy S.p.A" data_nascita="12/10/1985" doc_identita="RSSMRA85T10A562S" codice_paziente="" campi_personalizzati="email|mario.rossi@esempio.com||tel|3215487||Indirizzo|Via del Monte 32||sesso|M" />
<paziente nome="Ornella Pieri" qualifica="Segretaria" azienda="FCA Italy S.p.A" data_nascita="22/12/1979" doc_identita="PRIRLL79T62H501M" codice_paziente="1234" campi_personalizzati="email|ornella.pieri@esempio.com||tel|+393215487||Luogo di nascita|Roma||sesso|M" />
<paziente nome="Francesco Nuti" qualifica="operaio" azienda="ACME srl" data_nascita="01/01/1990" doc_identita="NTUFNC90A01F205X" codice_paziente="DB123" campi_personalizzati="email|francesco.nuti@esempio.com||tel|123456789||Luogo di nascita|Milano||vaccinato|si" />

Per quanto riguarda la voce ***campi_personalizzati*** può contenere campi/valori personalizzati nella forma *"campo1|valore1||campo2|valore2||campo3|valore3 … campoN|valoreN"*

bool ReaderSDK::RemoveIndividualByName(string _name)
Rimuove un paziente dal database compresa la sua anagrafe attraverso l'identificazione del nominativo.

bool ReaderSDK::RemoveIndividualByDocument(string _document)
Rimuove un paziente dal database compresa la sua anagrafe attraverso l'identificazione del documento di identità.

string ReaderSDK::GetIndividualRecordByName(string individual_name)
Torna una stringa in stile XML con i dati relativi al paziente selezionato attraverso l'identificato del nome.
Un esempio di stringa ritornata può essere:
<paziente nome="Mario Rossi" qualifica="Manager" azienda="FCA Italy S.p.A" data_nascita="12/10/1985" doc_identita="RSSMRA85T10A562S" codice_paziente="" campi_personalizzati="email|mario.rossi@esempio.com||tel|3215487||Indirizzo|Via del Monte 32||sesso|M" />

string ReaderSDK::GetIndividualRecordByDocument(string _document)
Torna una stringa in stile XML con i dati relativi al paziente selezionato attraverso l'identificato del documento di identità.
Un esempio di stringa ritornata può essere:
<paziente nome="Mario Rossi" qualifica="Manager" azienda="FCA Italy S.p.A" data_nascita="12/10/1985" doc_identita="RSSMRA85T10A562S" codice_paziente="" campi_personalizzati="email|mario.rossi@esempio.com||tel|3215487||Indirizzo|Via del Monte 32||sesso|M" />

string ReaderSDK::GetLastError()
Torna una descrizione, quando possibile, dell'ultimo errore accaduto. Potrebbe essere utile da richiamare subito dopo aver chiamato una funzione che abbia dato un risultato inatteso.

bool ReaderSDK::SetGenericFieldValue(string panel_name,string field_name,string text_value)
Questa funzione sperimentale, ti permette di impostare un valore in una qualunque casella di testo purché di tipo combo box , edit box, check box e memo.
Nella variabile panel_name va inserito il nome della finestra dove risiedono i controlli che oggetto delle modifiche, nella variabile field_name si passerà il nome del campo che vuoi compilare e nella text_value il valore che si vuol inserire.
Per ottenere un elenco preciso dei nomi dei controlli da utilizzare con questa funzione eseguire GetGenericFieldsDataInfo().

string ReaderSDK::GetGenericFieldsDataInfo()
Questa funzione, sperimentale, ritorna un testo di tipo xml con l'enumerazione delle informazioni dei controlli usabili in SetGenericFieldValue .
Sotto un esempio di ritorno della funzione:

```
<?xml version="1.0"?>
<Components>
    <Panel Name="frmMain" Caption="ScreenReader" >
        <fields>
            <field Name="cmbxEnumTwain" Type="text" Value=""/>
            <field Name="edtCodIdent" Type="text" Value=""/>
            …….
            <field Name="cmbxListDB" Type="text" Value="Screen 1 AMP (DAM-114)"/>
            <field Name="edtLotto" Type="text" Value="0001"/>
        </fields>
    </Panel>
    <Panel Name="frmExtraOptionRisultati" Caption="Opzioni aggiuntive del verbale" >
        <fields>
            <field Name="chkStampaFoto" Caption="Foto Test" Type="boolean" Value="true"/>
            <field Name="chkStampaDataNascita" Caption="Data di nascita" Type="boolean" Value="true"/>
            …….
            <field Name="chkFirmaOperatoreImmagine" Caption="Usa firma operatore da immagine" Type="boolean" Value="false"/>
           <field Name="edtFirmaOperatoreImmagine" Type="text" Value=""/>
        </fields>
    </Panel>
</Components>
```

In questo esempio se volessimo cambiare il valore del documento di identità basterà eseguire:
SetGenericFieldValue(" frmMain","edtCodIdent","NuovoDocIdentità")

oppure, nel caso di opzioni del verbale, se volessimo disabilitare la stampa della data di nascita:
SetGenericFieldValue(“ frmExtraOptionRisultati”,"chkStampaDataNascita",”false”)