SIGLA / START

Manuale tecnico

Rev01: 30/11/96 **Rev02:** 19/12/96 **Rev03:** 19/02/97 **Rev04:** 09/04/97 **Rev05:** 05/05/97 **Rev06:** 20/05/97 **Rev07:** 30/06/97 **Rev08:** 30/08/97 **Rev09:** 15/10/97 **Rev10:** 20/11/97 **Rev11:** 20/01/98 **Rev12:** 17/03/98 **Rev13:** 15/05/98 **Rev14:** 25/09/98 **Rev15:** 30/11/98 **Rev16:** 01/02/99 **Rev17:** 15/03/99 **Rev18:** 02/07/99 **Rev19:** 02/09/99 **Rev20:** 30/11/99 **Rev21:** 19/06/00 **Rev22:** 18/09/00 **Rev23:** 19/12/00 **Rev24:** 28/02/01 **Rev25:** 13/09/01 **Rev26:** 17/12/01 **Rev27:** 18/03/02 **Rev28:** 27/06/02 **Rev29:** 29/07/02 **Rev30:** 15/11/02 **Rev31:** 21/01/03 **Rev32:** 29/05/03 **Rev33:** 08/07/03 **Rev34:** 27/10/03 **Rev35:** 12/02/04 **Rev36:** 03/06/04 **Rev36:** 10/11/04 **Rev37:** 02/02/05 **Rev38:** 18/04/05 **Rev39:** 29/05/05 **Rev40:** 16/11/05 **Rev41:** 24/02/06 **Rev42:** 14/07/06 **Rev43:** 16/10/06 **Rev44:** 05/03/07 **Rev44:** 07/05/07 **Rev45:** 18/07/07 **Rev46:** 03/09/07 **Rev47:** 04/12/07 **Rev48:** 15/01/08 **Rev49:** 11/02/08 **Rev50:** 14/04/08 **Rev51:** 13/11/08 **Rev52:** 29/01/09 **Rev53:** 21/05/09 **Rev54:** 16/07/09 **Rev55:** 29/07/09 **Rev56:** 30/09/09 **Rev57:** 12/11/09 **Rev58:** 15/02/10 **Rev59:** 29/03/10 **Rev60:** 15/06/10 **Rev61:** 19/10/10 **Rev62:** 31/01/11 **Rev63:** 24/03/11 **Rev64:** 04/04/11 **Rev65:** 31/05/11 **Rev66:** 30/06/11 **Rev67:** 06/09/11 **Rev68:** 30/09/11 **Rev69:** 22/11/11 **Rev70:** 22/02/12 **Rev71:** 12/04/12 **Rev72:** 03/07/12

SIGLA: versioni disponibili

SIGLA versione 3 è attualmente rilasciato nelle seguenti versioni native:

- Windows 32 (Windows 95/98/Me, Windows NT/2000/Xp e Windows Vista[1]). N.B.: la procedura **NON è** certificata per l'uso dei drivers ODBC Microsoft a 32 bit per archivi flat (dBIII, FoxPro, Paradox etc.). **N.B.: per i clients Windows NT 3.51 NON è disponibile la funzione di telefonia assistita**
- Windows 32 (Windows 95/98/Me e Windows NT/2000/Xp) **Start Suite Multiuser** (START), applicazione record oriented che opera su files nativi FoxPro (DBF)
- Windows 32 (Windows 95/98/Me e Windows NT/2000/Xp) **Start Suite Client/Server,** versione client/server di START dotata di un motore per l'accesso alla base di dati Visual FoxPro (DBF)

SIGLA versione 4 è attualmente rilasciato nelle seguenti versioni native:

* Windows 32 (Windows Xp, Windows Vista e Windows 7[2]).

[1] SIGLA 3.10 ha ottenuto il logo "Certified for Windows Vista".

[2] SIGLA 4.2 ha ottenuto il logo "Compatible for Windows 7".

Prerequisiti Sistemistici

SIGLA v.3 I client su cui deve essere usato il prodotto devono avere le seguenti caratteristiche hardware/software:

* almeno 128 MB ram (consigliati 256MB)
* processore Pentium (o equivalente) o superiore
* porta parallela o porta USB
* sistema operativo Windows 95/98/Me, Windows NT/2000/Xp/Vista
* modem Hayes compatibile (se è installato il modulo 'Telefonia Assistita')
* scanner Twain (se sono installati i moduli 'Acquisizione A4' o 'Immagini Aziendali')
* MS Outlook o una qualunque email client compatibile con lo standard MAPI (per l'attivazione delle funzioni di mail se è installato il modulo Internet)
* browser Internet (se è installato il modulo Internet)

SIGLA v.4 I client su cui deve essere usato il prodotto devono avere le seguenti caratteristiche hardware/software:

* almeno 256 MB ram (consigliati 512MB)
* processore Pentium (o equivalente) o superiore
* porta USB
* sistema operativo Windows Xp/Vista/7 (con MS .Net Framework 3.5 SP1)
* modem Hayes compatibile (se è installato il modulo 'Telefonia Assistita')
* scanner Twain (se sono installati i moduli 'Acquisizione A4' o 'Immagini Aziendali')
* MS Outlook o una qualunque email client compatibile con lo standard MAPI (per l'attivazione delle funzioni di mail se è installato il modulo Internet)
* browser Internet (se è installato il modulo Internet)

Database Supportati

SIGLA utilizza lo standard ODBC per l'accesso ai dati. Poiché sono reperibili in commercio drivers ODBC per tutti i principali database commerciali il range di database manager teoricamente utilizzabili da SIGLA è estremamente vasto.

Durante la fase di sviluppo SIGLA viene testato sui seguenti database relazionali:

- DB2/UDB di IBM
- Oracle Database di Oracle
- SQLServer di Microsoft
- MySQL di MySQL AB[3]
- PostgreSQL[4]

Altri database manager (DBMS), seppur teoricamente utilizzabili, non sono ufficialmente impiegati come strumento di validazione e pertanto **non ne viene certificato l'utilizzo con SIGLA**.

Poiché SIGLA nasce per utilizzare l'architettura client/server **l'utilizzo di basi di dati non gestite da un DBMS (dbIII, Paradox, Access etc.), anche se tecnicamente possibile, non rientra tra le soluzioni certificate, e comunque sarebbe sconsigliato** per il degrado di performance che ne consegue.

SIGLA Start edition **è certificato** esclusivamente per l'utilizzo con Microsoft SQL Server 2005 express Edition.

SIGLA **non è certificato** per i drivers ODBC Microsoft a 32 bit (in particolare per i driver per archivi di tipo DB3, DB4, FoxPro e Access). Volendo usare archivi flat è quindi necessario utilizzare la versione 3 di SIGLA ed installare la speciale versione denominata Start Suite.

La versione **Start Suite** (START v3) lavora in modo nativo (non ODBC) su archivi e indici FoxPro ed è quindi utilizzabile quando non sia disponibile un DBMS.

I DBMS certificati vengono testati utilizzando sempre il driver ODBC **fornito dal produttore** stesso nell'apposito CDROM di distribuzione o con l'ultima versione disponibile qualora il produttore stesso ne indichi la necessità.

A titolo di riferimento si ricordano le versioni dei vari database manager utilizzati dal laboratorio di sviluppo per i test interni:

Database Manager	Versione
DB2/UDB	9.5 e 9.7
Oracle	10g
SQL Server	2005 e 2008
MySQL	5.0.15
PostgreSQL	8.4.2 e 9.1.3

Archivi Oracle[5]

Deve essere disabilitata, utilizzando la procedura di configurazione, la gestione degli statement preparati, e, per la versione 10g, deve anche impostata la gestione della clausola "Order By" sui cursori.

Driver Oracle: non si riscontrano particolari difficoltà di utilizzo.

Archivi SQL Server[6]

Il DBMS **deve** essere configurato in modalità **case sensitive**[7]. Nel caso in cui ciò non fosse possibile è necessario impostare, utilizzando la procedura di configurazione, l'opzione 'Accetta solo maiuscole' (**N.B.**:

[3] A partire dalla versione 3.03 di SIGLA (ottobre 2004).
[4] A partire dalla versione 3.17.3 di SIGLA (gennaio 2009).
[5] E' possibile utilizzare anche Oracle 10g Express Edition.
[6] E' possibile utilizzare anche la versione Express.

questa operazione può essere eseguita senza controindicazioni **solo se la base dati è vuota**). Deve essere impostata, utilizzando la procedura di configurazione,la gestione della clausola "Order By" sui cursori. La gestione degli statements preparati deve essere disabilitata.

Driver Microsoft: non si riscontrano particolari difficoltà di utilizzo. Poiché il driver prodotto da Microsoft non permette la gestione di più statements attivi sulla stessa connessione, SIGLA alloca quattro diverse connessioni per ogni database usato (in modo trasparente per l'utilizzatore**). Accertarsi quindi che il numero di connessioni gestibili dal Server sia sufficiente**.

Archivi DB2/400[8]

Gli ordinamenti alfabetici che si ottengono da SIGLA devono seguire i criteri della codifica EBCDIC, è pertanto necessario configurare l'uso dell'ordinamento EBCDIC. Deve essere impostata la gestione della clausola "Order By" sui cursori e deve essere disabilitata la gestione degli statement preparati. **I database devono essere creati su AS/400 attraverso lo statement SQL "CREATE COLLECTION".**

Ricordiamo che l'AS400 può essere utilizzato come database server di SIGLA solo con la release 4 del sistema operativo e Client Access versione 5.x e fino alla versione 3.26 di SIGLA.

Driver IBM Client/Access 400: non si riscontrano particolari difficoltà di utilizzo.

Archivi DB2/2[9] - DB2/6000 -DB2/NT - DB2/Windows 9x

Deve essere impostata, utilizzando la procedura di configurazione, la gestione della clausola "Order By" sui cursori e deve essere disabilitata la gestione degli statement preparati.

Driver IBM: non si riscontrano particolari difficoltà di utilizzo.

Archivi MySQL[10]

Deve necessariamente essere attivato il supporto per le tabelle di tipo InnoDB necessario per la gestione delle transazioni (gli statement di CREATE TABLE utilizzano la clausola aggiuntiva ENGINE=InnoDB[11]). Il valore di default per il livello di isolamento delle transazioni (`TRANSACTION ISOLATION LEVEL`) deve essere impostato a `READ COMMITTED` (il valore impostato dalla procedura di installazione di MySQL è invece `REPEATABLE READ`). A partire dalla versione 5.1.20 è necessario modificare il valore della varibile `sql_mode` (SQL Mode) aggiungendo l'opzione `PAD_CHAR_TO_FULL_LENGTH`.
Inoltre, dal momento che MySQL non opera in modo case sensitive, deve essere impostata, utilizzando la procedura di configurazione, l'opzione 'Accetta solo maiuscole' (**N.B.**: questa operazione può essere eseguita senza controindicazioni **solo se la base dati è vuota**). E' inoltre necessario attivare, sempre dal programma di configurazione, la gestione della clausola "Order By" sui cursori e disabilitare la gestione degli statements preparati.

Driver MySQL[12]: non si riscontrano particolari difficoltà di utilizzo. Il driver tratta il carattere '\' come inizio di una sequenza di escape e pertanto lo filtra. Peraltro è possibile che si ottengano dei comportamenti imprevisti in funzione del carattere che segue tale barra. La soluzione adottata da SIGLA è quella di impedire l'immissione del carattere '\' che viene automaticamente sostituito dal carattere '/', quindi i percorsi saranno inseriti utilizzando la barra UNIX e non quella DOS (esempio: il percorso C:\SIGLAPP\FILES viene inserito come C:/SIGLAPP/FILES, ma gestito correttamente al momento del suo utilizzo). La versione utilizzata nei test è la 3.51.12.00 .

Archivi PostgreSQL

Versioni precedenti la 8.2.0 **non** possono essere utilizzate poiché solo da tale versione sono state introdotte delle estensioni al linguaggio SQL necessarie (clausola IF EXISTS del comando DROP). Devono essere impostate, utilizzando la procedura di configurazione, le opzioni per la gestione della clausola "Order By" sui cursori. La gestione degli statement preparati DEVE ESSERE DISABILITATA.

---

[7] In realtà è necessario che la collating sequence dei database utilizzati da SIGLA distingua tra i caratteri maiscoli e minuscoli.

[8] A partire dalla versione 3.27, rilasciata il 6 settembre 2011, cessa il supporto per DB2/400. La versione 4 di SIGLA non è certificata per l'utilizzo di DB2/400.

[9] E' possibile utilizzare anche la versione Express-C.

[10] I test di compatibilità sono stati eseguiti a partire dalla versione 4.0.20, pertanto si sconsiglia l'utilizzo di versioni precedenti in quanto non utilizzate per le verifiche di laboratorio.

[11] Per le versioni precedenti la 5.1 la clausola utilizzata è, invece, TYPE=InnoDB.

[12] I test di compatibilità sono stati eseguiti a partire dalla versione 3.51.08.

Driver PostgreSQL ANSI: non si riscontrano particolari difficoltà di utilizzo. Il driver tratta il carattere '\' come inizio di una sequenza di escape e pertanto lo filtra. Peraltro è possibile che si ottengano dei comportamenti imprevisti in funzione del carattere che segue tale barra. E' pertanto **necessario** agire sul parametro di configurazione del DBMS "standard_conforming_strings" impostandolo al valore "on". Per i dettagli dell'operazione si rimanda alla specifica documentazione di PostgreSQL.

DBMS non validati durante lo sviluppo possono essere utilizzati salvo verifica di funzionamento. Il laboratorio software è a disposizione dei rivenditori che trovassero difficoltà nell'utilizzo di DBMS non ufficialmente certificati per fornire tutta l'assistenza possibile. Tuttavia, poiché non è ovviamente pensabile poter fornire assistenza su tutti i DBMS disponibili in commercio, il supporto rientrerà nei limiti delle competenze tecniche degli sviluppatori del laboratorio.

Attivazione della Procedura

Dopo aver installato la procedura dai minidischi di setup è necessario attivarla nel seguente modo:

1. creare un database per i dati comuni e registrarlo fra le fonti dati ODBC assegnandogli il nome SIGLAPP
2. lanciare la procedura di configurazione, ignorare il messaggio di errore che viene generato e selezionare l'opzione 'Creazione Database Dati Comuni' sul menù 'Servizi'. Selezionare in questa fase le tabelle precaricate nelle quali si desidera inserire i dati
3. al termine della fase 2 uscire dalla procedura di configurazione
4. per ognuna delle aziende da gestire:
   a. creare il database destinato a contenere i dati dell'azienda e registrarlo fra le fonti di dati ODBC
   b. lanciare la procedura di configurazione, registrare la ditta utilizzando il menu 'Ditte'
   c. aprire la ditta attraverso l' opzione 'Apri Ditta' del menù 'File', ignorare il messaggio d'errore che viene generato e selezionare l'opzione 'Creazione Database Ditta' sul menù 'Servizi'

Suggerimenti

La creazione di un database deve avvenire tramite l'uso delle utility fornite dal DBMS utilizzato. Nel caso di adozione di file xBase un database è semplicemente una directory su disco.

Usando DB2/400 un database è una libreria e può essere generato tramite lo statement SQL: CREATE COLLECTION.

Usando archivi Oracle è opportuno operare nel seguente modo:

* creare un tablespace destinato a contenere i dati comuni con lo statement
  CREATE TABLESPACE <nometable> DATAFILE <nomefile> SIZE <dimensione>
  (si consiglia di usare una dimensione di almeno 50MB)
* creare un utente abilitato a lavorare sul tablespace con lo statement
  CREATE USER <nome> IDENTIFIED BY <password> DEFAULT TABLESPACE <nometable>
* assegnare all'utente i diritti necessari con lo statement **GRANT CONNECT,RESOURCE TO <nome>**

<nomeuser> e <password> saranno richiesti da SIGLA e dal programma di configurazione all'atto dell'avvio.

Per ognuna delle ditte da gestire:

* creare un tablespace destinato a contenere i dati
* creare un utente di default abilitato a lavorare sul tablespace
* assegnare all'utente i diritti (CONNECT e RESOURCE)
* eseguire le operazioni del punto 4 di 'ATTIVAZIONE DELLA PROCEDURA'

Encriptazione della stringa di connessione al database "dati comuni"

All'avvio di SIGLA o del programma di configurazione viene eseguita una connessione al database dati comuni che, come abbiamo visto nel paragrafo precedente, deve essere registrato fra le fonti di dati ODBC con il nome SIGLAPP.

All'atto della connessione la procedura non è in grado di fornire automaticamente all'amministratore ODBC né l'ID dell'utente con cui effettuare il collegamento al DBMS né l'eventuale password. Come consequenza di questo fatto è necessario divulgare agli utilizzatori l'ID e la password dell'utente da usare per effettuare la connessione.

A partire dalla release 1.61 SIGLA è in grado di superare questo inconveniente con un meccanismo di encriptazione della stringa di connessione al database dati comuni. Per rendere attiva questa funzionalità è necessario operare come segue:

1. in una task dos lanciare il programma di utilità **SPPPWD.EXE** e digitare, quando richiesto, la stringa di connessione da usare per il database dati comuni (a.e. DSN=SIGLAPP;UID=HERBERT;PWD=CROSS). Terminare la digitazione della stringa di connessione premendo <Invio>
2. Copiare il file SPPPWD.PWD generato dalla procedura SPPPWD.EXE sui vari PC nella directory di installazione di SIGLA

Operando nel modo descritto l'applicazione fornisce direttamente al manager ODBC la stringa criptata nel file SPPPWD.PWD e, se il DBMS lo prevede, non richiede più alcun parametro di connessione all'utilizzatore.

Un importante conseguenza di quanto esposto è che decade, utilizzando questo meccanismo, l'obbligo di registrare il database dati comuni con il nome fisso SIGLAPP fra le fonti di dati. Infatti, se la stringa di connessione fornita è, ad esempio, "DSN=PROVA;UID=HERBERT;PWD=CROSS", la connessione alla base dei dati comuni avviene sulla fonte di dati PROVA.

Attenzione: il programma SPPPWD.EXE crea il file SPPPWD.PWD nella stessa cartella da cui viene eseguito, quindi è necessario accertarsi di disporre dei diritti necessari a scrivere in tale cartella.

Versione Start Suite (START v3)

La versione Start Suite del pacchetto del pacchetto SIGLA 3 è disponibile per il solo ambiente operativo Windows 32. L'accesso ai files dati (di tipo FoxPro compatibile) è stato implementato tramite l'uso della libreria CodeBase 6.0 normalmente reperibile in commercio. L'adozione di tale libreria nella realizzazione di eventuali customizzazioni permette di mantenere un assoluto livello di compatibilità con il formato degli indici gestiti da SIGLA.

A differenza della versione client/server nella quale il locking delle risorse è delegato al DBMS attraverso l'implementazione dell'attività transazionale, la versione Start Suite realizza il lock logico a livello di record in modo completamente compatibile con lo standard FoxPro.

La struttura delle tabelle utilizzate e le funzionalità disponibili sono perfettamente uguali a quelle della versione client/server.

Poiché l'architettura dell'applicazione richiede l'apertura contemporanea di molti files (tabelle e indici) è necessario, qualora si utilizzi Windows95/98/Me, accertarsi che il file CONFIG.SYS del personal sul quale si intende testare la procedura contenga la specifica **FILES=200.**

Se la base di dati è montata su un file server è necessario verificare che il client sia in grado di aprire almeno 200 files.

Il prodotto è rilasciato per Windows 95 e può essere installato anche su Windows 98/Me e workstation Windows NT 4.0, Windows 2000 o Windows XP[13].

N.B.: per la versione Start Suite le funzioni di ricostruzione indici sia per i dati comuni che per i database aziendali eseguono anche un Pack delle informazioni (i record logicamente cancellati vengono eliminati fisicamente).

Installazione e avviamento

Riteniamo sia cosa utile riportare brevemente alcuni suggerimenti sull'installazione del prodotto relativamente alle configurazioni tipiche.

Installazione Stand Alone (singolo posto di lavoro Win9x/Me/NT 4.0/2000/Xp)

Installare SIGLA dal CD di distribuzione seguendo le note riportate nel file *Installazione.pdf*. Successivamente:

* creare una cartella nel disco fisso destinato a contenere tabelle e indici del database dati comuni (a.e **MD C:\SPPGENER**)
* per ognuna delle ditte da gestire creare una cartella nel disco fisso destinato a contenere tabelle e indici di ciascuna azienda (a.e. **MD C:\SPPDIT01, MD C:\SPPDIT02 etc.**)
* (*WIN9x/Me*) editare il file **AUTOEXEC.BAT** definendo la variabile di environment **SPPROOT** e facendola puntare al percorso (path completo) della cartella contenente il database dati comuni (ad es. **aggiungere la linea SET SPPROOT=C:\SPPGENER**)
* (*WINNT*) inserire tra le variabili di sistema la variabile di ambiente **SPPROOT** attribuendole come valore il percorso (path completo) della cartella contenente il database dati comuni (dal menù *Avvio-Impostazioni-Pannello di controllo*, doppio click sull'icona *Sistema*, folder *Ambiente*, doppio click su una qualsiasi delle variabili di sistema, modificare il nome della variabile in **SPPROOT** e il valore in, ad esempio, **C:\SPPGENER**)
* (*WIN2000/Xp*) inserire tra le variabili di sistema la variabile di ambiente **SPPROOT** attribuendole come valore il percorso (path completo) della cartella contenente il database dati comuni (dal menù *Start-Impostazioni-Pannello di controllo*, doppio click sull'icona *Sistema*, folder *Avanzate*, premere *Variabili d'ambiente* e successivamente premere il bottone *Nuovo* nel settore delle variabili di sistema, inserire il nome della variabile **SPPROOT** e il valore, ad esempio, **C:\SPPGENER**)
* (*WIN9x/Me*) riavviare il computer

[13] Windows Vista e Windows 7 non rientrano tra le piattaforme di test previste per START v3, inoltre START v3 non è certificata per i sistemi operativi a 64 bit.

* lanciare la procedura di configurazione, ignorare il messaggio di avvertimento che viene generato e selezionare l'opzione "Creazione Database Dati Comuni" sul menù "Servizi". Selezionare in questa fase le tabelle precaricate che si desiderano generare

Per ognuna delle aziende da gestire:

* lanciare la procedura di configurazione, registrare la ditta utilizzando il menù "Ditte" specificando il percorso (path completo) della cartella che ne contiene i dati (ad es. **C:\SPPDIT01**);
* aprire la ditta attraverso l'opzione "Apri Ditta" del menù "File", ignorare il messaggio di avvertimento che viene generato e selezionare l'opzione "Creazione Database Ditta" sul menù "Servizi".

Installazione multiutente su rete peer to peer (Win9x/Me/NT 4.0/2000/Xp)

Installare START dal CD di distribuzione seguendo le note riportate nel file *Installazione.pdf* su tutti i PC. Decidere quale dei PC debba ospitare il database dati comuni e della/e ditta/e.

Successivamente su tale PC:

* creare una cartella nel disco fisso destinato a contenere tabelle e indici del database dati comuni (a.e **MD C:\ARCHIVI\SPPGENER**)
* assicurarsi che la cartella creata sia condivisa con diritti di **LETTURA e SCRITTURA** (nel caso dell'esempio specificato occorre condividere la cartella C:\ARCHIVI)
* (*WIN9x*) editare il file **AUTOEXEC.BAT** mappando come unità logica la cartella creata (ad es. **aggiungere il comando SUBST S: C:\ARCHIVI**)
* (*WINMe*) creare un file .bat con il comando **SUBST S: C:\ARCHIVI** e dopo su *Esecuzione automatica*, inserire un nuovo collegamento a tale file
* (*WINNT*) inserire in *Esecuzione automatica* il comando **SUBST S: C:\ARCHIVI** (fare click col tasto destro del mouse sulla tasto *Avvio* e selezionare *Apri All Users*, doppio click su *Programmi* e dopo su *Esecuzione automatica*, inserire un nuovo collegamento e nella riga di comando **%windir%\system32\subst.exe S: C:\ARCHIVI**)
* (*WIN2000/Xp*) inserire in *Esecuzione automatica* il comando **SUBST S: C:\ARCHIVI** (fare click col tasto destro del mouse sulla tasto *Start* e selezionare *Apri Cartella Utenti*, doppio click su *Programmi* e dopo su *Esecuzione automatica*, inserire un nuovo collegamento e nella riga di comando **%windir%\system32\subst.exe S: C:\ARCHIVI**)
* (*WIN9x/Me*) editare il file **AUTOEXEC.BAT** definendo la variabile di environment **SPPROOT** e facendola puntare al percorso (path completo) della cartella contenente il database dati comuni (ad es. **aggiungere la linea SET SPPROOT=S: \SPPGENER**)
* (*WINNT*) inserire tra le variabili di sistema la variabile di ambiente **SPPROOT** attribuendole come valore il percorso (path completo) della cartella contenente il database dati comuni (dal menù *Avvio-Impostazioni-Pannello di controllo*, doppio click sull'icona *Sistema*, folder *Ambiente*, doppio click su una qualsiasi delle variabili di sistema, modificare il nome della variabile in **SPPROOT** e il valore in, ad esempio, **S:\SPPGENER**)
* (*WIN2000/Xp*) inserire tra le variabili di sistema la variabile di ambiente **SPPROOT** attribuendole come valore il percorso (path completo) della cartella contenente il database dati comuni (dal menù *Start-Impostazioni-Pannello di controllo*, doppio click sull'icona *Sistema*, folder *Avanzate*, premere *Variabili d'ambiente* e successivamente premere il bottone *Nuovo* nel settore delle variabili di sistema, inserire il nome della variabile **SPPROOT** e il valore, ad esempio, **S:\SPPGENER**)
* riavviare il computer
* per ognuna delle ditte da gestire creare un indirizzario su disco fisso destinato a contenere tabelle e indici di ciascuna azienda (a.e. **MD S:\SPPDIT01, MD S: \SPPDIT02 etc.**)
* seguire la procedura già illustrata per la creazione dei database dati comuni e dati ditta

Sugli altri PC:

* mappare come unità logica la cartella condivisa con gli archivi (**N.B.:** utilizzare la stessa lettera di unità utilizzata sul precedente, ad es. **S:**), utilizzare *Risorse di Rete* per connettere l'unità di rete

* (*WIN9x/Me*) editare il file **AUTOEXEC.BAT** definendo la variabile di ambiente **SPPROOT** e facendola puntare al percorso (path completo) della cartella contenente il database dati comuni (ad es. **aggiungere la linea SET SPPROOT=S: \SPPGENER**)
* (*WINNT*) inserire tra le variabili di sistema la variabile di ambiente **SPPROOT** attribuendole come valore il percorso (path completo) della cartella contenente il database dati comuni (dal menù *Avvio-Impostazioni-Pannello di controllo*, doppio click sull'icona *Sistema*, folder *Ambiente*, doppio click su una qualsiasi delle variabili di sistema, modificare il nome della variabile in **SPPROOT** e il valore in, ad esempio, **S:\SPPGENER**)
* (*WIN2000/Xp*) inserire tra le variabili di sistema la variabile di ambiente **SPPROOT** attribuendole come valore il percorso (path completo) della cartella contenente il database dati comuni (dal menù *Start-Impostazioni-Pannello di controllo*, doppio click sull'icona *Sistema*, folder *Avanzate*, premere *Variabili d'ambiente* e successivamente premere il bottone *Nuovo* nel settore delle variabili di sistema, inserire il nome della variabile **SPPROOT** e il valore, ad esempio, **S:\SPPGENER**)

START versione CLIENT/SERVER

Introduzione

La versione *tradizionale* di START opera su archivi e indici Visual FoxPro compatibili in modalità multi user. Ciò significa che, lavorando in rete con più posti di lavoro, ogni workstation provvede autonomamente alla manutenzione della base di dati operando tutte le attività connesse con l'aggiornamento di record nelle varie tabelle (gestione dei locking, aggiornamento degli indici etc.). In altre parole l'applicazione in esecuzione provvede ad eseguire fisicamente tutti gli accessi necessari alla base di dati leggendo le informazioni occorrenti dal disco del file server.

Questo tipo di operatività presenta ovviamente degli aspetti critici.

- Le performance operative sono fortemente condizionate dalle prestazioni a dall'affidabilità della rete locale.
- Il traffico di rete è estremamente elevato. Poiché infatti i singoli posti di lavoro devono provvedere autonomamente alla manutenzione delle tabelle e degli indici gestiti, attività come la ricerca di un record in una tabella comportano la lettura di pagine dati dagli indici. E' evidente quindi che sulla rete locale, oltre all'informazione *utile*, passano anche una grossa quantità di dati *di servizio*.
- La presenza nell'anello di rete di una macchina *lenta* condiziona le performance complessive dell'intera installazione. Infatti, poiché durante l'esecuzione di certe attività (ad esempio l'aggiunta di un record a una tabella) è necessario mantenere dei lock sui file coinvolti, finché l'operazione non giunge a conclusione tutte le altre work station sono in condizione di *wait*.
- Il malfunzionamento o il crash di un posto di lavoro comporta, quasi certamente, il danneggiamento della base di dati gestita con conseguente blocco dell'operatività di tutta l'installazione.

Un approccio alternativo al problema è quello di realizzare, invece di una tradizionale architettura *multi user*, una architettura *client/server*. In un'ottica di questo tipo esiste un unico programma, il database server, che provvede alla manutenzione della base di dati. Le applicazioni client si limitano a richiedere al server l'esecuzione delle operazioni necessarie.

L'approccio client/server presenta alcuni vantaggi interessanti.

- Soltanto il database server opera fisicamente gli aggiornamenti alla base di dati. Poiché i client e il server colloquiano fra loro attraverso la rete, il traffico di rete si limita alle sole informazioni *utili*.
- Il crash di un client non comporta il danneggiamento fisico della base di dati perché il client non è in alcun modo coinvolto nelle attività di manutenzione dei dati stessi.
- Le performance complessive dell'installazione sono condizionate essenzialmente dalla performance del server: disponendo di un server *potente* è possibile anche utilizzare client che lo sono molto meno.
- La diminuzione del traffico di rete migliora le prestazioni delle architetture di tipo peer-to-peer.

Implementazione

La versione client/server di START utilizza un motore per l'accesso alla base di dati **che DeltaPhi SIGLA srl può distribuire gratuitamente assieme al prodotto.** Il motore funziona su archivi e indici Visual FoxPro compatibili (analogamente a quanto avviene per la versione multiuser).
Il programma servente (S4SERVER.EXE) è una applicazione Win32 e può quindi essere eseguito su macchine Windows 95/98/Me o NT/2000/Xp. Il protocollo di comunicazione utilizzato per permettere il colloquio fra client e server è **TCP/IP-Windows Sockets (la porta utilizzata è la 23165)**.
La macchina su cui deve essere eseguita l'applicazione server richiede una architettura Intel e deve avere buone caratteristiche prestazionali (processore veloce, disco ad accesso rapido, sufficiente quantità di RAM). Per ottimizzare le prestazioni è necessario che il programma server operi **direttamente** sul disco fisso della macchina ospite. E' altamente sconsigliato far lavorare l'applicazione server sul disco condiviso gestito da un'altra macchina. In tal modo infatti verrebbero meno gran parte dei benefici forniti dall'architettura client/server. E' inoltre consigliabile che il PC che funge da server sia **dedicato** a questo compito. Eseguendo su tale macchina qualunque tipo di applicazione si distolgono risorse dall'applicazione server penalizzando le prestazioni complessive dell'intera installazione.
I client devono essere macchine ad architettura Intel dotate del sistema operativo Windows 95/98/Me o NT/2000/Xp.

Installazione

Installazione e configurazione del server

Sulla macchina destinata a svolgere le funzioni di database server devono essere installate, in una cartella creata ad hoc, le componenti software necessarie che comportano un occupazione di spazio su disco di circa 1,6MB.
E' necessario configurare il protocollo TCP/IP[14] assegnando al PC un IP address univoco.
Poiché l'architettura di START richiede comunque la visibilità del disco del server è necessario che il disco stesso sia configurato come condivisibile in lettura/scrittura e che il server stesso sia in grado di vederlo con la stessa mappatura che sarà utilizzata sui client. Questo risultato può essere ottenuto tramite l'uso del comando **SUBST** .

Installazione e configurazione dei client

Sui client deve essere installata la versione client/server di START (SIGLA32.EXE).
E' necessario configurare il protocollo TCP/IP assegnando al PC un IP address univoco.
E' necessario mappare l'indirizzo IP del server nel file HOSTS del client aggiungendo una linea del tipo:
xxx.xxx.xxx.xxx siglapp
dove xxx.xxx.xxx.xxx è l'IP address del server e siglapp è il nome assegnato al server (il nome **deve** essere obbligatoriamente siglapp).

N.B.: quando si installa per la prima volta il protocollo TCPIP il file HOSTS non esiste, lo si può pertanto creare ex novo (fondamentale che non abbia alcuna estensione); è disponibile il file di esempio HOSTS.SAM che non viene utilizzato per risolvere i nomi simbolici in indirizzi numerici e che pertanto non deve essere modificato.

E' necessario mappare il disco del server destinato a contenere i dati assegnandogli la stessa unità logica che sarà usata sugli altri client e che deve essere anche impostata sul server stesso con il comando SUBST.

Esempio

Sul server

- Installare le componenti software necessarie
- Assegnare al server l'indirizzo IP 192.168.0.1
- Mappare il disco C:, destinato a contenere i dati, come J: col comando **SUBST J: C:** (è conveniente inserire il comando in un file batch ed inserirlo nel menù di Avvio)
- Lanciare il database server S4SERVER.EXE (è conveniente inserire il comando nel menù di Avvio)

[14] Nel caso in cui sia presente un firewall è necessario che sia configurato in modo da consentire la comunicazione sulla porta 23165.

Sul client

- Installare la versione client/server di START
- Assegnare al client un IP address univoco
- Editare il file HOSTS aggiungendo la riga: **192.168.0.1 siglapp**
- Testare il funzionamento della rete eseguendo un ping sul server con il comando: **ping siglapp**
- Mappare il disco del server come J:
- Impostare la variabile di ambiente SPPROOT (a.e set SPPROOT=J:\SIGLAPP)

N.B.: Il programma di configurazione CONFIG32.EXE non può essere eseguito in modalità client/server. Per attivarlo è necessario che il database server non sia in funzione.

Note operative

Così come viene distribuito il database server è configurato per aprire i file che devono essere gestiti in **modo esclusivo.** In questo modo nessun'altra applicazione è in condizione di operare sui file in questione (ne consegue che sul server non è possibile usare START nella versione multi user).
E' possibile variare la configurazione del server nel seguente modo per far sì che i file siano aperti in modalità shared:

- Disattivare il database server (S4SERVER.EXE)
- Posizionarsi nella directory dove è stato installato il database server e lanciare la stringa di comando: **S4EDIT S4SERVER OPENMODE 0**
- lanciare la stringa di comando: **S4EDIT S4SERVER OPTIMIZE 0**
- lanciare la stringa di comando: **S4EDIT S4SERVER OPTIMIZEWR 0**

E' possibile riattivare la configurazione di default (file aperti in modo esclusivo) con il comando:

S4EDIT S4SERVER OPENMODE 1
S4EDIT S4SERVER OPTIMIZE 1
S4EDIT S4SERVER OPTIMIZEWR 1

Se i file sono aperti dal server in modalità shared è possibile installare sul PC che funge da server la versione multiuser di START utilizzando quindi anche tale PC come posto di lavoro. **Ricordiamo comunque che questa operazione è altamente sconsigliata perché penalizza le performance complessive dell'installazione.**
In ogni caso è possibile installare sul PC che funge da server la versione client/server di START che non richiede l'apertura in modalità shared dei files. Questa configurazione è senza dubbio da preferirsi a quella che prevede l'installazione sul server della versione multiuser anche se le performance complessive dell'installazione vengono comunque penalizzate.
Il database server (S4SERVER.EXE) quando è in esecuzione si presenta a video come una normale finestra Windows. La voce di menù <Help/About> permette di vedere quanti client stanno usando il server al momento. **Il database server deve essere terminato solo quando nessun client lo sta utilizzando.**

Gestione del LOG

Il database server costruisce automaticamente due file:

- S4SERVER.ERR che contiene una traccia degli eventuali errori manifestatisi durante l'attività
- S4SERVER.LOG che contiene il log delle attività effettuate sulle varie tabelle

I due file (in particolare S4SERVER.LOG) crescono nel tempo e devono essere periodicamente cancellati.

N.B: l'aggiornamento del file di log viene eseguito SOLO da START client/server. Eventuali procedure esterne o la versione multiuser di START che lavorassero sugli stessi archivi non sono in grado di manutenere il file di log.

E' comunque del tutto evidente che la versione client/server di START non può in alcun modo essere paragonata a SIGLA. Infatti SIGLA, demandando la gestione dei dati ad un vero e proprio DBMS, è in grado di offrire garanzie irraggiungibili da START relativamente alla salvaguardia dell'integrità dei dati e al livello di performance raggiungibili.

Codici d'errore

Di seguito sono riportati i codici d'errore che possono essere generati a run time dalla libreria utilizzata da SIGLA Start suite per l'accesso alla base di dati. I codici sono elencati nella documentazione tecnica di "CodeBase 6.0".

Codice	Significato
-10	**errore durante la chiusura di un file**
-20	**errore durante la creazione di un file**
-30	**errore nella determinazione della lunghezza di un file**
-40	**errore durante la memorizzazione della lunghezza di un file**
-50	**errore durante il locking di un file**
-60	**errore durante l'apertura di un file**
-61	**negazione del permesso di accesso au un file**
-62	**modo di apertura di un file errato**
-63	**troppi files aperti**
-64	**file non trovato**
-69	**tentativo di aprire più volte lo stesso file**
-70	**errore di lettura su un file**
-80	**errore durante la cancellazione di un file**
-90	**errore durante il renaming di un file**
-110	**errore durante l'unlock di un file**
-120	**errore di scrittura su disco**
-200	**il file che si cerca di aprire non è un file DBF corretto. Possono ad esempio esserci dei problemi nella struttura dell'header (il file può essere logicamente o fisicamente danneggiato)**
-210	**nome campo non riconosciuto**
-220	**tipo campo non valido**
-230	**lunghezza record maggiore del valore massimo ammesso**
-240	**tentativo di aggiungere un record in posizione non consentita**
-250	**errore di posizionamento sul file**
-300	**tag non trovato**
-310	**indice non corretto (il file può essere logicamente o fisicamente danneggiato)**
-330	**tag non selezionato**
-340	**tentativo di aggiungere una chiave duplicata**
-350	**tentativo di creare un tag non valido**
da -400 a -510	**errore di sintassi in una espressione xBase**
da -610 a -630	**errore nell'ottimizzazione dell'accesso ai dati**
da -710 a -730	**errore nell'impostazione delle relazioni fra files**
-910	**dato non atteso**
-920	**memoria insufficiente**
-930	**parametro errato**
-935	**parametro nullo**
-950	**risultato non atteso**

-960	errore nella verifica di una struttura dati
-970	struttura dati corrotta o non inizializzata
-1090	funzione non supportata
-1200	operazione non consentita durante una transazione
-1210	transazione fallita
-1220	errore durante il rollback di una transazione
-1230	errore durante la Commit di una transazione
-1240	errore durante la gestione del file di log
-1300	errore di comunicazione
-1310	errore di connessione
-1320	errore generato dai sockets di windows
-1330	errore di rete generico
-1340	impossibile caricare la DLL di comunicazione
-1350	network timeout
-1360	pacchetto di comunicazione corrotto
-1430	errore nellavalidazione dei diritti di accesso
-2100	errore del server

Struttura del database

SIGLA opera su un diverso database per ognuna delle aziende attivate. Una ulteriore base di dati viene utilizzata per la gestione delle informazioni comuni a tutte le aziende (aliquote iva, sportelli bancari etc.). Questa particolare base di dati DEVE essere registrata nell'amministratore ODBC con il nome "SIGLAPP".

La documentazione del tracciato delle varie tabelle è riportata nel file di help per MS Windows **TECNOTES.CHM**.

Importazione di massa di dati esterni

La procedura di configurazione di SIGLA permette l'importazione di massa da file ASCII di movimenti contabili e iva,movimenti di magazzino e documenti di magazzino. Le voci di menù necessarie ad adempiere a questo compito sono elencate nell'opzione 'Servizi'.

Di seguito viene descritto il tracciato dei file ASCII necessari ad importare i dati elencati nel database di SIGLA. **I files descritti dovranno essere presenti all'interno dell'indirizzario specificato come 'Subdirectory Files' nella finestra 'Generali' (pagina 'Gener.#1') della procedura di configurazione.**

Importazione di massa movimenti contabili e iva

La funzione (accessibile da programma CONFIG32.EXE, icona configurazione) permette l'immissione automatica dei movimenti di contabilità generale e iva, nonchè l'aggiornamento dei progressivi sottoconto, da file di testo esterni al pacchetto. Devono essere preparati due file ASCII sequenziali (ogni record è terminato con i caratteri ascii 0D e 0A), uno per i movimenti contabili **XMOVCO.TXT**, ed uno per i movimenti iva **XMOVIV.TXT**.

Il file XMOVCO.TXT deve essere presente **sempre** mentre il file XMOVIV.TXT deve essere preparato solo se sono presenti movimenti iva.

I due file sono identici nel tracciato alle tabelle di SIGLA **MOVCO** e **MOVIVA**. E' obbligatorio identificare i diversi raggruppamenti contabili in XMOVCO.TXT con uno stesso numero (campo NUMERO di movco). E' obbligatorio attribuire lo stesso numero ai movimenti iva correlati in XMOVIV.TXT.

La procedura di importazione effettua, prima di procedere con l'accodamento dei dati, dei controlli di congruenza sulle informazioni contenute nei due files. In particolare sul file XMOVCO.TXT viene controllato che:

* il campo 'NUMERO' contenga un valore significativo (numero di sette digit zero-filled)
* il campo 'SEGNO' contenga il valore 'A' o 'D'
* il campo 'SOTTOCONTO' contenga il codice di un sottoconto esistente
* il campo 'CAUSALE' contenga il codice di una causale contabile esistente
* ogni raggruppamento di registrazioni (stesso 'NUMERO') soddisfi la quadrature dare/avere a meno dei movimenti sospesi
* se un movimento contabile ha il flag 'MOV_IVA' impostato al valore 'S' esista in XMOVIV.TXT almeno un record con lo stesso valore per il campo 'NUMERO'
* il campo 'REGIVA' contenga un codice di registro iva esistente (il controllo viene eseguito solo se la causale contabile utilizzata ha il campo 'DOCUMENTIVA' impostato al valore 'S')
* il campo 'CODIVA' contenga il valore NULL o il codice di una aliquota iva esistente

N.B. La prima regsitrazione contabile inserita in XMOVCO.TXT deve avere il campo numero=0000001

Ordine di inserimento nel database dei movimenti di tipo IVA: si assume che i movimenti abbiano lo stesso valore per il campo numero. L'ordine rimane stabilito dal valore del campo riga(1,2 etc.)

I dati relativi alla registrazione di fatture/corrispettivi devono essere inseriti nei files ASCII nel seguente ordine:

RIGA=00001: movimento totale documento (il campo 'SOTTOCONTO' deve essere del tipo 'cliente/fornitore', se il registro iva indicato nella causale contabile NON è di tipo corrispettivi. In caso contrario - il registro iva è di tipo corrispettivi- il campo 'SOTTOCONTO' può essere di qualsiasi tipo.

RIGA=00002: Movimento di contropartita (il campo 'SOTTOCONTO' puòessere solo del tipo 'altri' e non uguale SOTTOCONTO IVA)

RIGA=00003 e successivi : Movimento di contropartita oppure movimento per il SOTTOCONTO IVA (il campo 'SOTTOCONTO' puòessere solo del tipo 'altri').

Quando si incontra un movimento con SOTTOCONTO=SOTTOCONTO IVA si considera terminato il documento IVA e si controlla la quadratura.

Eventuali effetti/scadenze dovranno avere il campo 'NUMERO' di valore diverso e quindi (se non sono sospesi) costituire un articolo di contabilita generale. Gli effetti possono comunque essere collegati al documento iva tramite il campo 'RIFENUMERO' (uguale per tutti i record interessati).

I movimenti con il campo 'SOSPESO_SN'='S' non sono conteggiati nella quadratura dare-avere.

N.B.: 'RIFENUMERO' non ha nessuna relazione con il concetto di PARTITA (V.Gestione a partite aperte) per il quale il campo da riempire è 'NUMRIFSALD'.

Sul file XMOVIV.TXT viene controllato che:

* il campo 'NUMERO' contenga un vaolre significativo (numero di sette digit zerofilled)
* il campo 'SOTTOCONTO' contenga il codice di un sottoconto esistente
* il campo 'CODIVA' contenga il valore NULL o il codice di una aliquota iva esistente
* il campo 'REGISTRO' contenga un codice di registro iva esistente
* esista la quadratura della parte iva di uno stesso raggruppamento di registrazioni (stesso 'NUMERO') attraverso la condizione IVA **totfattura-sum(imponibili)-sum(imposta)==0**

Ha partire dalla versione 2.16 sono stati introdotti ulteriori controlli di obbligatorietà:

* il campo 'ESECOMPET' deve contenere un codice esercizio esistente.
* il campo 'DATACOMPET' deve contenere una data compresa fra le date d'inizio e fine esercizio per l'esercizio indicato in ESECOMPET.
* il campo 'DATAREG' deve contenere una data minore della data ultimo consolidamento per l'esercizio indicato in ESECOMPET.

NB: I pagamenti delle fatture in sospensione d'imposta e le fatture Intrastat con causale contabile con doppio registro iva non vengono né esportate né importate dalle procedure automatiche di SIGLA (esportazione/importazione di massa movimenti contabili).

Struttura del file XMOVCO.TXT

Il File XMOVCO.TXT è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (Esadecimali 0D,0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione. Le date sono memorizzate come stringhe di caratteri in formato ANSI (AAAAMMGG).

Nome Campo	Tipo	Lung	Descrizione
NUMERO	Char	7	numero univoco del raggruppamento di registrazioni,fillato di '0' a sinistra
RIGA	Char	5	numero della riga all'interno del raggruppamento, fillato di '0' a sinistra
ESEREGISTR	Char	4	codice esercizio di registrazione
DATAREG	Char	8	data registrazione
ESECOMPET	Char	4	codice esercizio di competenza
DATACOMPET	Char	8	data competenza
DATANEGOZ	Char	8	**non gestito: NON RIEMPIRE**
DATAVALUTA	Char	8	**non gestito: NON RIEMPIRE**
DATAMATURA	Char	8	**non gestito: NON RIEMPIRE**
DATASCAD	Char	8	data scadenza (per movimenti di scadenzario)
DATADECORR	Char	8	**non gestito: NON RIEMPIRE**
DATAPRESEN	Char	8	data presentazione (per effetti)
DATAPROTOC	Char	8	data protocollo
DATASALDAC	Char	8	**non gestito: NON RIEMPIRE**
DATACAMBIO	Char	8	data cambio (per movimenti in valuta estera)

DATAINIZIO	Char	8	**non gestito: NON RIEMPIRE**
DATAFINE	Char	8	**non gestito: NON RIEMPIRE**
DATAPRBANC	Char	8	**non gestito: NON RIEMPIRE**
DATADOCUM	Char	8	data documento
NUMDOCUM	Char	10	numero documento
NUMPROTOC	Char	7	numero protocollo
NUMRIFSALD	Char	7	numero riferimento per gestione partite aperte
NUMDISTINT	Char	7	numero distinta di presentazione (per effetti)
NUMEFFETTO	Char	7	numero effetto
NUMPRBANC	Char	7	**non gestito: NON RIEMPIRE**
SOTTOCONTO	Char	10	sottoconto movimentato
C_F	Char	1	"C" per clienti, "F" per fornitori
CONTROPART	Char	10	codice contropartita
SOTDISTINT	Char	10	sottoconto banca di presentazione (per effetti)
CAUSALE	Char	3	codice causale contabile
DESCRAGG	Char	30	descrizione aggiuntiva
DESLINGUA	Char	30	descrizione aggiuntiva in lingua (solo per le operazioni in valuta)
SEGNO	Char	1	segno: "D" per dare, "A" per avere
IMPORTO	Numeric	19	importo movimento in lire
CODVALUTA	Char	3	codice valuta estera
IMPVALUTA	Numeric	19	importo in valuta
CAMBIO	Numeric	19	rapporto di cambio con la lira
IMPRATEO	Numeric	19	**non gestito: NON RIEMPIRE**
IMPRISCONT	Numeric	19	**non gestito: NON RIEMPIRE**
NUMRATA	Numeric	19	numero rata (per effetti)
TOTRATE	Numeric	19	numero totale rate (per effetti)
IMPFATTURA	Numeric	19	importo totale fattura (per effetti)
CODBANCA	Char	11	codice ABI.CAB banca d'appoggio (per effetti)
CODBANCAPR	Char	11	codice ABI.CAB banca di presentazione (per effetti)
CCOSTO	Char	6	codice centro di costo
REGIVA	Char	2	codice registro iva (solo per movimenti iva)
TIPODOCIVA	Char	2	codice tipo documento iva (solo per movimenti iva)
CODPAGAM	Char	4	codice tipo pagamento
AGENTE	Char	5	codice agente
CAPOZONA	Char	5	codice capozona
NRIGGIORN	Char	7	numero definitivo riga di giornale contabile
SOSPESO_SN	Char	1	"S" se il movimento è sospeso, "N" altrimenti
SIMULAZ_SN	Char	1	"S" se il movimento è di simulazione, "N" altrimenti
COMPTER_SN	Char	1	"S" se il movimento è relativo a compensi a terzi, "N" altrimenti
DISTCTERZ	Char	7	**non gestito: NON RIEMPIRE**

SOLOANA_SN	Char	1	"S" se il movimento è di sola analitica,"C" se è solo contabile, "N" se è sia contabile che di analitica
TEMPORA_SN	Char	1	"S" se il movimento è temporaneo, "N" altrimenti
SCADENZ_SN	Char	1	"S" se il movimento è di scadenzario, "N" altrimenti
ATT_PASSIV	Char	1	"A" per scadenzario attivo, "P" per scadenzario passivo
TIPOEFFETT	Char	1	tipo effettivo di pagamento: 0 - rimessa diretta o contanti 1 - tratta 2 - ricevuta bancaria 3 - cessione credito 4 - paghero' 5 - lettera di credito 6 - tratta accettata 7 - RiBa 8 - ritardato pagamento 9 - cambiale A – altro pagamento B – bonifico bancario
PAGE_C_SN	Char	1	"S" per pagamento a presentazione estratto conto, "N" altrimenti
INSOLUT_SN	Char	1	"S" per registrazione insoluti, "N" altrimenti
ANTVALU_SN	Char	1	"S" per registrazione anticipazioni valutarie,"N" altrimenti
CTRLBAN_SN	Char	1	"S" per attivazione controllo banche, "N" altrimenti
TIPOCAUSAL	Char	1	tipo causale contabile: N - normale A - apertura C - chiusura
STAMPAT_SN	Char	1	"S" se il movimento è stato stampato sul giornale,"N" altrimenti
CONSECB_SN	Char	1	**non gestito: NON RIEMPIRE**
P_PAREG_SN	Char	1	**non gestito: NON RIEMPIRE**
MOV_IVA_SN	Char	1	"S" per i movimenti iva, "N" altrimenti
MOV_VAL_SN	Char	1	"S" per i movimenti in valuta,"N" altrimenti
TIPONUMERA	Char	1	codifica mezzo di pagamento: 0 - rimessa diretta 7 - RiBa B - bonifico bancario C - assegno c/c D - assegno circolare
AR74TER	Char	1	**non gestito: NON RIEMPIRE**
INTERESSI	Numeric	19	**non gestito: NON RIEMPIRE**
SPESE	Numeric	19	**non gestito: NON RIEMPIRE**
DATADECINT	Char	8	**non gestito: NON RIEMPIRE**
NUMLTSOLLE	Char	7	**non gestito: NON RIEMPIRE**
DATLTSOLLE	Char	8	**non gestito: NON RIEMPIRE**
FILELTSOLL	Char	8	**non gestito: NON RIEMPIRE**
NUMSOLLECI	Numeric	19	**non gestito: NON RIEMPIRE**
DATULTLT	Char	8	**non gestito: NON RIEMPIRE**
IMPRICEVUT	Numeric	19	**non gestito: NON RIEMPIRE**
SPPAGAT_SN	Char	1	**non gestito: NON RIEMPIRE**

INTPAGA_SN	Char	1	**non gestito: NON RIEMPIRE**
INSPAGA_SN	Char	1	**non gestito: NON RIEMPIRE**
RIEMISS_SN	char	1	**non gestito: NON RIEMPIRE**
CODICEIVA	Char	4	codice aliquota iva usata in registrazione
IMPORTOIVA	Numeric	19	importo imposta calcolata
DTCOMPCONT	Char	8	data competenza extracontabile (contabilità analitica)
FILE1	Char	8	nome dei files usati per la memorizzazione di fogli elettronici,testi di videoscrittura,immagini etc.
CASO	Char	1	**non gestito: NON RIEMPIRE**
STATO	Char	1	"S" se record attivo,"N" altrimenti
ULT_AGG	Char	8	data ultimo aggiornamento
UTENTE	Char	8	codice utente che ha operato l'ultimo aggiornamento
RIFENUMERO	Char	7	E' usato per collegare gli effetti/scadenze alla fattura
AVVOCATO	Char	10	per insoluti: sottoconto fornitore avvocato
PRATICANUM	Char	7	per insoluti: numero di riferimento pratica avvocato
PRATICADAT	Char	8	per insoluti: data pratica avvocato
INSOLUTNUM	Char	7	per insoluti: numero registrazione insoluto (per i movimenti contabili riferiti ad un insoluto)
INSOLUTRIG	Char	5	per insoluti: riga registrazione insoluto (per i movimenti contabili riferiti ad un insoluto)
AGGIORN_SN	Char	1	"S" se il movimento è stato aggiornato da SIGLA,"N" altrimenti
CONTCORR	Char	15	numero conto corrente
EIMPORTO	Numeric	19	Importo in EURO
EIMPFATTUR	Numeric	19	Importo fattura in EURO
ESPESE	Numeric	19	Importo spese insoluti in EURO
EIMPRICEVU	Numeric	19	Importo ricevuto insoluto in EURO
EIMPORTOIV	Numeric	19	Importo IVA in EURO
ECAMBIO	Numeric	19	Cambio in EURO
ARROTOND	Char	1	“L” compensazione in lire “E” compensazione in EURO e “N” (default) normale
VALCONT_SN	Char	1	“S” se il movimento è stato introdotto in valuta di conto uguale a quella dell’esercizio, altrimenti “N”
DOCUM_DEST	Char	3	codice luogo destinazione documento (utilizzato nella gestione degli effetti attivi).
PROBPAGAM	Char	1	Probabilità pagamento.
BLOCSCADE	Char	1	"S" se la scadenza è bloccata,"N" altrimenti
CIN	Char	1	Codice di Controllo CIN
BLOCSOLLE	Char	1	Vale ‘S’ se la scadenza non deve essere sollecitata al cliente; in tutti gli altri case deve essere ad N
MOTBLOCCO	Char	50	Motivo del blocco del sollecito (descrittivo) a senso solo con BLOCSOLLE=’S’
SCADENZA	Char	19	Per ogni pagamento che preveda la cancellazione di una scadenza si dovrà compilare nel tracciato di xmovco.txt il campo **XMOVCO.SCADENZA**. Il campo dovrà essere compilato con la chiave: **XMOVCO.NUMRIFSALD+MOVCO.NUMERO+MOVCO.RIGA.** In modo da individuare in maniera univoca la scadenza

			pagata/incassata da cancellare secondo la chiave : **MOVCO.NUMERO+MOVCO.RIGA.** Sarà eseguito un ulteriore controllo perché la cancellazione abbia luogo: **I primi sette caratteri di** **XMOVCO.SCADENZA** **Dovranno essere uguali a:** **XMOVCO.NUMRIFSALD** In modo da identificare sicuramente la volontà da parte dell'utente di cancellare le scadenze indicate. Per ottenere un pagamento/incasso parziale della scadenza indicata si devono compilari i seguenti campi. Per il solo record di partita (cliente/fornitore) P_PAREG_SN Riempire con 'S' EIMPORTO Contiene l'importo del pagamento parziale EIMPFATTUR Contiene il totale scadenza in Euro Se si tratta di un pagamento/incasso in valuta: MOV_VAL_SN Riempire con 'S' IMPVALUTA Contiene l'importo del pagamento in valuta IMPFATTURAContiene il totale scadenza in valuta. CODVALUTA Codice valuta DATACAMBIO Data del cambio ECAMBIO Valore del cambio Per il record di contropartita (Cassa o banca etc.etc) NUMPRBANC Riempire con '0000000'
IBAN	Char	34	Codice IBAN. Se il campo contiene la stringa : "IT99999999999999999999999999999999" il codice IBAN è calcolato automaticamente attraverso i campi • CODBANCA • CONTCORR • CIN Se non si ottiene un codice IBAN valido o uno dei campi precedenti è vuoto il campo risulterà vuoto. Il calcolo può essere fatto solo per clienti/fornitori italiani.
ESCBENSTR	Char	1	"S" per escludere dal calcolo dei totali beni strumentali in comunicazione annuale dati iva, "N" altrimenti.
CODEVENTO	Char	30	Codice di raggruppamento articoli contabili. Se presenti CIG e/o CUP deve essere uguale a CIG+CUP oppure vuoto.
CIG	Char	10	Codice identificativo gara. Utilizzato per tracciatura pagamenti P.A. non può essere vuoto se presente codice CUP.
CUP	Char	15	Codice unico progetto. Utilizzato per tracciatura pagamenti P.A.

N.B: Se i campi in euro sono omessi il tracciato sarà comunque accettato e i valori in euro saranno attenuti per conversione.

Struttura del file XMOVIV.TXT

Il File XMOVIV.TXT è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (Esadecimali 0D,0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione. Le date sono memorizzate come stringhe di caratteri in formato ANSI (AAAAMMGG).

Nome Campo	Tipo	Lung	Descrizione
NUMERO	Char	7	Numero univoco del raggruppamento di registrazioni,fillato di '0' a sinistra
RIGA	Char	3	Numero della riga all'interno del raggruppamento, fillato di '0' a sinistra
ANNOIVA	Char	4	anno di registrazione IVA
DATAREG	Char	8	Data registrazione
NUMPROTOC	Char	7	Numero protocollo, fillato di '0' a sinistra
DATADOC	Char	8	data documento
NUMERODOC	Char	10	Numero documento
SOTTOCONTO	Char	10	Codice sottoconto movimentato
C_F	Char	1	"C" per clienti, "F" per fornitori
DATAINIZIO	Char	8	**non gestito: NON RIEMPIRE**
DATAFINE	Char	8	**non gestito: NON RIEMPIRE**
TIPODOCIVA	Char	2	Codice tipo documento iva
REGISTRO	Char	2	Codice registro iva
REGCOLLEG	Char	2	Codice registro iva collegato (per operazioni che prevedono la movimentazionedi dueregistri iva)
NUMREGCOLL	Char	7	Numero della registrazione eseguita sul registro collegato
ANNOREGCOL	Char	4	anno di registrazione IVA del registro collegato
VALUTA	Char	3	Codice valuta estera
TOTFTVALUT	Numeric	19	Totale fattura in valuta
TOTFATTURA	Numeric	19	Totale fattura in lire
IMPONIBILE	Numeric	19	Imponibile in lire (il valore deve essere negativo per le note di credito)
CODIVA	Char	4	Codice aliquota iva
IMPOSTA	Numeric	19	Imposta in lire
IMPRATEO	Numeric	19	**non gestito: NON RIEMPIRE**
IMPRISCONT	Numeric	19	**non gestito: NON RIEMPIRE**
RITARDO_SN	Char	1	"S" per i movimenti di acquisto registrati in ritardo,"N" altrimenti
STLIQ_SN	Char	1	"S" se la liquidazione iva è stata stampata
STAMPAT_SN	Char	1	"S" se il movimento è stato stampato sul registro
MESEPLAFO	Char	2	**non gestito: NON RIEMPIRE**
ANNOPLAFO	Char	4	**non gestito: NON RIEMPIRE**
TEMPORA_SN	Char	1	**non gestito: NON RIEMPIRE**
PAGATA_SN	Char	1	"S" per le fatture con iva in sospensione completamente pagate
STATO	Char	1	"S" se record attivo,"N" altrimenti

UTENTE	Char	8	Codice utente che ha operato l'ultimo aggiornamento
ULT_AGG	Char	8	data ultimo aggiornamento
ANNOCOMPET	Char	4	Anno competenza IVA
MESECOMPET	Char	2	Mese di competenza IVA
SOSPESO_SN	Char	1	Iva in sospensione
ETOTFATTUR	Numeric	19	Totale fattura in EURO
EIMPONIBIL	Numeric	19	Imponibile in EURO
EIMPOSTA	Numeric	19	Imposta in EURO
VIMPONIBIL	Numeric	19	Importo fattura in valuta
VIMPOSTA	Numeric	19	Imposta fattura in valuta
RIFANNPREC	Char	1	"S" se il movimento identifica una nota di variazione riferita ad annualita' precedenti, "N" altrimenti.
COMUNIC_SN	Char	1	opzione per comunicazione operazioni IVA: "S" per includere, "N" per escludere
FORZATO_SN	Char	1	opzione per comunicazione forzata dall'utente
VALIDAT_SN	Char	1	validato per invio comunicazione
CODEVENTO	Char	30	codice evento
CODRIFERI	Char	7	numero riferimento fattura (significativo solo per le note di variazione)
MODPAGAM	Char	1	modalità di pagamento: "A" - importo non frazionato "B" - importo frazionato "C" - corrispettivo periodico

N.B.: La procedura se va a buon fine rinomina i file in XMOVCO.TXT in MCXXXXXX.TXT e XMOVIVA.TXT in MIXXXXXX.TXT dove XXXXXX indica la data odierna nel formato AAMMGG. Nel caso in cui si eseguono più importazioni nello stesso giorno i file XMOVCO.TXT e XMOVIVA.TXT sono rinominati solo alla prima importazione.

Importazione di massa movimenti di magazzino

La funzione (accessibile da programma CONFIG.EXE, icona configurazione) permette l'immissione automatica dei movimenti di magazzino, nonché l'aggiornamento dei saldi di giacenza/consistenza per magazzino/articolo, da file di testo esterni al pacchetto. Deve essere preparato un file ASCII sequenziale RMOVMA.TXT (ogni record è terminato con i caratteri ascii 0D e 0A).

Struttura del file RMOVMA.TXT

Il File RMOVMA.TXT è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (Esadecimali 0D,0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione. Le date sono memorizzate come stringhe di caratteri in formato ANSI (AAAAMMGG).

Nome Campo	Tipo	Lung	Descrizione
ESERCIZIO	Char	4	Esercizio contabile
NUMERO	Char	7	Numero univoco del raggruppamento di registrazioni,fillato di '0' a sinistra
RIGA	Char	5	Numero della riga all'interno del raggruppamento, fillato di '0' a sinistra
TIPO	Char	2	tipo record. **Deve** essere impostato a 'MM'
DATA	Char	8	data del movimento
ESEDOC	Char	4	Esercizio contabile del documento

NUMERODOC	Char	7	Numero documento
RIGADOC	Char	5	**non gestito: NON RIEMPIRE**
TIPODOC	Char	2	Codice tipo documento
DATADOC	Char	8	data documento
ESERCIPROT	Char	4	Esercizio protocollo
NUMEROPROT	Char	7	Numero protocollo, fillato di '0' a sinistra
RIGAPROT	Char	5	Numero di riga all'interno del numero protocollo, fillato di '0' a sinistra
TIPOPROT	Char	2	Codice tipo documento di protocollatura
DATAPROT	Char	8	data protocollo
TIPOEFFET	Char	1	tipo effettivo documento: B – bolla di carico G – bolla di scarico F – fattura R – fattura riepilogativa P - fattura proforma A - fattura accompagnatoria M - movimento di magazzino N - nota di credito O - ordine cliente T - ordine fornitore V - preventivo U - ricevuta fiscale S - scontrino C - documento generico
ESERIF	Char	4	esercizio documento di riferimento
NUMERORIF	Char	7	numero registrazione documento di riferimento
RIGARIF	Char	5	numero riga documento di riferimento
TIPORIF	Char	2	codice tipo doc. del documento di riferimento
DATARIF	Char	8	data documento di riferimento
ESECOLLEG	Char	4	esercizio documento collegato
NUMCOLLEG	Char	7	numero documento collegato
RIGACOLLEG	Char	5	numero riga documento collegato
TIPOCOLLEG	Char	2	codice tipo doc. del documento collegato
DATACOLLEG	Char	8	data documento collegato
DATAPREEVA	Char	8	data prevista evasione
ARTICOLO	Char	30	codice articolo di magazzino
VARIANTE	Char	10	**non gestito: NON RIEMPIRE**
STAGIONE	Char	3	**non gestito: NON RIEMPIRE**
MAGAZZINO	Char	3	codice magazzino
IMBALLO	Char	5	codice imballo
UNIMISURA	Char	3	unità di misura
MOVITAG_SN	Char	1	"S" se il movimento contiene l'esplosione delle taglie
TIPOTAGLIA	Char	2	codice taglia
CALZATA	Char	2	**non gestito: NON RIEMPIRE**

CAUSALE	Char	3	codice causale di magazzino
DESCRAGG	Char	20	descrizione aggiuntiva
C_F	Char	1	"C" per clienti, "F" per fornitori
CLI_FOR	Char	10	codice cliente/fornitore
CONTROPART	Char	10	codice sottoconto di costo/ricavo
CODVALUTA	Char	3	codice valuta estera
DATAVALUTA	Char	8	data cambio
CAMBIO	Numeric	19	rapporto di cambio con la lira
QUANTITA	Numeric	19	quantità
QUANTCONFE	Numeric	19	quantità per confezione
EVASO	Numeric	19	quantità evasa (solo per le righe d'ordine)
PRNETTO	Numeric	19	prezzo netto
PRLORDO	Numeric	19	prezzo lordo
PRIVACOMPR	Char	1	"S" se il prezzo è iva compresa
IVAOMAGGI	Char	1	"S" se deve essere addebitata l'iva sugli omaggi
PRLISTINO	Numeric	19	prezzo di listino
BASERICAR	Numeric	19	**non gestito: NON RIEMPIRE**
PERCRICAR	Numeric	19	**non gestito: NON RIEMPIRE**
PRVALUTA	Numeric	19	prezzo lordo in valuta
PRVALUTAN	Numeric	19	prezzo netto in valuta
SCONTO1	Numeric	19	percentuale primo sconto riga
SCONTO2	Numeric	19	percentuale secondo sconto riga
SCONTO3	Numeric	19	percentuale terzo sconto riga
SCONTO4	Numeric	19	percentuale quarto sconto riga
SCONTO5	Numeric	19	percentuale quinto sconto riga
MAGGIOR1	Numeric	19	percentuale prima maggiorazione riga
MAGGIOR2	Numeric	19	percentuale seconda maggiorazione riga
CODIVA	Char	4	codice aliquota iva
CODLISTIN	Char	5	codice listino applicato
PESOKG	Numeric	19	peso in kg.
NUMCOLLI	Numeric	19	numero colli
NUMPARTITA	Numeric	19	**non gestito: NON RIEMPIRE**
AGENTE	Char	5	codice agente
PROVAGENTE	Numeric	19	provvigioni agente in percentuale
CAPOZONA	Char	5	codice capozona
PROVCAPOZO	Numeric	19	provvigioni capozona in percentuale
SOSPESO_SN	Char	1	**non gestito: NON RIEMPIRE**
NOTA_SN	Char	1	"S" se alla record è associata una nota
COMM_SN	Char	1	"S" se il movimento è stato generato da una commessa di lavorazione

NETTO_SN	Char	1	**non gestito: NON RIEMPIRE**
OMAGGIO_SN	Char	1	"S" se il materiale è omaggio
DEFINIT_SN	Char	1	**non gestito: NON RIEMPIRE**
PROVVIS_SN	Char	1	**non gestito: NON RIEMPIRE**
MAGFISC_SN	Char	1	"S" se il movimento è effettuato su un magazzino fiscale
EVASION_SN	Char	1	"S" se il rigo è evaso (solo per gli ordini)
DISTINT_SN	Char	1	**non gestito: NON RIEMPIRE**
INVENTARIO	Char	1	"N" se non è stato movimentato il campo carico da inventario, "+" o "-" altrimenti
CARICO	Char	1	"N" se non è stato movimentato il campo carico da acquisto, "+" o "-" altrimenti
SCARICO	Char	1	"N" se non è stato movimentato il campo scarichi per vendita, "+" o "-" altrimenti
ACARICO	Char	1	"N" se non è stato movimentato il campo altri carichi, "+" o "-" altrimenti
ASCARICO	Char	1	"N" se non è stato movimentato il campo altri scarichi, "+" o "-" altrimenti
IMPEGNATO	Char	1	"N" se non è stato movimentato il campo impegnato, "+" o "-" altrimenti
ROTTAMATO	Char	1	"N" se non è stato movimentato il campo attesa rottamazione, "+" o "-" altrimenti
ORDCLI	Char	1	"N" se non è stato movimentato il campo ordinato clienti, "+" o "-" altrimenti
ORDFORN	Char	1	"N" se non è stato movimentato il campo ordinato fornitori, "+" o "-" altrimenti
ATTCOLLAUD	Char	1	"N" se non è stato movimentato il campo attesa collaudo, "+" o "-" altrimenti
ATTSPEDIZ	Char	1	"N" se non è stato movimentato il campo attesa spedizione, "+" o "-" altrimenti
AVALORE	Char	1	"S" se è il movimento è a valore
AQUANTITA	Char	1	"S" se è il movimento è a quantità
AGG_TOTALI	Char	1	"S" se è stato aggiornato il saldo totale di tutti i magazzini
SELEZIONE	Char	40	Codice di estrazione per fatturazione ad uso interno. **NON USARE**
NUMCARTELL	Numeric	19	**Non gestito: NON RIEMPIRE**
NUMCOMMESS	Numeric	19	**Non gestito: NON RIEMPIRE**
FILE1	Char	8	Nome dei files usati per la memorizzazione di fogli elettronici,testi di videoscrittura,immagini etc.
FILE2	Char	8	**Non gestito: NON RIEMPIRE**
FILE3	Char	8	**Non gestito: NON RIEMPIRE**
UTENTE	Char	8	codice utente che ha operato l'ultimo aggiornamento
ULT_AGG	Char	8	data ultimo aggiornamento
PRSTATIST1	Numeric	19	prezzo statistico 1
PRSTATIST2	Numeric	19	prezzo statistico 2
PRSTATIST3	Numeric	19	prezzo statistico 3
CENTRCOSTO	Char	6	codice centro di costo
REVISIO_SN	Char	1	"S" il movimento è revisionabile,"N" altrimenti
AGGIORN_SN	Char	1	"S" il movimento è stato aggiornato da SIGLA,"N" altrimenti
CHIAVE_EST	Char	30	chiave esterna a disposizione di procedure integrate con SIGLA
FORZEVA_SN	Char	1	Impostare a "N"

DESARTICOL	Char	40	descrizione articolo
ECAMBIO	Numeric	19	Rapporto di cambio con l'euro
EPRNETTO	Numeric	19	prezzo netto in euro
EPRLORDO	Numeric	19	prezzo lordo in euro
EPRLISTINO	Numeric	19	prezzo di listino in euro
EBASERICARI	Numeric	19	**Non gestito: NON RIEMPIRE**
UBICAZIONE	Char	10	Codice ubicazione (obbligatorie se configurata la giacenza a ubicazione)
QTACQUISTO	Numeric	19	Quantità in unità di misura di acquisto
PRACQUISTO	Numeric	19	prezzo unitario lordo riferito alla quantità espressa in unità di misura di acquisto.
UNITAMIACQ	Char	3	unità di misura di acquisto

N.B.: i campo PRSTATIST1,PRSTATIST2 e PRSTATIST3 sono stati aggiunti a partire dalla release 1.61 del prodotto. Per compatibilità con le versioni precedenti la procedura di importazione tuttavia continua a lavorare correttamente anche se tali dati non sono stati inseriti nel tracciato (vengono in questo caso impostati a 0)

Sul file viene controllato che:

- il campo 'NUMERO' contenga un valore significativo (numero di sette digit zerofilled).
- il campo 'ARTICOLO' contenga il codice di un articolo di magazzino esistente e non sia un articolo gestito a Lotti.
- il campo 'CAUSALE' contenga il codice di una causale di magazzino esistente.
- il campo 'CLI_FOR' contenga il codice di un cliente/fornitore esistente.
- il campo 'UBICAZIONE' contenga un codice ubicazione esistente per gli articoli (ed eventualmente mgazzino) a ubicazione.

N.B.: La procedura se va a buon fine rinomina il file RMOVMA.TXT in MMXXXXXX.TXT dove XXXXXX indica la data odierna nel formato AAMMGG. Nel caso in cui si eseguono più importazioni nello stesso giorno il file RMOVMA.TXT è rinominato solo alla prima importazione.

Importazione di massa documenti di magazzino

La funzione (accessibile da programma CONFIG32.EXE, icona configurazione) permette l'immissione automatica dei documenti di magazzino (ordini, bolle etc.), nonché l'aggiornamento dei saldi di giacenza/consistenza per magazzino/articolo,da file di testo esterni al pacchetto. Devono essere preparati due file ASCII sequenziali: TDOCUM.TXT destinato a contenere le testate dei documenti e RDOCUM.TXT destinato a contenere le righe (ogni record è terminato con i caratteri ascii 0D e 0A).

Struttura del file TDOCUM.TXT

Il File TDOCUM.TXT è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (Esadecimali 0D,0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione. Le date sono memorizzate come stringhe di caratteri in formato ANSI (AAAAMMGG).

Nome Campo	Tipo	Lung	Descrizione
NUMERO	Char	7	numero univoco della registrazione, fillato di '0' a sinistra

ESERPROTOC	Char	4	esercizio protocollo
NUMEPROTOC	Char	7	numero protocollo
TIPOPROTOC	Char	2	codice tipo documento protocollo
DATAPROTOC	Char	8	data protocollo
ESERRIFERI	Char	4	esercizio documento di riferimento
NUMERIFERI	Char	7	numero documento di riferimento
TIPORIFERI	Char	2	codice tipo doc. documento di riferimento
DATARIFERI	Char	8	data documento di riferimento
ESERCOLLEG	Char	4	esercizio documento collegato
NUMECOLLEG	Char	7	numero documento collegato
TIPOCOLLEG	Char	2	codice tipo doc. documento collegato
DATACOLLEG	Char	8	data documento collegato
ESERDEFINI	Char	4	esercizio documento definitivo
NUMEDEFINI	Char	7	numero documento definitivo
TIPODEFINI	Char	2	codice tipo doc. documento definitivo
DATADEFINI	Char	8	data documento definitivo
NUMEDOCUME	Char	7	numero documento
DATADOCUME	Char	8	data documento
DATAPREEVA	Char	8	data prevista evasione (per ordini)
C_F	Char	1	"C" per clienti, "F" per fornitori
CLI_FOR	Char	10	codice cliente/fornitore
DESTINATAR	Char	10	cliente destinatario della fattura
LUOGO_DEST	Char	3	codice luogo destinazione merce
DOCUM_DEST	Char	3	codice luogo destinazione documento
MAGAZZINO	Char	3	codice magazzino
MAGACOLLEG	Char	3	codice magazzino documento collegato
PAGAMENTO	Char	4	codice pagamento
DATADECPAG	Char	8	data decorrenza pagamento
SCONTOVALU	Numeric	19	sconto fisso
SCONTOPAGA	Numeric	19	sconto pagamento
ABI	Char	5	codice abi banca d'appoggio
CAB	Char	5	codice cab banca d'appoggio
ASPESTBENI	Char	5	codice aspetto esteriore beni
NUMCOLLI	Numeric	19	numero colli
ADDESPE_SN	char	1	"S" se devono essere addebitate le spese
SPETRASPOR	Numeric	19	spese di trasporto
IVATRASPOR	Char	4	codice iva per spese trasporto
SPESEVARIE	Numeric	19	spese varie
IVASPEVARI	Char	4	codice iva per spese varie
ADDEBOL_SN	Char	1	"S" se devono essere addebitati i bolli

IVABOLLI	Char	4	codice iva per i bolli
DATACONSE	Char	8	data consegna
ORACONSE	Char	5	ora consegna (HH:MM)
DATAPARTEN	Char	8	data partenza
ORAPARTEN	Char	5	ora partenza (HH:MM)
AGENTE	Char	5	codice agente
PROVVIAGEN	Numeric	19	percentuale provvigioni agente
CAPOZONA	Char	5	codice capozona
PROVVICAZO	Numeric	19	percentuale provvigioni capozona
TRASPACURA	Char	1	trasporto a cura: M - mittente V - vettore D - destinatario
PESOKG	Numeric	19	peso in kg.
VETTORE_1	Char	5	codice vettore 1
VETTORE_2	Char	5	codice vettore 2
VETTORE_3	Char	5	codice vettore 3
SPEDIZIO_1	Char	5	codice spedizioniere 1
SPEDIZIO_2	Char	5	codice spedizioniere 2
PORTO	Char	3	codice porto
MODALSPEDI	Char	3	codice modalità spedizione
ANNOTAZION	Char	60	annotazioni
LISTINO	Char	5	codice listino applicato
FILLER	Char	19	Lasciare 19 spazi vuoti.
SCONTO1	Numeric	19	primo sconto riga da proporre
SCONTO2	Numeric	19	secondo sconto riga da proporre
SCONTO3	Numeric	19	terzo sconto riga da proporre
SCONTO4	Numeric	19	quarto sconto riga da proporre
SCONTO5	Numeric	19	quinto sconto riga da proporre
MAGGIORAZ1	Numeric	19	prima maggiorazione riga da proporre
EVASO_SN	Char	1	"S" se l'ordine è completamente evaso (solo per gli ordini)
STAMPAT_SN	Char	1	"S" se il documento è stato stampato
DEFINIT_SN	Char	1	"S" se il documento è definitivo
DAFATTU_SN	Char	1	"S" se il documento è da fatturare con fattura riepilogativa
ASSOGCLIEN	Char	4	codice iva assoggettamento cliente
IVAMASSIMA	Char	4	**non gestito: NON RIEMPIRE**
TIPODOCUM	Char	1	tipo effettivo documento: B - bolla di carico G - bolla di scarico F - fattura R - fattura riepilogativa P - fattura proforma A - fattura accompagnatoria M - movimento di magazzino

			N - nota di credito O - ordine cliente T - ordine fornitore V - preventivo U - ricevuta fiscale S - scontrino C - documento generico
VALUTA	Char	3	codice valuta estera
CAMBIO	Numeric	19	rapporto di cambio con la lira
DATACAMBIO	Char	8	data del cambio
NAZIONE	Char	3	codice stato estero
TOTIMPORTO	Numeric	19	totale documento in lire
TOTIMPVALU	Numeric	19	totale documento in valuta
CAUSTRASPO	Char	3	codice causale trasporto
CAUSMAGAZZ	Char	3	codice causale di magazzino
CAUSCONTAB	Char	3	codice causale contabile
FATTURATA	Char	1	"S" se il documento è stato fatturato (solo per le bolle)
LINGUA	Char	1	codice lingua
CENTRCOSTO	Char	6	codice centro di costo
FLAG1	Char	1	**non gestito: NON RIEMPIRE**
FLAG2	Char	1	**non gestito: NON RIEMPIRE**
FLAG3	Char	1	**non gestito: NON RIEMPIRE**
FLAG4	Char	1	**non gestito: NON RIEMPIRE**
FLAG5	Char	1	**non gestito: NON RIEMPIRE**
FILE1	Char	8	nome dei files usati per la memorizzazione di fogli elettronici,testi di videoscrittura, immagini etc.
FILE2	Char	8	**non gestito: NON RIEMPIRE**
FILE3	Char	8	**non gestito: NON RIEMPIRE**
FILE4	Char	8	**non gestito: NON RIEMPIRE**
FILE5	Char	8	**non gestito: NON RIEMPIRE**
STATO	Char	1	"S" se record attivo,"N" altrimenti
UTENTE	Char	8	codice utente che ha operato l'ultimo aggiornamento
ULT_AGG	Char	8	data ultimo aggiornamento
FATIVASOSP	Char	1	"S" se il documento deve essere fatturato con iva in sospensione

DESASPBENI	Char	60	descrizione aspetto beni
RGSOC_DEST	Char	50	ragione sociale destinazione merce
INDIR_DEST	Char	35	indirizzo destinazione merce
LOCAL_DEST	Char	35	località destinazione merce
CAP_DEST	Cha	5	cap destinazione merce
STAES_DEST	Char	3	codice stato estero destinazione merce
COM_DEST	Char	4	codice comune destinazione merce
DCOM_DEST	Char	50	descrizione comune destinazione merce
PTEL_DEST	Char	10	prefisso telefonico destinazione merce

TEL_DEST	Char	30	numero telefonico destinazione merce
PFAX_DEST	Char	10	prefisso fax destinazione merce
FAX_DEST	Char	30	numero fax destinazione merce
PRBOLLA_SN	Char	1	"S" se devono essere stampati i prezzi sulla bolla (solo per bolle accompagnatorie)
SPETRAVAL	Numeric	19	spese trasporto in valuta
SPEVARVAL	Numeric	19	spese varie in valuta
NUMEDOCRIF	Char	10	numero documento di riferimento
DATADOCRIF	Char	8	data documento di riferimento
TIPODOCIVA	Char	2	tipo documento iva
REVISIO_SN	Char	1	"S" il documento è revisionabile,"N" altrimenti
ACCONTO	Numeric	19	acconto riscosso
VACCONTO	Numeric	19	acconto in valuta riscosso
PESONETTO	Numeric	19	peso netto
VOLUME	Numeric	19	Volume
TARA	Numeric	19	tara
ESPETRASPO	Numeric	19	spese trasporto in euro
ESPESEVARI	Numeric	19	spese varie in euro
ECAMBIO	Numeric	19	rapporto di cambio con l'euro
ETOTIMPORT	Numeric	19	totale documento in euro
EACCONTO	Numeric	19	acconto riscosso in euro
VALCONT	Char	1	"E" se il documento è stato immesso in euro, "L" altrimenti
ADDSPPA_SN	Char	1	"S" se devono essere addebitate le spese di pagamento
CODSPEDIZI	Char	25	Codice spedizione (usato da NET++) . Dove ‘‘=”Generico”;’A’=”UPS”;B=”DHL”;’C’=”SDA”.
IDVETTORE	Char	1	tipo vettore (usato da NET++)
PROV_DEST	Char	5	provincia destinazione merce
CASSA	Char	10	sottoconto cassa
RDA_SN	Char	1	Se **S** indica che il documento è soggetto a ritenuta di acconto nei casi previsti dalla 296/2006 (comma 43 dell' articolo 1) per i condominii; altrimenti vale **N**
RDAPERCENT	Numeric	19	Percentuale ritenuta di acconto da configurazione (attualmente al 4%)
IVAINCASSO	Char	4	Codice iva incasso.
KGCO2MEDIA	Numeric	19	Chilogrammi CO2 emessi
KMSPEDITI	Numeric	19	Chilometri totali percorsi
CALCOLACO2	Char	1	Dati immessi dall'utente=’N’; cacolato da SIGLA in automatico =’S’
TRUCKKMCO2	Numeric	19	Chilometri percorsi con camion
TRAINKMCO2	Numeric	19	Chilometri percorsi in treno
SHIPKMCO2	Numeric	19	Chilometri percorsi in Nave
PLANEKMCO2	Numeric	19	Chilometri percorsi in aereo
CIG	Char	10	Codice CIG

CUP	Char	15	Codice CUP

Sul file viene controllato che:

- Il campo 'CLI_FOR' contenga il codice di un cliente/fornitore esistente.
- Il campo 'LUOGO_DEST' contenga il codice di un indirizzo di spedizione valido

Struttura del file RDOCUM.TXT

Il file RDOCUM.TXT è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (Esadecimali 0D, 0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione. Le date sono memorizzate come stringhe di caratteri in formato ANSI (AAAAMMGG).

Nome Campo	Tipo	Lung	Descrizione
ESERCIZIO	Char	4	esercizio contabile
NUMERO	Char	7	numero univoco del raggruppamento di registrazioni,fillato di '0' a sinistra
RIGA	Char	5	numero della riga all'interno del raggruppamento, fillato di '0' a sinistra
TIPO	Char	2	tipo record. **Deve** essere impostato a 'DC'
DATA	Char	8	data del movimento
ESEDOC	Char	4	esercizio contabile del documento
NUMERODOC	Char	7	numero documento
RIGADOC	Char	5	**non gestito: NON RIEMPIRE**
TIPODOC	Char	2	codice tipo documento
DATADOC	Char	8	data documento
ESERCIPROT	Char	4	esercizio protocollo
NUMEROPROT	Char	7	numero protocollo, fillato di '0' a sinistra
RIGAPROT	Char	5	numero di riga all'interno del numero protocollo, fillato di '0' a sinistra
TIPOPROT	Char	2	codice tipo documento di protocollatura
DATAPROT	Char	8	data protocollo
TIPOEFFET	Char	1	tipo effettivo documento: B - bolla di carico G - bolla di scarico F - fattura R - fattura riepilogativa P - fattura proforma A - fattura accompagnatoria M - movimento di magazzino N - nota di credito O - ordine cliente T - ordine fornitore V - preventivo U - ricevuta fiscale S - scontrino C - documento generico
ESERIF	Char	4	esercizio documento di riferimento
NUMERORIF	Char	7	numero registrazione documento di riferimento
RIGARIF	Char	5	numero riga documento di riferimento

TIPORIF	Char	2	codice tipo doc. del documento di riferimento
DATARIF	Char	8	data documento di riferimento
ESECOLLEG	Char	4	esercizio documento collegato
NUMCOLLEG	Char	7	numero documento collegato
RIGACOLLEG	Char	5	numero riga documento collegato
TIPOCOLLEG	Char	2	codice tipo doc. del documento collegato
DATACOLLEG	Char	8	data documento collegato
DATAPREEVA	Char	8	data prevista evasione
ARTICOLO	Char	30	codice articolo di magazzino
VARIANTE	Char	10	**non gestito: NON RIEMPIRE**
STAGIONE	Char	3	**non gestito: NON RIEMPIRE**
MAGAZZINO	Char	3	codice magazzino
IMBALLO	Char	5	codice imballo
UNIMISURA	Char	3	unità di misura
MOVITAG_SN	Char	1	"S" se il movimento contiene l'esplosione delle taglie
TIPOTAGLIA	Char	2	codice taglia
CALZATA	Char	2	**non gestito: NON RIEMPIRE**
CAUSALE	Char	3	codice causale di magazzino
DESCRAGG	Char	20	descrizione aggiuntiva
C_F	Char	1	"C" per clienti, "F" per fornitori
CLI_FOR	Char	10	codice cliente/fornitore
CONTROPART	Char	10	codice sottoconto di costo/ricavo
CODVALUTA	Char	3	codice valuta estera
DATAVALUTA	Char	8	data cambio
CAMBIO	Numeric	19	rapporto di cambio con la lira
QUANTITA	Numeric	19	quantità
QUANTCONFE	Numeric	19	quantità per confezione
EVASO	Numeric	19	quantità evasa (solo per le righe d'ordine)
PRNETTO	Numeric	19	prezzo netto
PRLORDO	Numeric	19	prezzo lordo
PRIVACOMPR	Char	1	"S" se il prezzo è iva compresa
IVAOMAGGI	Char	1	"S" se deve essere addebitata l'iva sugli omaggi
PRLISTINO	Numeric	19	prezzo di listino
BASERICAR	Numeric	19	**non gestito: NON RIEMPIRE**
PERCRICAR	Numeric	19	**non gestito: NON RIEMPIRE**
PRVALUTA	Numeric	19	prezzo lordo in valuta
PRVALUTAN	Numeric	19	prezzo netto in valuta
SCONTO1	Numeric	19	percentuale primo sconto riga
SCONTO2	Numeric	19	percentuale secondo sconto riga

SCONTO3	Numeric	19	percentuale terzo sconto riga
SCONTO4	Numeric	19	percentuale quarto sconto riga
SCONTO5	Numeric	19	percentuale quinto sconto riga
MAGGIOR1	Numeric	19	percentuale prima maggiorazione riga
MAGGIOR2	Numeric	19	percentuale seconda maggiorazione riga
CODIVA	Char	4	codice aliquota iva
CODLISTIN	Char	5	codice listino applicato
PESOKG	Numeric	19	peso in kg.
NUMCOLLI	Numeric	19	numero colli
NUMPARTITA	Numeric	19	**non gestito: NON RIEMPIRE**
AGENTE	Char	5	codice agente
PROVAGENTE	Numeric	19	provvigioni agente in percentuale
CAPOZONA	Char	5	codice capozona
PROVCAPOZO	Numeric	19	provvigioni capozona in percentuale
SOSPESO_SN	Char	1	**non gestito: NON RIEMPIRE**
NOTA_SN	Char	1	"S" se alla record è associata una nota
COMM_SN	Char	1	"S" se il movimento è stato generato da una commessa di lavorazione
NETTO_SN	Char	1	**non gestito: NON RIEMPIRE**
OMAGGIO_SN	Char	1	"S" se il materiale è omaggio
DEFINIT_SN	Char	1	**non gestito: NON RIEMPIRE**
PROVVIS_SN	Char	1	**non gestito: NON RIEMPIRE**
MAGFISC_SN	Char	1	"S" se il movimento è effettuato su un magazzino fiscale
EVASION_SN	Char	1	"S" se il rigo è evaso (solo per gli ordini)
DISTINT_SN	Char	1	**non gestito: NON RIEMPIRE**
INVENTARIO	Char	1	"N" se non è stato movimentato il campo carico da inventario, "+" o "-" altrimenti
CARICO	Char	1	"N" se non è stato movimentato il campo carico da acquisto, "+" o "-" altrimenti
SCARICO	Char	1	"N" se non è stato movimentato il campo scarichi per vendita, "+" o "-" altrimenti
ACARICO	Char	1	"N" se non è stato movimentato il campo altri carichi, "+" o "-" altrimenti
ASCARICO	Char	1	"N" se non è stato movimentato il campo altri scarichi, "+" o "-" altrimenti
IMPEGNATO	Char	1	"N" se non è stato movimentato il campo impegnato, "+" o "-" altrimenti
ROTTAMATO	Char	1	"N" se non è stato movimentato il campo attesa rottamazione, "+" o "-" altrimenti
ORDCLI	Char	1	"N" se non è stato movimentato il campo ordinato clienti, "+" o "-" altrimenti
ORDFORN	Char	1	"N" se non è stato movimentato il campo ordinato fornitori, "+" o "-" altrimenti
ATTCOLLAUD	Char	1	"N" se non è stato movimentato il campo attesa collaudo, "+" o "-" altrimenti
ATTSPEDIZ	Char	1	"N" se non è stato movimentato il campo attesa spedizione, "+" o "-" altrimenti
AVALORE	Char	1	"S" se è il movimento è a valore
AQUANTITA	Char	1	"S" se è il movimento è a quantità

AGG_TOTALI	Char	1	"S" se è stato aggiornato il saldo totale di tutti i magazzini
SELEZIONE	Char	40	codice di estrazione per fatturazione ad uso interno. **NON USARE**
NUMCARTEL	Numeric	19	**non gestito: NON RIEMPIRE**
NUMCOMMES	Numeric	19	**non gestito: NON RIEMPIRE**
FILE1	Char	8	nome dei files usati per la memorizzazione di fogli elettronici,testi di videoscrittura,immagini etc.
FILE2	Char	8	**non gestito: NON RIEMPIRE**
FILE3	Char	8	**non gestito: NON RIEMPIRE**
UTENTE	Char	8	codice utente che ha operato l'ultimo aggiornamento
ULT_AGG	Char	8	data ultimo aggiornamento
PRSTATIST1	Numeric	19	prezzo statistico 1
PRSTATIST2	Numeric	19	prezzo statistico 2
PRSTATIST3	Numeric	19	prezzo statistico 3
CENTROCOSTO	Char	6	Codice Centro di costo
REVISIO_SN	Char	1	"S" il movimento è revisionabile,"N" altrimenti
AGGIORN_SN	Char	1	"S" il movimento è stato aggiornato da SIGLA,"N" altrimenti
CHIAVE_EST	Char	30	chiave esterna a disposizione di procedure integrate con SIGLA
FORZEVA_SN	Char	1	impostare a "N"
DESARTICOL	Char	40	descrizione articolo
ECAMBIO	Numeric	19	rapporto di cambio con l'euro
EPRNETTO	Numeric	19	prezzo netto in euro
EPRLORDO	Numeric	19	prezzo lordo in euro
EPRLISTINO	Numeric	19	prezzo di listino in euro
EBASERICARI	Numeric	19	**Non gestito: NON RIEMPIRE**
UBICAZIONE	Char	10	Codice ubicazione (obbligatorie se configurata la giacenza a ubicazione) .
QTACQUISTO	Numeric	19	quantità in unità di misura di acquisto
PRACQUISTO	Numeric	19	prezzo unitario lordo riferito alla quantità espressa in unità di misura di acquisto
UNITAMIACQ	Char	3	unità di misura di acquisto
BARCODE	Char	21	Codice barcode
RETSERV_SN	Char	1	"S" se la riga deve essere considerato come rettifica di una riga di servizi per gestione INTRA, "N" altrimenti
SEZDOGAN	Char	6	sezione doganale per gestione INTRA
PROTINVIO	Char	6	numero protocollo invio telematico (per gestione INTRA)
PROGSEZIO	Char	5	progressivo sezione invio telematico (per gestione INTRA)
DATA_RET	Char	8	data fattura da rettificare (per gestione INTRA)
NUMERO_RET	Char	15	numero fattura da rettificare (per gestione INTRA)
EIMPSRVRET	Numeric	19	importo rettifica servizi per gestione INTRA
DESARCOMPL	Char	120	Descrizione articolo di 120 caratteri. (N.B: quando è configurata la descrizione estesa DESARTICOL rappresenta i primi quaranta caratteri della descrizione estesa.)
DESCRART01	Char	40	Secondi quaranta caratteri della descrizione estesa
DESCRART02	Char	40	Terzi quaranta caratteri della descrizione estesa

CIG	Char	10	Codice identificativo gara. Utilizzato per tracciatura pagamenti P.A. non può essere vuoto se presente codice CUP.
CUP	Char	15	Codice unico progetto. Utilizzato per tracciatura pagamenti P.A.

N.B.: i campo PRSTATIST1,PRSTATIST2 e PRSTATIST3 sono stati aggiunti a partire dalla release 1.61 del prodotto. Per compatibilità con le versioni precedenti la procedura di importazione tuttavia continua a lavorare correttamente anche se tali dati non sono stati inseriti nel tracciato (vengono in questo caso impostati a 0)

Sul file viene controllato che:

- il campo 'ARTICOLO' contenga il codice di un articolo di magazzino esistente e non sia un articolo gestito a Lotti.
- il campo 'CAUSALE' contenga il codice di una causale di magazzino esistente.
- il campo 'CLI_FOR' contenga il codice di un cliente/fornitore esistente.
- il campo 'UBICAZIONE' contenga un codice ubicazione esistente per gli articoli (ed eventualmente mgazzino) a ubicazione.

N.B.: La procedura se va a buon fine rinomina i file in TDOCUM.TXT in MTXXXXXX.TXT e RDOCUM.TXT in MRXXXXXX.TXT dove XXXXXX indica la data odierna nel formato AAMMGG. Nel caso in cui si eseguono più importazioni nello stesso giorno i file TDOCUM.TXT e RDOCUM.TXT sono rinominati solo alla prima importazione.

Importazione di massa rilevato di inventario

La funzione (accessibile da menù inventati) permette l'immissione automatica dei valori di rilevato di magazzino degli articoli inseriti in un inventario. Nel caso non siano definiti i listini si possono inserire nel file anche i prezzi (in lire o euro oppure entrambi) degli articoli. Deve essere preparato un file ASCII sequenziale contenente i dati da inserire negli archivi righe inventario ed eventulamnte se gestite delle tagli inventario (ogni record è terminato con i caratteri ascii 0D e 0A). Il file puo avere un nome qualsiasi e deve avere estensione .TXT.

Struttura del file

Il file è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (Esadecimali 0D, 0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione.

Nome Campo	Tipo	Lung	Descrizione
ARTICOLO	Char	30	Articolo di magazzino
MAGAZZINO	Char	3	Magazzino (se configurato)
PREZZO LIRE	Numeric	19	Prezzo in lire dell'articolo (se non configurati i listini)
PREZZO EURO	Numeric	19	Prezzo in euro dell'articolo (se non configurati i listini)
RILEVATO TOT.	Numeric	19	Rilevato totale
RILEVATO TAG.01	Numeric	19	Rilevato per taglia
RILEVATO TAG.02	Numeric	19	Rilevato per taglia

RILEVATO TAG.03	Numeric	19	Rilevato per taglia
RILEVATO TAG.04	Numeric	19	Rilevato per taglia
RILEVATO TAG.05	Numeric	19	Rilevato per taglia
RILEVATO TAG.06	Numeric	19	Rilevato per taglia
RILEVATO TAG.07	Numeric	19	Rilevato per taglia
RILEVATO TAG.08	Numeric	19	Rilevato per taglia
RILEVATO TAG.09	Numeric	19	Rilevato per taglia
RILEVATO TAG.10	Numeric	19	Rilevato per taglia
RILEVATO TAG.11	Numeric	19	Rilevato per taglia
RILEVATO TAG.12	Numeric	19	Rilevato per taglia
RILEVATO TAG.13	Numeric	19	Rilevato per taglia
RILEVATO TAG.14	Numeric	19	Rilevato per taglia
RILEVATO TAG.15	Numeric	19	Rilevato per taglia
RILEVATO TAG.16	Numeric	19	Rilevato per taglia
RILEVATO TAG.17	Numeric	19	Rilevato per taglia
RILEVATO TAG.18	Numeric	19	Rilevato per taglia
RILEVATO TAG.19	Numeric	19	Rilevato per taglia
RILEVATO TAG.20	Numeric	19	Rilevato per taglia

Al termine dell'importazione di massa viene prodotta una stampa con riportati eventuali errori rilevati in fase di immissione di massa. Le righe con errori non verranno importate. Lunghezza del file 470 caratteri

Importazione di massa rilevato di inventario con ubicazioni

Contiene tutti i campi precedenti e in aggiunta:

UBICAZIONE	CHAR	10	Codice Ubicazione

Al termine dell'importazione di massa viene prodotta una stampa con riportati eventuali errori rilevati in fase di immissione di massa. Le righe con errori non verranno importate. Lunghezza del file 480 caratteri

Importazione di massa aperture contabili

La funzione (accessibile da programma CONFIG32.EXE, icona configurazione) permette l'immissione automatica dei movimenti di apertura contabile, nonché l'aggiornamento dei progressivi sottoconto, da file di testo esterni al pacchetto. Deve essere preparato un file ASCII sequenziale (ogni record è terminato con i caratteri ascii 0D e 0A), **XMOVAP.TXT**.

Lo scopo della procedura è di permettere il trasferimento veloce delle aperture contabili e delle partite ancora aperte da un sistema informativo diverso da SIGLA.

La tipologia delle righe immesse prevede che a parità di numero si registri un unico articolo contabile ed un'unica partita. La parte del movimento complessivo di apertura che si deve immettere è quella relativa alla partita. La riga di contropartita è generata automaticamente a partire dai dati richiesti a video (bilancio di apertura). Per generare le scadenze relative alle partite aperte si deve registrare una riga per ogni scadenza. Così come avviene in SIGLA la scadenza sarà registrata tramite i dati indicati in Configurazione "Dati Standard #1,#2,#3" (causale contabile e sottoconto di contropartite).

Struttura del file XMOVAP.TXT

Il File XMOVAP.TXT è sequenziale, ogni record è terminato con i caratteri ascii 13 e 10 (esadecimali 0D, 0A).

I campi numerici sono comprensivi dei decimali e del punto (.) di separazione. Le date sono memorizzate come stringhe di caratteri in formato ANSI (AAAAMMGG).

Nome Campo	Tipo	Lung.	Descrizione
NUMERO	Char	7	Numero univoco del raggruppamento di registrazioni che aprono il Cliente/Fornitore ,fillato di '0' a sinistra. Deve partire da 1
SOTTOCONTO	Char	10	Sottoconto movimentato
IMPORTO	Numeric	19	Importo movimento; positivose Dare negativo se Avere
DATASCAD	Char	8	data scadenza (per movimenti di scadenzario)
DATAPROTOC	Char	8	Data protocollo
NUMPROTOC	Char	7	Numero protocollo
DATADOCUM	Char	8	Data documento
NUMDOCUM	Char	10	Numero documento
TIPOEFFETT	Char	1	Tipo effettivo di pagamento: 0 – rimessa diretta o contanti 1 – tratta 2 – ricevuta bancaria 3 – cessione credito 4 – pagherò 5 – lettera di credito 6 – tratta accettata 7 – RiBa 8 – ritardato pagamento 9 – cambiale A – altro pagamento B – bonifico bancario
IMPFATTURA	Numeric	19	Importo totale fattura (per effetti)
NUMRATA	Numeric	5	Numero rata (per effetti)
TOTRATE	Numeric	4	Numero totale rate (per effetti)
VALCONT_SN	Char	1	"L" o " " (blank) se il movimento è stato introdotto in lire "E" se il movimento è in euro

Per ogni riga del file che ha il campo TIPOEFFETT non vuoto (non blank) sarà generata una riga di scadenziario con le stesse modalità dell'immissione manuale da Prima Nota Generale.

Il programma controlla che:

- il campo numero sia diverso da zero (la numerazione deve essere progressiva e partire da 1);
- il campo sottoconto sia un cliente o fornitore esistente;
- il campo tipo effetto sia fra quelli ammessi;
- il campo data scadenza sia riempito se il campo tipo effetto è riempito e valido;
- nel caso sia attivo l'eurokit sulla ditta il campo VALCONT_SN sia riempito ed abbia un valore valido;
- nel caso non sia attivo l'eurokit il campo non è letto e può essere omesso;
- il segno del campo importo (se positivo il movimento sarà in DARE altrimenti in AVERE).

Gestione delle fincature

SIGLA permette di generare direttamente su una stampante grafica le fincature di stampa per i vari tipi di documento gestiti (a partire dalla versione **1.61**)**. La fincatura viene sovrapposta alla stampa solo al momento della produzione fisica della pagina su carta**. Al momento l'eventuale presenza della fincatura non è visibile nelle anteprime di stampa a video.

Ciascuna fincatura da usare deve essere descritta all'interno di un file ASCII, manutenibile con un comune editor, il cui nome deve terminare con il postfisso **.SRC** (a.e. FATTURA.SRC). Il file deve essere memorizzato nella directory di lavoro di SIGLA.

Le fincature sono ridimensionate automaticamente dall'applicazione in funzione delle dimensioni del font scelto per la stampa. **Tuttavia se le dimensioni del font sono tali da non permettere la completa riproduzione della fincatura sulla pagina di stampa possono essere ottenuti risultati non determinabili.** Si consiglia pertanto di fissare la dimensione del font da usare su ciascuna fincatura in modo tale che l'utente non possa alterarlo (vedi punti successivi).

L'associazione delle fincature da usare ai vari tipi documento viene effettuata tramite l'uso della procedura di configurazione. In corrispondenza della definizione del layout di stampa dei vari documenti è possibile associare ad ognuno di essi il nome del file di descrizione della fincatura (massimo 8 caratteri) da usare. Il postfisso .SRC viene aggiunto automaticamente dal programma **e non deve quindi essere digitato.** Nella stessa fase è possibile prestabilire la dimensione del font da usare per la stampa. Se questo dato non viene impostato l'utente sarà lasciato libero di variare le dimensioni del font a run time (e quindi l'applicazione procederà automaticamente al ridimensionamento della fincatura in funzione del font usato).

Struttura del file di descrizione del fincato

I files di descrizione dei fincati utilizzati sono composti da una serie di righe ognuna delle quali descrive una azione elementare di disegno del modulo. La struttura della riga è **posizionale** e le posizioni devono essere rigidamente rispettate. Il primo carattere della riga stabilisce il tipo di azione da intraprendere mentre la restante parte della riga riporta i parametri necessari per l'azione. **Tutte le coordinate necessarie per le funzioni grafiche sono espresse in decimi di carattere** (a.e. la coordinata orizzontale 0100 rappresenta l'inizio del carattere a colonna 10). Le coordinate dell'angolo alto sinistro del foglio sono x=0000, y=0000 (intendendo con x l'asse orizzontale e con y l'asse verticale). Questo accorgimento permette di tracciare linee o riquadri che non si sovrappongono ai caratteri di stampa (e' sufficiente "shiftare" di qualche decimo di carattere le linee).

Il file può contenere delle righe di commento che devono iniziare con il carattere <**;>** (punto e virgola) e non si devono trovare all'interno di una riga di comando.

```
; Questo è un commento
;
; Ogni riga di commento può contenere qualunque cosa
```

Disegno di una linea

La riga di comando per disegnare una linea ha la seguente sintassi:

Lpp xxxx yyyy XXXX YYYY
dove:

* **L** in prima posizione individua il tipo di azione da intraprendere (disegno di una linea)
* **pp** è un numero espresso con due cifre e compreso fra **00** e **20** che rappresenta lo spessore della linea da tracciare (00 è la linea più sottile, 20 la più spessa)
* **xxxx** e **yyyy** rappresentano le coordinate del punto di inizio della linea espresse in decimi di carattere
* **XXXX** e **YYYY** rappresentano le coordinate del punto finale della linea espresse in decimi di carattere

; Questo esempio traccia una linea verticale sottile dal punto (0100,0000) al punto (0100,0220)

L00 0100 0000 0100 0220

; la linea che segue è orizzontale, parte da (0100,0220) e arriva a (0750,0220)

L00 0100 0220 0750 0220

Disegno di un riquadro

La riga di comando per disegnare un riquadro (rettangolo colorato) ha la seguente sintassi:

Rpp xxxx yyyy XXXX YYYY
dove:

* **R** in prima posizione individua il tipo di azione da intraprendere (disegno di un riquadro)
* **pp** è un numero espresso con due cifre e compreso fra **00** e **15** che rappresenta l'intensità della colorazione del riquadro (00 è bianco, 01 grigio tenue e così via)
* **xxxx** e **yyyy** rappresentano le coordinate del vertice sinistro alto del riquadro
* **XXXX** e **YYYY** rappresentano le coordinate del vertice destro basso del riquadro.

; Questo esempio disegna un riquadro grigio tenue da posizione (0080,0100) a posizione (0800,0110)

R01 0080 0100 0800 0110

Disegno di una didascalia

La riga di comando per disegnare una didascalia (riga di testo) ha la seguente sintassi:

Tff xxxx yyyy <stringa di caratteri>
dove:

* **T** in prima posizione individua il tipo di azione da intraprendere (disegno di una didascalia)
* **ff** è un numero espresso con due cifre e compreso fra **00** e **30** che rappresenta la dimensione del font da usare per disegnare la didascalia (i font più piccoli sono individuati da numeri più piccoli)
* **xxxx** e **yyyy** rappresentano le coordinate del punto di inizio della didascalia
* **<stringa di caratteri>** rappresenta il testo della didascalia

N.B.: l'applicazione sceglie il tipo di font in funzione della dimensione prescelta fra l'elenco dei font disponibili sul sistema. Se non esiste un font con le caratteristiche necessarie viene utilizzato quello che ha le caratteristiche dimensionali più vicine al font desiderato.

; Questo esempio inserisce varie didascalie utilizzando font di dimensione diversa

T03 0020 0204 CODICE ARTICOLO

T04 0240 0204 DESCRIZIONE

Disegno di una bitmap

La riga di comando per disegnare una bitmap ha la seguente sintassi

B xxxx yyyy XXXX YYYY <nome file>
dove:

* **B** in prima posizione individua il tipo di azione da intraprendere (disegno di una bitmap)
* **xxxx** e **yyyy** rappresentano le coordinate del vertice sinistro alto della bitmap
* **XXXX** e **YYYY** rappresentano le coordinate del vertice destro basso della bitmap

* **<nome file>** è il nome del file contenente la bitmap privo del path[15]

Le bitmap devono essere in formato BMP Windows **per qualunque piattaforma operativa** (anche per OS/2) **e devono essere memorizzate nella directory di lavoro di SIGLA.**

La bitmap da disegnare viene ridimensionata in modo da riempire il rettangolo specificato.

; Questo esempio disegna una bitmap nella parte superiore sinistra della pagina

B 0010 0010 0300 0100 logo.bmp

Esempi

L'esempio che segue, anche se non ha una particolare utilità all'interno di SIGLA, dimostra la versatilità del meccanismo di gestione dei fincati adottato.

Il listato riportato genera la fincatura per un modello INPS DM10/2-89 (versamento contributi previdenziali) che è stato scelto come esempio per la sua complessità.

```
; DM/10
;
L00 0000 0000 0800 0000
L00 0000 0000 0000 0670
L00 0800 0000 0800 0670
L00 0000 0670 0800 0670
;
R01 0090 0100 0800 0110
R01 0090 0120 0450 0130
R01 0690 0560 0800 0580
;
L00 0000 0030 0660 0030
L00 0000 0140 0800 0140
L00 0090 0030 0090 0140
L00 0660 0000 0660 0100
L00 0090 0100 0800 0100
L00 0090 0110 0800 0110
L00 0090 0120 0800 0120
L00 0090 0130 0450 0130
L00 0230 0100 0230 0120
L00 0290 0100 0290 0120
L00 0490 0100 0490 0120
L00 0630 0100 0630 0120
L00 0170 0120 0170 0140
L00 0270 0120 0270 0140
L00 0410 0120 0410 0140
L00 0450 0120 0450 0140
L00 0620 0120 0620 0140
L00 0450 0120 0620 0140
L00 0450 0140 0620 0120
L00 0620 0120 0800 0140
L00 0620 0140 0800 0120
L00 0335 0000 0335 0030
;
L00 0000 0160 0800 0160
L00 0030 0140 0030 0160
L00 0420 0140 0420 0160
L00 0450 0140 0450 0160
L00 0530 0140 0530 0160
L00 0560 0140 0560 0160
L00 0030 0150 0420 0150
;
L00 0000 0170 0800 0170
L00 0000 0180 0800 0180
L00 0000 0190 0800 0190
L00 0000 0200 0800 0200
L00 0000 0210 0800 0210
L00 0000 0220 0800 0220
L00 0000 0230 0800 0230
L00 0000 0240 0800 0240
L00 0000 0250 0800 0250
L00 0000 0260 0800 0260
L00 0000 0270 0800 0270
L00 0000 0280 0800 0280
L00 0000 0290 0800 0290
L00 0000 0300 0800 0300
L00 0000 0310 0800 0310
L00 0000 0320 0800 0320
L00 0000 0330 0800 0330
L00 0000 0340 0800 0340
L00 0000 0350 0800 0350
L00 0000 0360 0800 0360
L00 0000 0370 0800 0370
L00 0000 0380 0800 0380
L00 0000 0390 0800 0390
L00 0000 0400 0800 0400
L00 0000 0410 0800 0410
```

[15] Il nome del file bitmap deve essere valido, ovvero non può contenere caratteri non ammessi dal sistema operativo per i nomi file, deve comprendere l'estensione (bmp) e non devono essere presenti ulteriori caratteri dopo l'estensione stessa.

```
L00 0000 0420 0800 0420
L00 0000 0430 0800 0430
L00 0000 0440 0800 0440
L00 0000 0450 0800 0450
L00 0000 0460 0800 0460
L00 0000 0470 0800 0470
L00 0000 0480 0800 0480
L00 0000 0490 0800 0490
L00 0000 0500 0800 0500
L00 0000 0510 0800 0510
L00 0000 0520 0800 0520
L00 0000 0530 0800 0530
L00 0000 0540 0800 0540
L00 0000 0550 0800 0550
L00 0000 0560 0800 0560
L00 0000 0570 0800 0570
L00 0000 0580 0800 0580
;
L00 0090 0150 0090 0580
L00 0100 0150 0100 0580
L00 0150 0150 0150 0580
L00 0160 0150 0160 0570
L00 0220 0150 0220 0570
L00 0230 0150 0230 0570
L00 0300 0150 0300 0570
L00 0310 0150 0310 0570
L00 0420 0150 0420 0670
L00 0530 0150 0530 0580
L00 0620 0160 0620 0550
L00 0630 0160 0630 0560
L00 0680 0160 0680 0640
L00 0690 0160 0690 0580
;
L00 0230 0550 0300 0560
L00 0230 0560 0300 0550
L00 0310 0550 0420 0560
L00 0310 0560 0420 0550
L00 0420 0560 0530 0570
L00 0420 0570 0530 0560
L00 0150 0570 0420 0580
L00 0150 0580 0420 0570
;
L00 0030 0590 0420 0590
L00 0000 0600 0030 0600
L00 0030 0580 0030 0600
L00 0420 0600 0800 0600
L00 0450 0580 0450 0600
L00 0450 0590 0800 0590
L00 0420 0610 0800 0610
L00 0420 0620 0800 0620
L00 0420 0630 0800 0630
L00 0420 0640 0800 0640
L00 0420 0650 0800 0650
L00 0450 0650 0450 0670
L00 0450 0660 0800 0660
L00 0570 0580 0570 0640
L00 0580 0600 0580 0650
L00 0690 0600 0690 0640
L00 0790 0600 0790 0650
L00 0480 0660 0480 0670
L00 0670 0660 0670 0670
L00 0440 0600 0440 0650
;
L00 0000 0050 0030 0050
L00 0030 0030 0030 0050
;
T13 0003 0003 ISTITUTO NAZIONALE
T13 0003 0023 DELLA PREVIDENZA SOCIALE
T07 0210 0005 Mod. DM 10/2-89
T07 0210 0024 ORIGINALE PER L'INPS
T08 0340 0008 Denuncia per il versamento dei contributi e per la richiesta di
T08 0340 0018 rimborso dell'eventuale differenza a credito del datore di lavoro.
T10 0680 0003 Codice Guida
T04 0135 0104 MATRICOLA
T04 0250 0104 CSC
T04 0330 0104 CODICI AUTORIZZAZIONI
T04 0540 0104 SEDE DI
T04 0645 0104 CODICE FISCALE/PARTITA IVA
T03 0113 0121 PERIODO
T03 0103 0126 mese
T03 0140 0126 anno
T04 0175 0124 DATA ESECUTIVITA'
T04 0290 0124 N. DIPEND. OCCUPATI
T03 0413 0124 TIPO DEN.
T04 0185 0144 Dati retributivi
T03 0460 0146 Somme a debito
T03 0456 0151 del datore di lavoro
T04 0590 0149 Somme a credito del datore di lavoro
T18 0010 0038 A
T18 0010 0148 B
T18 0430 0148 C
T18 0540 0148 D
T18 0010 0588 I
T18 0430 0588 F
T18 0430 0658 H
T02 0120 0154 Cod.
T02 0168 0154 N. dipendenti
T02 0248 0154 N. giornate
T02 0358 0154 Retribuzioni
T02 0003 0164 Operai
T02 0533 0162 Assegni correnti
T02 0533 0166 per nucleo familiare
T02 0003 0174 Impiegati
T02 0533 0174 Fiscalizzazione
T02 0533 0184 Fiscalizzazione part time
```

```
T02 0533 0194 CIG ordinaria
T02 0533 0204 CIG straordinaria
T02 0533 0212 Sgravio L. 1089/68
T02 0533 0216 L. 589/71 - L. 463/72
T02 0533 0224 Sgravio L. 183/76
T02 0533 0234 Indennita' malattia
T02 0533 0242 Indennita' maternita'
T02 0533 0246 obbligatoria
T02 0533 0252 Indennita' malattia
T02 0533 0256 L. 88/87
T02 0003 0522 Apprendisti
T02 0003 0526 soggetti INAIL
T02 0003 0532 Apprendisti non
T02 0003 0536 soggetti INAIL
T02 0003 0542 Lavoro
T02 0003 0546 straordinario
T02 0003 0552 Trattenute a
T02 0003 0556 pensionati AGO
T02 0233 0522 N. settimane
T02 0233 0532 N. settimane
T02 0233 0542 N. ore
T01 0533 0562 Differenza a debito da versare
T01 0533 0566 dal datore di lavoro
T01 0533 0572 Differenza a credito di cui si
T01 0533 0576 chiede il rimborso dal dat. di lav.
T04 0135 0584 Dichiarazione del datore di lavoro
T03 0040 0594 Il sottoscritto dichiara che:
T03 0010 0604 - I dati esposti nella presente denuncia sono conformi alle registrazioni effettuate sui libri paga e
T03 0010 0609   matricola;
T03 0010 0619 - Le somme esposte a conguaglio sono state determinate in conformita' alle vigenti disposizioni di
T03 0010 0624   legge;
T03 0010 0634 - E' a conoscenza ed accetta che il pagamento della eventuale differenza a proprio credito, chiesta in
T03 0010 0639   rimborso, verra' eseguito, ove nulla osti, a titolo provvisorio sulla base dei dati forniti e con riserva da
T03 0010 0644   parte dell'I.N.P.S. di verificare la posizione contributiva aziendale in sede di acquisizione dei dati dalle
T03 0010 0649   corrispondenti denunce individuali.
;
T04 0465 0584 Autorizzazione CIG
T04 0590 0584 Interv. ordinari
T04 0690 0584 Interventi straordinari
T02 0495 0594 Numero
T02 0605 0594 Numero ore
T02 0720 0594 Numero ore
T02 0442 0642 Quote indennita' anzianita'
T02 0582 0642 Numero dipendenti
T02 0482 0664 Residui contributi altre dipendenze
T04 0530 0654 Sgravio onderi sociali per il Mezzogiorno
;
T05 0400 0164 .000
T05 0400 0174 .000
T05 0400 0184 .000
T05 0400 0194 .000
T05 0400 0204 .000
T05 0400 0214 .000
T05 0400 0224 .000
T05 0400 0234 .000
T05 0400 0244 .000
T05 0400 0254 .000
T05 0400 0264 .000
T05 0400 0274 .000
T05 0400 0284 .000
T05 0400 0294 .000
T05 0400 0304 .000
T05 0400 0314 .000
T05 0400 0324 .000
T05 0400 0334 .000
T05 0400 0344 .000
T05 0400 0354 .000
T05 0400 0364 .000
T05 0400 0374 .000
T05 0400 0384 .000
T05 0400 0394 .000
T05 0400 0404 .000
T05 0400 0414 .000
T05 0400 0424 .000
T05 0400 0434 .000
T05 0400 0444 .000
T05 0400 0454 .000
T05 0400 0464 .000
T05 0400 0474 .000
T05 0400 0484 .000
T05 0400 0494 .000
T05 0400 0504 .000
T05 0400 0514 .000
T05 0400 0524 .000
T05 0400 0534 .000
T05 0400 0544 .000
T05 0400 0564 .000
;
T05 0510 0164 .000
T05 0510 0174 .000
T05 0510 0184 .000
T05 0510 0194 .000
T05 0510 0204 .000
T05 0510 0214 .000
T05 0510 0224 .000
T05 0510 0234 .000
T05 0510 0244 .000
T05 0510 0254 .000
T05 0510 0264 .000
T05 0510 0274 .000
T05 0510 0284 .000
T05 0510 0294 .000
T05 0510 0304 .000
T05 0510 0314 .000
T05 0510 0324 .000
```

```
T05 0510 0334 .000
T05 0510 0344 .000
T05 0510 0354 .000
T05 0510 0364 .000
T05 0510 0374 .000
T05 0510 0384 .000
T05 0510 0394 .000
T05 0510 0404 .000
T05 0510 0414 .000
T05 0510 0424 .000
T05 0510 0434 .000
T05 0510 0444 .000
T05 0510 0454 .000
T05 0510 0464 .000
T05 0510 0474 .000
T05 0510 0484 .000
T05 0510 0494 .000
T05 0510 0504 .000
T05 0510 0514 .000
T05 0510 0524 .000
T05 0510 0534 .000
T05 0510 0544 .000
T05 0510 0554 .000
T05 0510 0574 .000
;
T05 0780 0164 .000
T05 0780 0174 .000
T05 0780 0184 .000
T05 0780 0194 .000
T05 0780 0204 .000
T05 0780 0214 .000
T05 0780 0224 .000
T05 0780 0234 .000
T05 0780 0244 .000
T05 0780 0254 .000
T05 0780 0264 .000
T05 0780 0274 .000
T05 0780 0284 .000
T05 0780 0294 .000
T05 0780 0304 .000
T05 0780 0314 .000
T05 0780 0324 .000
T05 0780 0334 .000
T05 0780 0344 .000
T05 0780 0354 .000
T05 0780 0364 .000
T05 0780 0374 .000
T05 0780 0384 .000
T05 0780 0394 .000
T05 0780 0404 .000
T05 0780 0414 .000
T05 0780 0424 .000
T05 0780 0434 .000
T05 0780 0444 .000
T05 0780 0454 .000
T05 0780 0464 .000
T05 0780 0474 .000
T05 0780 0484 .000
T05 0780 0494 .000
T05 0780 0504 .000
T05 0780 0514 .000
T05 0780 0524 .000
T05 0780 0534 .000
T05 0780 0544 .000
T05 0780 0554 .000
T05 0780 0564 .000
T05 0780 0574 .000
T05 0780 0664 .000
;
T12 0120 0162 10
T12 0120 0172 11
T12 0120 0522 20
T12 0120 0532 21
T12 0120 0542 22
T12 0120 0552 23
T12 0002 0562 TOTALI
T12 0120 0562 24
T12 0002 0572 TOTALE A
T12 0120 0572 33
T12 0650 0162 35
T12 0650 0172 37
T12 0650 0182 38
T12 0650 0192 39
T12 0650 0202 40
T12 0650 0212 45
T12 0650 0222 49
T12 0650 0232 52
T12 0650 0242 53
T12 0650 0252 54
T12 0532 0552 TOTALE B
T12 0650 0552 57
T12 0620 0562 TOT A-B
T12 0620 0572 TOT B-A
T12 0422 0602 64
T12 0422 0612 65
T12 0422 0622 66
T12 0422 0632 67
T12 0422 0642 68
T12 0462 0662 79
B 0002 0002 0072 0052 inps.bmp
```

Definizione delle classi C++

Con il sistema di sviluppo di SIGLA sono distribuiti gli header files che definiscono alcune della classi C++ usate dal pacchetto. In questo paragrafo vengono documentate alcune delle classi fondamentali.

DPDbase

La classe DPDbase è usata dal pacchetto per gestire l'accesso ai dati via ODBC o CLI (versione SIGLA) o tramite l'uso delle API della libreria Codebase (versione START) ed è definita nell'header file DPDbase.h.

Possono essere d'interesse, al fine dello sviluppo di personalizzazioni via SIGPPDLL.DLL, alcuni dei dati membro della classe che vengono di seguito documentati:

Versione ODBC

* *HDBC hdbc* handle della connessione ODBC/CLI utilizzata da SIGLA[16]
* *HDBC hdbc1,hdbc2* handles di ulteriori connessioni usate **solo verso SQL Server Microsoft**
* *HDBC hdbc3* handle della connessione usata per gestire le operazioni di INSERT/UPDATE/DELETE **solo verso SQL Server Microsoft**
* *HDBC hdbc4,hdbc5* attualmente non utilizzati e contenenti valori non predicibili
* *BOOL IsConnected* contiene il valore TRUE se è stata eseguita la connessione al database, FALSE altrimenti
* *BOOL IsCommittable* contiene il valore TRUE se il DBMS supporta le transazioni, FALSE altrimenti
* *BOOL IsTransactionPending* contiene il valore TRUE se una transazione non è stata ancora committata, FALSE altrimenti
* *short int sql_active_statements* numero di cursori che possono essere contemporaneamente aperti su una singola connessione. Questo dato vale 1 per SQL Server , 0 per tutti gli altri DBMS certificati
* *char DPCHAR[128]* array di caratteri null-terminated contenente il tipo dato nativo usato dal DBMS per individuare i campi carattere
* *char DPNUM[128]* array di caratteri null-terminated contenente il tipo dato nativo usato dal DBMS per individuare i campi numerici
* *BOOL TransactionOn* contiene l valore TRUE se è stata impostata la gestione delle transazioni, FALSE altrimenti
* *BOOL NeedOrderBy* contiene il valore TRUE se è stata impostata la clausola Order By in configurazione dell'applicazione, FALSE altrimenti
* *BOOL CanPrepareStatement* contiene il valore TRUE se possono essere usati statements SQL preparati, FALSE altrimenti
* *BOOL IsEBCDIC* contiene il valore TRUE se è stato impostato l'ordinamento EBCDIC in configurazione dell'applicazione
* *char DbmsType[128]* array di caratteri null-terminated contenente il nome del DBMS

Versione Start Suite

* *char databasepath[128]* array di caratteri null-terminated contenente la path dell'indirizzario dove riesiedono le tabelle della base di dati

DPObject

Dalla classe astratta DPObject derivano tutti gli oggetti che SIGLA utilizza per accedere alle varie tabelle che compongono il database del prodotto. Per questo motivo alle funzioni chiamate nella libreria SIGPPDLL.DLL (vedi punti successivi) vengono passati come parametri puntatori a oggetti di tipo DPObject che, per essere utilizzati in modo corretto, devono essere castati sulla classe effettiva documentata nel corpo della presente documentazione.

[16] Poiché il manager ODBC alloca memoria in modo non condivisibile le connessioni allocate da SIGLA **NON** possono essere utilizzate all'esterno dell'applicazione stessa e delle sue DLL (compresa SIGPPDLL.DLL).

Possono essere d'interesse, al fine dello sviluppo di personalizzazioni via SIGPPDLL.DLL, alcuni dei dati membro della classe che vengono di seguito documentati:

* *DPDbase *theDB* puntatore all'oggetto di classe DPDbase utilizzato per gestire l'accesso al database
* *char Esercizio[5]* array di carattery null-terminated contenente il codice dell'esercizio per il quale l'oggetto è stato creato (il dato membro è significativo solo per oggetti che mappano tabelle dipendenti dal codice esercizio)
* *BOOL bBOF* contiene il valore TRUE se per nessuna riga del cursore associato all'oggetto è ancora stato eseguito un fetch
* *BOOL bEOF* contiene il valore TRUE se per l'ultima riga del cursore associato all'oggetto è stato eseguito un fetch

Personalizzazione dell'applicazione

L'architettura di SIGLA ed in particolare l'apertura della sua base di dati (gestita dal pacchetto via ODBC o via CLI) permette il facile sviluppo di applicazioni esterne in grado di operare sulla stessa base di dati del pacchetto.

Con questa logica è possibile la realizzazione di procedure che appaiano all'utilizzatore come 'icone' separate rispetto a SIGLA. In molti casi, tuttavia, è auspicabile raggiungere un maggior grado di integrazione fra l'applicazione principale e le sue customizzazioni. Per consentire di raggiungere questo risultato il pacchetto esegue, in punti prestabiliti, delle chiamate a funzioni contenute in una libreria a link dinamico (**SIGPPDLL.DLL**) della quale viene rilasciato il sorgente (relativamente agli ambienti Window16/Windows32). Le chiamate alle funzioni esterne vengono effettuate sia dall'applicazione SIGLA che dal programma di configurazione.

N.B.: i sorgenti della libreria SIGPPDLL.DLL non sono disponibili per le versioni OS/2 di SIGLA.

Modificando opportunamente e ricompilando la libreria SIGPPDLL.DLL è possibile intervenire sul comportamento standard dell'applicazione. Il paragrafo successivo spiega i principi di funzionamento della DLL. In particolare è possibile, **utilizzando strumenti adeguati come Borland Delphi o altri linguaggi in grado di generare librerie a link dinamico**, realizzare delle funzioni aggiuntive o sostitutive di quelle di SIGLA e agganciare tali funzioni al pacchetto. Le funzioni aggiuntive devono essere contenute in una libreria a link dinamico perché si possa garantire la perfetta sincronicità fra il programma chiamante (SIGLA) e il programma chiamato (customizzazione).

Per agevolare lo sviluppo di applicazioni esterne le principali funzioni di basso livello di SIGLA (registrazione movimenti contabili, registrazione movimenti di magazzino etc.) sono rese disponibili all'interno della libreria a link dinamico **SPPSERV.DLL**. Le specifiche funzionali di questa libreria sono dettagliate successivamente.

N.B.: la libreria SPPSERV.DLL non è disponibile per la versione OS/2 di SIGLA.

La libreria a link dinamico SIGPPDLL.DLL

La libreria è stata realizzata in C++ e compilata utilizzando Microsoft Visual C++ (versione 1.51 per l'ambiente a 16 bit, versione 4.0 per quello a 32 bit). Il sorgente della procedura è contenuto nel file SIGPPDLL.CPP e utilizza gli header files SIGPPDLL.H e DPDBASE.H.

Per modificare e ricompilare la libreria e' necessario disporre delle seguenti componenti software:

- ***Compilatore C++ a 16/32 bit in funzione della versione da ricompilare***
- ***ODBC SDK 2.01 (contenuto nei CD d'installazione dei compilatori Microsoft o acquistabile separatamente) per la versione Client/Server***
- ***libreria CodeBase 6.x per la versione START++ (la libreria in questione e' necessaria per gli header files che sono indispensabili per la ricompilazione)***

Ricompilazione della libreria

Le seguenti istruzioni sono valide per il compilatore Microsoft Visual C++ 1.51. La ricompilazione nell'ambiente a 32 bit può essere effettuata usando il compilatore adeguato in modo analogo a quello descritto.

* Avviare l'IDE del compilatore, selezionare la voce di menù <Project> e successivamente <New>
* Impostare come Project Name SIGPPDLL eventualmente preceduto dalla path della directory contenente il sorgente.
* Impostare come Project Type <Window dynamic-link library>
* Decheckare il controllo <Use Microsoft Foundation Classes>
* Premere il push button <OK>

* Inserire nel progetto i files SIGPPDLL.CPP e SIGPPDLL.DEF[17]
* Inserire nel progetto la libreria ODBC32.LIB
* Impostare l'allineamento delle strutture a 1 byte (Options/Project/Compiler/Code Generation)
* Compilare l'applicazione

N.B.: durante la ricompilazione devono essere disponibili al compilatore tutti gli header files aggiuntivi eventualmente utilizzati (vedi punti successivi).

Per la ricompilazione della versione a 32 bit deve essere usato il compilatore Micriosoft 4.0 o successivi con i seguenti accorgimenti:

* includere l'header file DPDBASE.H specifico per la versione a 32 bit
* definire la macro DPODBC
* impostare l'allineamento delle strutture a 8 bit
* compilare per una esecuzione single-threaded

Per la versione Start Suite è necessario disporre dell'apposito header DPDBASE.H. E' necessario inoltre definire la macro DPCODEBASE.

ATTENZIONE: se nella realizzazione di alcune operazioni attraverso questa DLL vengono inclusi alcuni header files delle librerie di oggetti di SIGLA è necessario utilizzare sempre i file di inclusione corrispondenti alla versione di SIGLA che si sta utilizzando. Per questo motivo è chiaro che ad ogni rilascio di SIGLA è necessario aggiornare i file di include e ricompilare la DLL con i nuovi files.

Funzioni contenute in SIGPPDLL.DLL

Di seguito vengono elencate le funzioni contenute nella libreria a link dinamico. La macro DLLCALL è definita nel file SIGPPDLL.H in modo diverso in funzione della piattaforma di compilazione e serve ad evidenziare le funzioni esportabili. Per la documentazione delle API di Windows utilizzate negli esempi si rimanda alla consultazione dei manuali dell'SDK di Windows.

LibMain

SCOPO: viene chiamata da LibEntry. LibEntry è chiamata da Windows quando la DLL viene caricata. LibEntry inizializza l'heap della DLL se è stato specificato un valore per il parametro HEAPSIZE nel file DEF della DLL. Successivamente viene chiamata la funzione LibMain che deve eseguire le inizializzazioni necessarie per la DLL. LibMain deve tornare 1 se l'inizializzazione ha avuto successo. Se il valore tornato è 0 la DLL viene scaricata.

La sintassi della funzione, che è presente solo nella versione a 16 bit della libreria, è la seguente:

int FAR PASCAL LibMain(HANDLE hModule**,WORD** wDataSeg**,WORD** cbHeapSize**,LPSTR** lpszCmdLine**)**

N.B.: è necessario modificare questa funzione per personalizzare SIGLA.

WEP

SCOPO: esegue le azioni di cleanup quando la DLL viene scaricata. WEP() viene chiamata automaticamente da Windows quando la DLL viene scaricata (nessuna task continua ad usarla).

La sintassi della funzione, che è presente solo nella versione a 16 bit della libreria, è la seguente:

int CALLBACK WEP(int bSystemExit**)**

N.B.: è necessario modificare questa funzione per personalizzare SIGLA.

[17] Il file SIGPPDLL.DEF è necessario solo per la costruzione della libreria nella versione a 16 bit. La struttura consigliata per tale file è la seguente:

```
LIBRARY   SIGPPDLL
EXETYPE   WINDOWS
CODE      PRELOAD MOVEABLE DISCARDABLE
DATA      PRELOAD MOVEABLE SINGLE
HEAPSIZE  1024
EXPORTS
          WEP PRIVATE
```

SIGLAPPStart

SCOPO: viene chiamata da SIGLA immediatamente dopo che la procedura è stata avviata.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPStart(void)

Così come viene distribuita la funzione non effettua alcuna operazione. E' possibile utilizzare SIGLAPPStart() per compiere le azioni di inizializzazione necessarie. Ad esempio, se sono state sviluppate delle personalizzazioni contenute nella libreria a link dinamico CUSTOM.DLL, è possibile caricare tale DLL modificando SIGLAPPStart() nel seguente modo:

- definire un handle globale di tipo HINSTANCE
- caricare CUSTOM.DLL utilizzando la funzione LoadLibrary dell' SDK di Windows

Il sorgente modificato potrebbe avere la seguente forma:

```
HINSTANCE customdll;
void DLLCALL SIGLAPPStart(void)
{
        customdll=LoadLibrary("CUSTOM.DLL");
}
```

L'handle *customdll* dovrà essere utilizzato in seguito per lanciare le funzioni contenute nella libreria CUSTOM.DLL e per scaricare la DLL dalla memoria quando SIGLA viene arrestato (vedi SIGLAPPStop()).

SIGLAPPStop

SCOPO: viene chiamata da SIGLA immediatamente prima che l'applicazione venga terminata.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPStop(void)

Così come viene distribuita la funzione non effettua alcuna operazione. E' possibile utilizzare SIGLAPPStop() per compiere le azioni di terminazione necessarie. Ad esempio, se si sono state sviluppate delle personalizzazioni contenute nella libreria a link dinamico CUSTOM.DLL (caricata in memoria attraverso una modifica a SIGLAPPStart()) , è possibile scaricare tale DLL modificando SIGLAPPStop() nel seguente modo:

- scaricare la libreria CUSTOM.DLL utilizzando la funzione FreeLibrary dell'SDK di Windows

Il sorgente modificato potrebbe avere la seguente forma:

```
void DLLCALL SIGLAPPStop(void)
{
        extern HINSTANCE customdll;
        FreeLibrary(customdll);
}
```

SIGLAPPCreateComuni

SCOPO: viene chiamata dal programma di configurazione di SIGLA immediatamente dopo che si è proceduto alla creazione delle tabelle nel database dati comuni.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPCreateComuni(void)

Così come viene distribuita la funzione non effettua alcuna operazione. E' possibile utilizzare SIGLAPPCreateComuni() per costruire ulteriori tabelle nel database dati comuni rispetto a quelle previste da SIGLA o per aggiungere campi alle tabelle standard. Ad esempio è possibile sviluppare una funzione custom CREATABELLECOMUNI (contenuta in CUSTOM.DLL) e chiamarla da SIGLAPPCreateComuni nel seguente modo:

```
void DLLCALL SIGLAPPCreateComuni(void)
{
        extern HINSTANCE customdll;
        void  (FAR PASCAL *customfunct) (void);
        FARPROC pp;

        if(customdll>HINSTANCE_ERROR) {
                pp=GetProcAddress(customdll,"CREATABELLECOMUNI");
                if(pp!=NULL) {
                        customfunct=(void (FAR PASCAL *)()) pp;
                        (*customfunct)();
                }
        }
}
```

SIGLAPPCreateDitta

SCOPO: viene chiamata dal programma di configurazione di SIGLA immediatamente dopo che si è proceduto alla creazione delle tabelle nel database di una ditta.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPCreateDitta(void)

Così come viene distribuita la funzione non effettua alcuna operazione. E' possibile utilizzare SIGLAPPCreateDitta() per costruire ulteriori tabelle nel database delle aziende rispetto a quelle previste da SIGLA o per aggiungere campi alle tabelle standard. Ad esempio è possibile sviluppare una funzione custom CREATABELLEDITTA (contenuta in CUSTOM.DLL) e chiamarla da SIGLAPPCreateDitta nel seguente modo:

```
void DLLCALL SIGLAPPCreateDitta(void)
{
        extern HINSTANCE customdll;
        void  (FAR PASCAL *customfunct) (void);
        FARPROC pp;

        if(customdll>HINSTANCE_ERROR) {
                pp=GetProcAddress(customdll,"CREATABELLEDITTA");
                if(pp!=NULL) {
                        customfunct=(void (FAR PASCAL *)()) pp;
                        (*customfunct)();
                }
        }
}
```

SIGLAPPInit

SCOPO: viene chiamata da SIGLA immediatamente dopo che l'utente ha scelto la ditta su cui intende operare.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPInit(LPSTR utente**,LPSTR** todaydate**,LPSTR** codditta**,LPSTR** ragsoc**,LPSTR** termname**,LPSTR** dittaconnectstring**,LPSTR** comuniconnectstring**,DPDbase ***ditta**,DPDbase ***comuni**)**

I parametri della funzione hanno il seguente significato:

* *utente* puntatore ad un array di 8 caratteri null-terminated contente il codice dell'utente con cui l'operatore si è dichiarato a SIGLA. Se la gestione utenti non è attiva il dato è comunque impostato sul valore di default "SIGLA ".**Se la chiamata viene effettuata dalla procedura di configurazione viene passato in ogni caso il valore costante "CONFIGUR".**
* *todaydate* puntatore ad un array di 8 caratteri null-terminated contente la data di sistema in formato ANSI (AAAAMMGG)
* *codditta* puntatore ad un array di 5 caratteri null-terminated contente il codice della ditta selezionata
* *ragsoc* puntatore ad un array di 50 caratteri null-terminated contente la ragione sociale della ditta selezionata
* *termname* puntatore ad un array di 8 caratteri null-terminated contente il nome del terminale ricavato dalla variabile di environment SIGLAPP
* *dittaconnectstring* puntatore ad un array di caratteri null-terminated contente la stringa di connessione ODBC da usare per collegarsi al database della ditta selezionata
* *comuniconnectstring* puntatore ad un array di caratteri null-terminated contente la stringa di connessione ODBC da usare per collegarsi al database dei dati comuni (la stringa non contiene mai la password d'accesso al database)
* *ditta* puntatore all' oggetto di tipo DPDbase che SIGLA usa per gestire l'attività sul database della ditta selezionata
* *comuni* puntatore all' oggetto di tipo DPDbase che SIGLA usa per gestire l'attività sul database dei dati comuni

L'header file che definisce la classe DPDbase è fornito assieme al sorgente della libreria SIGPPDLL.DLL.

Così come viene distribuita la funzione non effettua alcuna operazione. E' possibile utilizzare SIGLAPPInit() per far si che il pacchetto di personalizzazioni sia messo in condizione di operare sul database dell'azienda prescelta dall'utente. Ad esempio è possibile prevedere una funzione custom di avvio delle connessioni sui database necessarie al pacchetto personalizzato per svolgere le sue attività. Tale funzione avra' probabilmente necessita' di conoscere le stringhe di connessione ODBC verso i database che potranno quindi esserle passate come parametri:

```
void DLLCALL SIGLAPPInit(LPSTR utente, LPSTR todaydate, LPSTR codditta, LPSTR ragsoc, LPSTR termname
                         LPSTR dittaconnectstring,
                         LPSTR comuniconnectstring,
                         DPDbase *ditta,
                         DPDbase *comuni)
{
        extern HINSTANCE customdll;
        void  (FAR PASCAL *customconnect) (LPSTR,LPSTR);
        FARPROC pp;
        if(customdll>HINSTANCE_ERROR) {
                pp=GetProcAddress(customdll,"STARTCONNECT");
                if(pp!=NULL) {
                        customconnect=(void (FAR PASCAL *)(LPSTR,LPSTR)) pp;
                        (*customconnect)(comuniconnectstring,dittaconnectstring);
                }
        }
}
```

SIGLAPPSetEsercizio

SCOPO: viene chiamata da SIGLA immediatamente dopo che l'utente ha scelto l'esercizio contabile su cui operare.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPSetEsercizio(LPSTR esercizio**)**

I parametri della funzione hanno il seguente significato:

- *exercizio* puntatore ad un array di 4 caratteri null-terminated contenente il codice dell'esercizio contabile selezionato

Così come viene distribuita la funzione non effettua alcuna operazione. E' possibile utilizzare SIGLAPPSetEsercizio() per far si che il pacchetto di personalizzazioni sia messo in condizione di operare sull' esercizio prescelto dall'utente. Ad esempio è possibile prevedere una funzione custom di impostazione dell'esercizio per il pacchetto personalizzato. Tale funzione dovrà accettare come parametro il codice dell'esercizio prescelto:

```
void DLLCALL SIGLAPPSetEsercizio(LPSTR esercizio)
{
        extern HINSTANCE customdll;
        void  (FAR PASCAL *customset esert) (LPSTR);
        FARPROC pp;
        if(customdll>HINSTANCE_ERROR) {
                pp=GetProcAddress(customdll,"SETESER");
                if(pp!=NULL) {
                        customseteser=(void (FAR PASCAL *)(LPSTR)) pp;
                        (*customseteser)(esercizio);
                }
        }
}
```

SIGLAPPDoMenuCommand

SCOPO: viene chiamata da SIGLA immediatamente dopo la selezione di una voce di menu da parte dell'utente e prima che il pacchetto attivi la funzione richiesta. La funzione DEVE tornare TRUE se SIGLA

deve processare l'evento, FALSE altrimenti. Il parametro theMenuID identifica la voce di menù interessata. Le macro SPPMENU_* sono definite nel file SIGPPDLL.H e mappano alcuni valori possibili.

La sintassi della funzione è la seguente:

void DLLCALL SIGLAPPDoMenuCommand(int theMenuID**)**

I valori possibili per il parametro theMenuID sono mappati nella seguente tabella:

Valore	Voce di menù
1001	**Apri Ditta**
1002	**Chiudi Ditta**
1003	**Informazioni**
1004	**Tabella Raggruppamenti Magazzini**
1005	**Tabella Magazzini**
1006	**Tabella Raggruppamenti Fiscali**
1007	**Tabella Imballi**
1008	**Tabella Gruppi Merceologici**
1009	**Tabella Famiglie Merceologiche**
1010	**Tabella Sottofamiglie Merceologiche**
1011	**Tabella Ubicazioni**
1012	**Tabella Taglie**
1013	**Tabella Sconti**
1014	**Tabella Raggruppamento Causali di Magazzino**
1015	**Tabella Offerte**
1016	**Tabella Causali di Magazzino**
1017	**Tabella Listini**
1018	**Tabella Lingue Estere**
1019	**Tabella Tipi Documento di Magazzino**
1020	**Tabella Classificazione Doganale**
1021	**Tabella Numeratori**
1022	**Tabella Sportelli Bancari**
1023	**Tabella Cariche Sociali**
1024	**Tabella Codici Attività**
1025	**Tabella Comuni d'Italia**
1026	**Tabella Codici Statistici**
1027	**Tabella Stati Esteri**
1028	**Tabella Aliquote Iva**
1029	**Tabella Uffici Concessionari**
1030	**Tabella Centri Servizio**
1031	**Tabella Uffici Iva**
1032	**Tabella Uffici del Registro**
1033	**Tabella Uffici Imposte Dirette**
1034	**Tabella Libri Sociali**

1035	**Codici Servizi**
1500	**Aiuto/Usare la guida**
1501	**Auto/Sommario**
2001	**Imposta Stampante**
2002	**Imposta Modo di Stampa**
2003	**Stampe Bloccate**
3001	**Anagrafica di Magazzino**
3002	**Anagrafica Listini**
3003	**Anagrafica Codici a Barre**
3004	**Anagrafica Descrizioni in Lingua Articoli**
3005	**Anagrafica Codici Articoli Clienti/Fornitori**
3006	**Condizioni Particolari di Acquisto/Vendita**
3007	**Tabella Sconti**
3008	**Variazione Aliquote Iva su Anagrafica di Magazzino**
3102	**Archivio LIFO/FIFO fiscale**
3103	**Aggiornamento Automatico LIFO/FIFO**
3201	**Immissione Movimenti di Magazzino**
3202	**Revisione Movimenti di Magazzino**
3203	**Apertura Magazzino**
3204	**Visualizzazione Scheda Articolo**
3205	**Stampa Valorizzazione Storica di Magazzino**
3206	**Stampa Giornale Fiscale di Magazzino**
3300	**Tabella Porti**
3301	**Tabella Vettori/Spedizionieri**
3302	**Tabella Agenti/Capizona**
3304	**Tabella Modalita' di Spedizione**
3305	**Tabella Aspetto dei Beni**
3306	**Tabella Causali di Trasporto**
4001	**Stampa Tabelle**
4002	**Stampa Anagrafica di Magazzino**
4003	**Stampa Classificazione ABC**
4004	**Stampa Situazioni di Magazzino**
4005	**Stampa Giornale di Magazzino (non fiscale)**
4006	**Stampa Schede Articoli di Magazzino**
4007	**Stampa Indici di Rotazione Scorte**
4008	**Stampa Codifica Articoli Cliente/Fornitore**
4009	**Stampa Articoli non Movimentati**
4010	**Stampa Tabella Sconti**
4011	**Stampa Listini**
4012	**Statistiche di Magazzino**

4500	**Immissione Documenti**
4501	**Revisione Documento**
4502	**Stampa Fatture Riepilogativa**
4503	**Stampa Documenti**
4504	**Situazione Ordini**
5000	**Tabella Condizioni di Pagamento**
5001	**Tabella Valute Estere**
5002	**Tabella Cambi Giornalieri**
5003	**Duplicazione Listini**
5004	**Variazione Prezzi di Listino**
5005	**Tabella Tipologie Contropartite**
5006	**Tabella Classificazione Centri di Costo**
5007	**Tabella Centri di Costo**
5008	**Tabella Ripartizione Centri di Costo**
5009	**Tabella Mastri**
5010	**Tabella Zone**
5011	**Tabella Tipi Documento Iva**
5012	**Tabella Tipi Assoggettamento mod. 770**
5013	**Tabella Tipi Documento mod. 770**
5014	**Tabella Codici Classificazione Bilancio**
5015	**Tabella Conti**
5016	**Tabella Registri IVA**
5017	**Tabella Periodi Fatturazione**
5018	**Tabella Marchi**
5019	**Tabella Sottoconti**
5020	**Tabella Contropartite**
5021	**Tabella Suddivisione Sconti Clienti**
5022	**Anagrafica Clienti/Fornitori**
5023	**Tabella Mansioni Aziendali**
5024	**Tabella Causali Contabili**
5025	**Interrogazione Giacenze di Magazzino**
5026	**Visualizzazione Bilancio Completo**
5027	**Visualizzazione Bilancio Clienti**
5028	**Visualizzazione Bilancio Fornitori**
5029	**Visualizzazione Bilancio Sottoconti**
5030	**Anagrafica Indirizzi di Spedizione**
5031	**Anagrafica Riferimenti Aziendali**
5032	**Stampa Giornale Contabile**
5033	**Stampa Registri Iva**
5034	**Stampa Bilancio di Verifica**

5035	Stampa Scadenzario
5036	Anagrafica Banche
5037	Destinazione Effetti Portafoglio Attivo
5038	Immissione Provvigioni
5039	Revisione Provvigioni
5040	Stampa Provvigioni
5041	Stampa Riferimenti Aziendali
5042	Visualizzazione Mastrini/Partitari
5043	Stampa Liquidazione Periodica Iva
5044	Variazione Credito Iva
5045	Stampe Clienti/Fornitori
5046	Immissione Prima Nota
5047	Stampa Liquidazione Annuale Iva
5048	Stampa Mastrini/Partitari
5049	Visualizzazione Schede Centri di Costo
5050	Stampa Schede Centri di Costo
5051	Tabella Sottotipi Classi di Costo
5052	Tabella Classi di Costo
5053	Servizi/Programmazione Telefonate
5054	Servizi/Word
5055	Servizi/Excel
5056	Servizi/Registratore Suoni
5057	Servizi/Paint Brush
5058	Servizi/Write
5059	Revisione Prima Nota
5060	Gestione Distinte Effetti Attivi
5061	Preparazione minidisco RIBA
5062	Stampa Distinta Effetti Attivi
5063	Stampa Effetti Attivi
5064	Stampa Domiciliazione Bancaria Effetti Attivi
5065	Stampa Lista Effetti Attivi
5066	Stampa Lista Distinte Effetti Attivi
5077	Revisione Distinte Effetti Attivi
5078	Tabella Budget
5079	Numerazione Pagine Bollato
5080	Stampa Registro Riepilogativo Iva
5081	Stampa Brogliaccio Contabile
5082	Anagrafica Distinte Base
5083	Stampa Distinte Base
5084	Immissione Modelli INTRA

5085	Revisione Modelli INTRA
5086	Stampa Modelli INTRA
5087	Generazione Automatica Chiusure/Aperture Bilancio
5088	Stampa Bilancio Analitico
5089	Ripartizione Automatica Movimenti Analitica
5090	Tabella Note Standard
5091	Anagrafica Commesse di Lavorazione
5092	Stampa Commesse di Lavorazione
5093	Stampa Indirizzi di Spedizione
5094	Calcolo Fabbisogno Commesse
5095	Calcolo Fabbisogno Ordini
5096	Impegno Materia Prima
5097	Scarico Materia Prima
5098	Carico Prodotto Finito
5099	Aggregazione Articoli per Listini
5100	Stampa Valorizzazione Attuale di Magazzino
5101	Revisione Movimenti di Magazzino da Produzione
5102	Stampa Elenchi Clienti/Fornitori
5103	Destinazione Pagamenti Fornitori
5104	Revisione Distinte Pagamenti Fornitore
5105	Presentazione Pagamenti a Fornitore
5106	Stampa Lista Scadenze Pagamenti a Fornitore
5107	Destinazione Pagamenti a Fornitore
5108	Tabella Classificazione Immagini Aziendali
5109	Cartelle Immagini Aziendali
5110	Inserimento Immagini Aziendali
5111	Ricerca Immagini Aziendali
5112	Cancellazione Movimenti Contabili di Simulazione
5113	Definizione Scorta di Sicurezza per Magazzino
5114	Preparazione minidisco INTRA
5115	Stampa Valorizzazione Fatturato
5116	Stampa Controllo di Gestione
5117	Definizione Riclassificazioni di Bilancio
5118	Tabella Riclassificazioni di Bilancio
5119	Contenuto Riclassificazioni di Bilancio
5120	Stampa Bilancio Riclassificato
5121	Descrizione in Lingua Trasporto a Cura
5122	Descrizione in Lingua Tabelle
5123	Ciclo Passivo
5124	Ricerca Documenti Fornitori

5125	Internet: SIGLA Home Page
5126	Send Mail
5127	Tabella categorie beni ammortizzabili
5128	Numeratori Cespiti
5129	Anagrafica Cespiti
5130	Immissione Movimenti Cespiti
5131	Revisione Movimenti Cespiti
5132	Stampa simulazione ammortamenti
5133	Simulazione ammortamenti
5134	Stampa registro beni ammortizzabili
5135	Opzioni
5136	Connetti a…
5137	Disconnetti
5138	Condividi SIGLA
5139	Classificazione Clienti/Fornitori
5140	Creazione Fatture Raggruppate
5141	Ripristino Bolle
5142	Sedi INPS
5143	Tabella Tributi
5144	Stampa tabella condizioni particolari acquisto/vendita
5145	Immissione/revisione movimenti comensi a terzi
5146	Revisione versamenti IRPEF/INPS
5147	Recupero versamenti sospesi
5148	Visualizzazione schede percipiente
5149	Stampa distinta versameno INPS
5150	Stampa distinta versamento IRPEF/ENASARCO
5151	Ristampa distinte
5152	Stampa CPA
5153	Stampa Certificati
5154	Stampa movimenti nota di credito
5155	Stampa schede percipiente
5156	Stampa versamenti
5157	Lista insoluti per cliente/agente
5158	Riepilogo insoluti
5159	Stampa lettere di sollecito
5251	Tabella unità di misura
5268	Flussi di cassa
5269	Protocollo telematico Intrastat
5272	Stampa Black List
5273	Analisi di Benford

La funzione SIGLAPPDoMenuCommand consente di gestire eventuali voci di menù aggiunte attraverso la funzione SIGLAPPGetMenuItem() o di sovrapporre gestioni alternative a quelle native di SIGLA.

Ad esempio, volendo sostituire la funzione di aggiornamento dell'anagrafica di magazzino di SIGLA con altra definita in CUSTOM.DLL si dovrà modificare SIGLAPPDoMenuCommand nel seguente modo:

```
BOOL DLLCALL SIGLAPPDoMenuCommand(int theMenuID)
{
        extern HINSTANCE customdll;
        switch(theMenuID) {
                case 3001:
                        {
                                void  (FAR PASCAL *newanag) (void);
                                FARPROC pp;
                                if(dll>HINSTANCE_ERROR) {
                                        pp=GetProcAddress(customdll,"NEWANAMAG");
                                        if(pp!=NULL) {
                                                newanag=(void (FAR PASCAL *)()) pp;
                                                (*newanag)();
                                        }
                                }
                        }
                        return FALSE;
}
```

SIGLAPPGetMenuItem

SCOPO: viene chiamata da SIGLA durante la generazione del menù operativo. Consente di creare un menù 'Personalizzazioni' (il menù viene creato se la funzione torna almeno una volta il valore TRUE). Per ogni voce del menù la funzione deve tornare l'ID e la descrizione della voce (max. 30 bytes). L'ID DEVE essere compreso fra 6000 e 6100. SIGLA continua a chiamare la funzione finché questa non torna il valore FALSE. Il primo carattere della descrizione viene usato come accellerator.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPGetMenuItem(int *MenuID**, LPSTR** menudescr**)**

Così come viene distribuita la funzione torna immediatamente il valore FALSE.

Volendo, ad esempio, aggiungere un menù "Personalizzazioni" composto di tre voci ci si deve comportare come segue:

* definire una variabile globale intera **int menuitem**
* azzerare tale variabile nel corpo della funzioen SIGLAPPInit che viene chiamata PRIMA che SIGLA generi il menù operativo
* modificare SIGLAPPGetMenuItem nel modo che segue:

```
BOOL DLLCALL SIGLAPPGetMenuItem(int  *MenuID,LPSTR menudescr)
{
        if(menuitem>=3) return FALSE;
        menuitem++;
        switch(menuitem) {
                case 1:
                        *MenuID=6001;
                        strcpy(menudescr,"A Prova 1");
                        break;
                case 2:
                        *MenuID=6002;
                        strcpy(menudescr,"B Prova 2");
                        break;
                case 3:
                        *MenuID=6003;
                        strcpy(menudescr,"C Prova 3");
                        break;
                }
                return TRUE;
        }
}
```

Si dovranno ovviamente gestire all'interno di SIGLAPPDoMenuCommand() gli eventi generati.

SIGLAPPSaveObject

SCOPO: SIGLA chiama questa funzione in determinati punti dell'applicazione DOPO aver proceduto alla memorizzazione di un dato (in immissione o in revisione). L'identificatore theActionID consente di stabilire quale procedura di SIGLA ha invocato la funzione mentre theObject fornisce le informazioni relative ai dati da salvare.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject**, int** theActionID**)**

Il puntatore theObject deve essere castato sull'oggetto appropriato (vedere le macro SPPSAVE_ che mappano alcuni valori possibili per theActionID).

Gli header files per effettuare il cast dell'oggetto passato come parametro sono distribuiti con il sistema di sviluppo di SIGLA. Poiché i nomi delle variabili membro delle classi di oggetti di SIGLA mappano, nella maggior parte dei casi, i nomi dei campi delle tabelle del database la ricerca fra i dati membro delle informazioni necessarie è piuttosto semplice.

Ad esempio, volendo intercettare il salvataggio di un articolo nell'anagrafica di magazzino per chiamare una funzione custom a cui passare codice e descrizione dell'articolo si può modificare la funzione come segue:

```
BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        extern HINSTANCE customdll;
         switch (theActionID) {
          case SPPSAVE_ANAMAG: {
                DPAnMag *theAnaMag = (DPAnMag *)theObject;
                void  (FAR PASCAL *SaveAnaMag) (LPSTR codice, LPSTR descrizione);
                FARPROC pp;
                if(customdll>HINSTANCE_ERROR) {
                pp=GetProcAddress(customdll,"SAVEANAMAG");
                if(pp!=NULL) {
                        SaveAnaMag=(void (FAR PASCAL *)()) pp;
                        (*SaveAnaMag)(theAnaMag->codice,theAnaMag->descrizion);
                        break;
                }
        }
}
return TRUE;
}
```

La seguente tabella mappa i valori possibili per il parametro theActionID e riporta le specifiche dell'oggetto theObject il cui reference viene passato

theActionID	theObject	Ambiente chiamante
1	DPAnMag	salvataggio anagrafica di magazzino
2	DPTaClF	salvataggio anagrafica clienti/fornitori
3	DPInSpe	salvataggio indirizzi di spedizione
4	DPTRiaz	salvataggio riferimenti aziendali
5	DPTaBan	salvataggio anagrafica banche
6	DPTaTli	salvataggio tabella listini
7	DPBaCod	salvataggio anagrafica codici a barre
8	DPDarli	salvataggio anagrafica descrizioni in lingua articoli
9	DPArFor	salvataggio codifica articoli clienti/fornitori
10	DPScont	salvataggio tabella sconti
11	DPCopVe	salvataggio condizioni particolari acquisto/vendita
12	DPTeDis	salvataggio distinta base
16	DPMovMag	salvataggio movimento di magazzino
17	DPMovMag	revisione movimento di magazzino (dopo aver salvato il dato)
18	DPLifoF	aggiornamento manuale storico LIFO/FIFO
19	DPLifoF	aggiornamento automatico storico LIFO/FIFO
20	DPMovMag	salvataggio automatico movimenti di apertura magazzino
21	DPPrAge	salvataggio manuale record di provvigioni
24	DPTeCom	salvataggio testata commessa di lavorazione

25	DPAbiCab	salvataggio sprortello bancario
26	DPCauCo	salvataggio causale contabile
27	DPTMans	salvataggio incarico aziendale
28	DPTCauMa	salvataggio causale di magazzino
29	DPAgArt	salvataggio aggregazione articoli per listini
30	DPTaRLi	salvataggio righe listini
33	DPCpg	salvataggio tipo pagamento
34	DPComun	salvataggio tabella comuni d'italia
35	DPUfCon	salvataggio uffici concessionari
36	DPUfCSe	salvataggio uffici centri servizio
37	DPUfIva	salvataggio uffici iva
38	DPUfReg	salvataggio uffici del registro
39	DPUImDi	salvataggio uffici imposte dirette
40	DPNuDoc	creazione numeratori
41	DPPerFa	salvataggio periodo fatturazione
43	DPTaCam	salvataggio cambio del giorno per una valuta
44	DPTaClB	salvataggio record tabella classificazione bilancio
45	DPTaVal	salvataggio valuta estera
46	DPTD770	salvataggio tipi documento 770
47	DPTDIva	salvataggio tipi documento Iva
48	DPTiDoc	salvataggio tipi documento di magazzino
49	DPAsBen	salvataggio aspetto beni
50	DPCaTra	salvataggio causali di trasporto
51	DPCoDog	salvataggio codici classificazione doganale
52	DPMoSpe	salvataggio modalita' di spedizione
53	DPTaPor	salvataggio tabella porti
54	DPTZone	salvataggio tabella zone
55	DPVetto	salvataggio vettore/spedizioniere
56	DPBudg	salvataggio budget
57	DPDirit	aggiornamento diritti dì accesso utenti
59	DPGmFam	salvataggio famiglie merceologiche
60	DPGmGru	salvataggio gruppi merceologici
61	DPGmSFa	salvataggio sottofamiglie merceologiche
62	DPReglv	salvataggio tabella registri iva
63	DPScCli	salvataggio tabella suddivisioni sconto per cliente/fornitore
64	DPTaSuS	salvataggio tabella suddivisioni sconto per articolo
65	DPTCont	salvataggio conto
66	DPTMast	salvataggio mastro
67	DPPiacon	salvataggio sottoconto

68	DPImbal	salvataggio imballi
69	DPRagFis	salvataggio raggruppamenti fiscali
70	DPRagCa	salvataggio raggruppamento causali di magazzino
71	DPTabMa	salvataggio tabella magazzini
72	DPTMarc	salvataggio tabella marchi
73	DPAgent	salvataggio anagrafica agenti/capizona
74	DPBolli	salvataggio record definizione bolli su tratte/RB
75	DPDeLin	salvataggio tabella lingue
76	DPTabUb	salvataggio tabella ubicazioni di magazzino
77	DPTaCon	salvataggio tabella contropartite
78	DPTaOff	salvataggio tabella offerte
79	DPCoAtt	salvataggio tabella codici attività
80	DPCoSta	salvataggio tabella codici statistici
82	DPEserc	creazione nuovo esercizio
83	DPCaSoc	salvataggio tabella cariche sociali
84	DPCentr	salvataggio tabella centri di costo
85	DPLiSoc	salvataggio tabella libri sociali
86	DPTA770	salvataggio tipi assoggettamento 770
87	DPTabNo	salvataggio nota documenti
88	DPTesClC	salvataggio tabella classi di costo
89	DPDitte	salvataggio tabella ditte
90	DPTalva	salvataggio aliquote iva
91	DPRagMa	salvataggio raggruppamenti magazzini
92	DPRipCC	salvataggio ripartizione centri di costo
93	DPStaEs	salvataggio stati esteri
94	DPTaTag	salvataggio tabella taglie
95	DPTCCCo	salvataggio tabella classificazione centri di costo
96	DPTStCC	salvataggio tabella raggruppamento classi di costo
97	DPTTiCo	salvataggio tipologie contropartite
98	DPDitte	variazione credito iva
99	DPUser	salvataggio utente
100	DPIntra	salvataggio manuale record INTRA
101	DPCoPar	salvataggio dati standard configurazione #1
102	DPCoPar	salvataggio dati standard configurazione #2
103	DPScMag	salvataggio scorta minima per magazzino
104	DPCarte	salvataggio cartelle immagini aziendali
105	DPTCla	salvataggio classificazione immagini aziendali
106	DPCoPar	salvataggio causali contabili e sottoconti per registrazione effetti

107	DPMovCo	revisione prima nota La funzione viene chiamata PRIMA che SIGLA esegua la revisione del raggruppamento individuato dal numero che è riportato nel campo <numero> dell'oggetto DPMovCo. Se la funzione torna il valore TRUE allora SIGLA esegue la revisione, altrimenti la revisione non viene effettuata. Questa feature consente di implementare un meccanismo di revisione prima nota alternativo a quello del pacchetto e di decidere quando usare l'applicazione standard e quando no
108	DPMovCo	termine registrazione prima nota non iva. L'oggetto DPMovCo ha come unico campo significativo <numero> che contiene il progressivo del raggruppamento di registrazione dei movimenti contabili inseriti.
109	DPMovMag	aggiunta riga durante la revisione di un documento. L'oggetto DPMovMag contiene tutte le informazioni relative alla riga salvata.
110	DPMovCo	Termine registrazione in prima nota. TheObject è di tipo DPMovCo, il dato membro <numero> (unico campo significativo) contiene il progressivo del raggruppamento di registrazione dei movimenti contabili e iva inseriti.
111	DPMovCo	Termine revisione prima nota. TheObject è di tipo DPMovCo e i sui dati membro contengono solo due dati significativi: <numero> contiene il progressivo del raggruppamento di registrazione dei movimenti contabili e iva modificati, <numprotoc> contiene il numero protocollo .
112	DPMovMag	modifica riga durante inserimento documenti. L'oggetto DPMovMag contiene tutte le informazioni relative alla riga modificata.
113	DPTeDoc	salvataggio documenti. Nel campo <numero> dell'oggetto DPTeDoc è contenuta la chiave che consente il recupero totale sia della testata del documento che delle righe
114	DPMovMag	modifica riga durante revisione documenti. L'oggetto DpMovMag contiene tutte le informazioni relative alla riga modificata.
115	DPTeDoc	salvataggio documento da revisione documenti. L'oggetto DPTeDoc contiene le informazioni relative alla testata del documento salvato.
116	DPMovCo	registrazione movimento contabile in immissione prima nota iva. N.B. la chiamata viene effettuata PRIMA che SIGLA abbia eseguito la COMMIT dell'intera registrazione
117	DPMovCo	registrazione movimento contabile in revisione prima nota iva. N.B. la chiamata viene effettuata PRIMA che SIGLA abbia eseguito la COMMIT dell'intera registrazione
118	DPMovCo	registrazione movimentazione contabile generata da un documento di magazzino. La variabile rifenumero di DPMovCo contiene il numero di registrazione della scrittura contabile.
119	DPMovCo	registrazione dello storno omaggi durante la movimentazione contabile generata da un documento di magazzino. La variabile di DPMovCo contiene il numero di registrazione della scrittura contabile.
120	DPTCCFo	salvataggio classificazione clienti/fornitori.
121	DPTeDoc	Revisione documenti: la chiamata viene effettuata dopo la pressione del tasto "Registra". Se la funzione torna il valore

		FALSE la registrazione del documento viene inibita.
122	DPTeDoc	Immissione documenti: la chiamata viene effettuata dopo la pressione del tasto "Registra". Se la funzione torna il valore FALSE la registrazione del documento viene inibita.
123	char *	Invio fax. La funzione viene chiamata quando SIGLA ha generato su disco le informazioni necessarie a DeltaFax per inoltrare il Fax. Il parametro passato è una stringa null terminated contenente il nome dei files generati (vedi specifiche fax)
124	DPTaTri	salvataggio tabella tributi.
125	DPSInps	salvataggio sedi inps.
126	DPMo770	salvataggio compensi a terzi. Nel campo <numero> dell'oggetto DPMo770 è contenuta la chiave per recuperare i movimenti salvati e i pagamenti contenuti nella tabella PAG770 eventualmente associati ai movimenti.
127	DPVeInps	salvataggio versamenti INPS mediante programma revisione versamenti INPS.
128	DPVIrpef	salvataggio versamenti IRPEF mediante programma revisione versamenti IRPEF.
129	DPTeDoc	termine registrazione immissione documenti, invocata dal bottone salva e stampa subito prima dell'invocazione della routine di stampa. Se la funzione torna FALSE allora SIGLA non esegue la routine di stampa.
130	DPTeDoc	termine registrazione revisione documenti, invocata dal bottone salva e stampa subito prima dell'invocazione della routine di stampa. Se la funzione torna FALSE allora SIGLA non esegue la routine di stampa.
131	DPMovMag	salvataggio riga durante l'immissione di un documento. L'oggetto DPMovMag contiene tutte le informazioni relative alla riga inserita.
132	DPCespi	Inserimento riga in anagrafica cespiti
133	DPMovCe	Immissione nuovo movimento cespiti
134	DPMovCe	Revisione movimento cespiti.
135	DPCespi	Immissione/Revisione movimenti cespiti: la chiamata viene effettuata subito prima di procedere all'aggiornamento automatico dell'anagrafica cespiti.
136	DPMovMag	Inserimento riga da ordine durante l'immissione dei documenti. L'oggetto DPMovMag contiene tutte le informazioni relative alla riga inserita.
137	DPMovMag	Inserimento riga da ordine durante la revisione dei documenti. L'oggetto DPMovMag contiene tutte le informazioni relative alla riga inserita.
138	DPTeDoc	Revisione documenti: invocata sul comando salva e stampa prima di fare i controlli di esistenza della contabilizzazione del documento può essere utilizzata per cancellare il lato contabile del documento, in modo da far ricontabilizzare.
139	DPMovCo	Revisione prima nota generale: invocata alla pressione del bottone cancella (solo per SIGLAPPDeleteObject).
140	DPMovCo	Immissione prima nota generale: invocata dopo aver effettuato la registrazione. TheObject è di tipo DPMovCo, il

		dato membro <numero> (unico campo significativo) contiene il progressivo del raggruppamento di registrazione dei movimenti contabili inseriti.
141	DPMovCo	Revisione prima nota generale: invocata dopo aver effettuato la registrazione. TheObject è di tipo DPMovCo, il dato membro <numero> (unico campo significativo) contiene il progressivo del raggruppamento di registrazione dei movimenti contabili inseriti.
142	DPQryMCo	Stampa effetti: invocata prima di procedere alla stampa, se torna FALSE non viene effettuata la stampa; nel campo <numdistint> dell'oggetto DPQryMCo è contenuto il numero della distinta (di scadenzario attivo) che è stata selezionata.
143	DPQryMCo	Stampa effetti: invocata dopo la stampa (anche se non fosse stata effettuata per la chiamata precedente); nel campo <numdistint> dell'oggetto DPQryMCo è contenuto il numero della distinta (di scadenzario attivo) che è stata selezionata.
144	DPMovMag	Impegno materia prima: invocata dopo avere registrato i mov. di impegno materie prime e aggiornato l'elenco delle commesse; nel campo <numero> c'e' la chiave per reperire i movimenti generati.
145	DPMovMag	Scarico materia prima: invocata dopo avere registrato i mov. di scarico materie prime e aggiornato l'elenco delle commesse; nel campo <numero> c'e' la chiave per reperire i movimenti generati.
146	DPMovMag	Carico prodotto finito: invocata dopo avere registrato i mov. di carico prodotti finiti e aggiornato l'elenco delle commesse; nel campo <numero> c'e' la chiave per reperire i movimenti generati.
147	DPMovMag	Immissione documenti decodifica codice articolo, se la funzione torna FALSE allora i valori impostati nei campi dell'oggetto DPMovMag vengono riportati a video; i campi dell'oggetto DPMovMag vengono riempiti con i valori digitati a video sino a quel momento.
148	DPMovMag	Revisione documenti decodifica codice articolo, se la funzione torna FALSE allora i valori impostati nei campi dell'oggetto DPMovMag vengono riportati a video; i campi dell'oggetto DPMovMag vengono riempiti con i valori digitati a video sino a quel momento.
149	DPMovMag	Immissione documenti: finche' la funzione torna FALSE, i valori restituiti nell'oggetto DPMovMag opportunamente riempito da un programma esterno che ha anche memorizzato le righe in MOVIMAG,vengono aggiunte alla List Box delle righe. Tale programma esterno non deve preoccuparsi di aggiornare le giacenze che vengono aggiornate automaticamente da SIGLA. I campi dell'oggetto DPMovMag vengono riempiti con i valori digitati a video sino a quel momento.
150	DPMovTag	Pressione del bottone OK sulla finestra di immissione quantita per taglia; i campi qnttag[0]..qnttag[19] dell'oggetto DPMovTag contengono le quantità per taglia digitate.
151	DPTeDoc	Pressione bottone SALVA/STAMPA in immissione documenti dopo aver stampato.
152	DPTeDoc	Pressione bottone SALVA/STAMPA in revisione documenti dopo aver stampato.

153	DPQryMCo	Registrazione Distinte port. attivo: prima della stampa definitiva se torna FALSE non viene effettuata la stampa (ma la disitnta viene comunque registrata).
154	DPQryMCo	Registrazione Distinte port. attivo: dopo la registrazione della distinta per effettuare la stampa, nel campo NUMDISTINT viene passato il numero della distinta, nel campo TIPOEFFETT viene passato il codice del tipo di effetto e quindi di distinta.
155	DPDist	Presentazione pagamento a fornitore: dopo la stampa definitiva delle distinte fornitori di tipo bonifico bancario e disposizione di pagamento (l'oggetto passato contiene tutti i dati della distinta).
156	DPDist	Presentazione pagamenti a fornitore (bon. e disp. pag.): prima della stampa definitiva della distinta se torna FALSE non viene effettuata la stampa ma viene comunque registrata la distinta (bon. e disp. pag.). L'oggetto DPDist conterrà nel dato membro mezzodipag "B" per il bonifico o "E" per la disposizione di pagamento.
157		USO INTERNO: NON USARE
158	DPTeDoc	Stampa documenti non di prova: in corrispondenza di questa uscita se la funzione SIGLAPPSaveObject ritorna FALSE (il valore di default normale è TRUE) si segnala alla stampa documenti di non eseguire la stampa le operazioni complementari, registrazione contabile, provviggioni ETC, ETC., verranno comunque eseguite. L'oggetto di classe DPTeDoc contiene i dati di testata del primo documento in stampa. NB: dalla versione 3.07 la variabile flag1 contiene il tipo documento scelto dall'operatore.
159	DPMovCo	Stampa di un qualsiasi documento Permette di raccogliere l'uscita in corrispondenza della fine di stampa di un singolo documento o di una singola fattura riepilogativa. L'oggetto DPMovCo contiene i seguenti valori validi: numero, eseregistr, datareg, esecompet, datacompet, dtcompcontab, dataprotoc, datadocum, numprotoc, numdocum, tipodociva, regiva. I campi i campi causale e c_f assumono dei valori particolari: causale: contiene il codice del tipo documento di magazzino in stampa; c_f: contiene "R" se si sta stampando una fattura riepilogativa oppure "I" in tutti gli altri casi.
160	DPPerSt	Stampa di un qualsiasi documento: prima della stampa di ogni riga vengono resi disponibili sull'oggetto DPPerSt i valori di tutti i campi configurabili sui moduli di stampa. N.B.: i valori calcolati dei campi relativi al piede pur essendo disponibili non sono comunque validi.
161	DPTeDoc	Stampa Fatture Riepilogative non di prova: in corrispondenza di questa uscita se la funzione SIGLAPPSaveObject ritorna FALSE (il valore di default normale è TRUE) si segnala alla stampa documenti di non eseguire la stampa; le operazioni complementari, registrazione contabile, provviggioni ETC, ETC., verranno comunque eseguite. L'oggetto di classe DPTeDoc contiene i dati di testata del

		primo documento in stampa.
162	DPTeDoc	Salva e Stampa Imissione Documenti: in corrispondenza di questa uscita se la funzione SIGLAPPSaveObject ritorna FALSE (il valore di default normale è TRUE) si segnala alla stampa documenti di non eseguire la stampa, mentre le eventuali operazioni complementari nel caso di stampa non di prova (registrazione contabile, provviggioni ecc.) verranno comunque eseguite.
163	DPTeDoc	Salva e Stampa Revisione Documenti: in corrispondenza di questa uscita se la funzione SIGLAPPSaveObject ritorna FALSE (il valore di default normale è TRUE) si segnala alla stampa documenti di non eseguire la stampa, mentre le eventuali operazioni complementari nel caso di stampa non di prova (registrazione contabile, provviggioni ecc.) verranno comunque eseguite.
164	DPTeDoc	Stampa Fatture Riepilogative di prova (V. uscita 161)
165	DPTeDoc	Stampa Documenti di prova (V. uscita 158)
166	DPMovMag	Immis. Documenti sostituzione codice articolo
167	DPMovMag	Revis. Documenti sostituzione codice articolo
168	DPTeDoc	Immis. Documenti sost./inser. delle informazioni di testata e piede
169		USO INTERNO: NON USARE
170	DPTeDoc	Pressione bottone ANNULLA da REVISIONE, pressione bottone FINE della finestra di lancio della revisione documenti, viene invocata SIGLAPPDeleteObject. TheObject è di tipo DPTeDoc e il campo valorizzato è numero.
171-175	DPStObj	Stampa di un qualsiasi documento. Subito dopo la stampa completa di un documento vengono resi disponibili sugli oggetti di classe DPStObj i valori dei campi configurabili sui moduli di stampa.
176-180	DPStObj	Valgono per queste uscite le cose dette per le 5 uscite precedenti, ma in questo caso gli importi restituiti sono in valuta se il documento è in valuta.
181-185	DPStObj	Prima della stampa di ogni riga vengono resi disponibili sugli oggetti DPStObj i valori dei campi configurabili sui moduli di stampa validi in quel momento.
186-190	DPStObj	Valgono per queste uscite le cose dette per le 5 uscite precedenti, ma in questo caso gli importi restituiti sono in valuta se il documento è in valuta.
191	DPPerSt	Stampa di un qualsiasi documento. Viene restituito il nome completo di path del file temporaneo che contiene la stampa nella variabile “nomefile” della classe DPPerSt. Per gli utenti della SppFrame l’oggetto passato alla SIGLAPPSaveObject è di tipo (char *). Quindi si deve provvedere alla lettura del “nomefile” agendo direttamente su un parametro di tipo puntatore a carattere.
192	DPMovCo	Registrazione riga pagamento in contabilita per compensi a terzi.
193	DPSalda	Modifica o cancellazione delle scadenze in contabilita per compensi a terzi

194	DPStObj	Stampa di un qualsiasi documento: Subito dopo la stampa di ogni riga vengono rese disponibili le note della riga documento. Si effettuano tante uscite quante sono le note da stampare. Il testo della nota si ottiene dal campo descrizione articolo (i=7;j=1).
195	DPSTObj	Segnala la fine della stampa stampa di un qualsiasi documento.
196	DPTeDoc	Ristampa Fatture Riepilogative in corrispondenza di questa uscita se la funzione SIGLAPPSaveObject ritorna FALSE (il valore di default normale è TRUE) si segnala a S++ di non stampare. L'oggetto di classe DPTeDoc contiene i dati di testata del primo documento in stampa .
197	DPAnMag	Anagrafica di magazzino alla pressione del tasto CANCELLA. Prima di cancellare il record si invoca la funzione SIGLAPPDeleteObject con theObject di tipo DPAnMag. Se il valore di ritorno è TRUE il record viene cancellato, se FALSE non viene cancellato.
198	DPTeDoc	Durante la stampa dei documenti riepilogativi (fatture,note credito) segnala il cambio di documento(bolla) fornendo i dati di testata.
199		Sviluppi Futuri non utilizzare
200	DPTaClF	Alla Pressione tasto ELIMINA dell'Anagrafica Clienti/Fornitori è invocata la funzione SIGLAPPDeleteObject con theObject di classe DPTaClF. Se la funzione ritorna TRUE SIGLA cancella il Cliente/Fornitore. Se la funzione ritorna FALSE SIGLA ignora il comando di cancellazione.
201	DPTabMa	Alla Pressione tasto ELIMINA della Tabella di magazzino è invocata la funzione SIGLAPPDeleteObject con theObject di classe DPTabMa. Se la funzione ritorna TRUE SIGLA cancella il Magazzino. Se la funzione ritorna FALSE SIGLA ignora il comando di cancellazione.
202	DPCauMa	Alla Pressione tasto ELIMINA della Tabella Causali Magazzino invocata la funzione SIGLAPPDeleteObject con theObject di classe DPCauMa. Se la funzione ritorna TRUE SIGLA cancella la Causale di Magazzino. Se la funzione ritorna FALSE SIGLA ignora il comando di cancellazione.
203	DPTiDoc	Alla Pressione tasto ELIMINA della Tabella Tipi Documento di Magazzino è invocata la funzione SIGLAPPDeleteObject con theObject di classe DPTiDoc. Se la funzione ritorna TRUE S++ cancella il Tipo Documento. Se la funzione ritorna FALSE S++ ignora il comando di

		cancellazione.
204	DPNote	Alla Pressione dei tasti: ESEGUI In stampa documenti e SalvaEStampa in immissione/revisione documenti si invoca la funzione SIGLAPPSaveObject con theObject di classe DPNote. Se la funzione restituisce TRUE S++ ignora l'uscita. Se la funzione restituisce FALSE S++ aggiunge alla WHERE per la selezione del cursore di stampa la stringa contenuta nel dato membro NOTA della classe DPNote. N.B.La stringa non può essere più lunga di 40 caratteri.
205	DPRiDis	Alla uscita del codice articolo in immissione di una nuova riga in anagrafica distinta base. theObject è di tipo DPRiDis, il dato membro codart contiene il codice articolo. Se la funzione restituisce FALSE il valore contenuto nei dati membro quantita e percricari (quest'ultimo solo se diverso da zero) viene impostato come quantità e percentuale di ricarico del componente.
206	DPDist	Dopo la creazione e stampa DEFINITIVA della distinta di presentazione effetti. theObject è di tipo DPDist, il dato membro numdistint contiene il numero della distinta creata, il dato membro tipoeffetto contiene la codifica del tipo di effetto che compone la distinta, il dato membro sottoconto contiene il sottoconto corrispondente alla banca di presentazione.
207	DPMovLot	Durante le stampe dei documenti fornisce per ogni riga documento i dati del lotto (se sono configurati in stampa)
208	DPQryMCo	Prima della stampa e creazione della distinta di presentazione effetti nel caso di stampa definitiva; viene invocata la funzione SIGLAPPSaveObject: theObject è di tipo DPQryMCo. Il dato membro sotdistint contiene il sottoconto banca di presentazione, il dato membro utente contiene l'eventale utente indicato, il dato membro att_passiv contiene 'A', il dato membro tipoeffett contiene la codifica del tipo di effetto, il dato membro datascad contiene la data scadenza iniziale, il dato membro datadocum contiene la data scadenza finale, il dato membro sospeso_sn contiene 'S' o 'N' in dipendenza dal tipo di effetto.
209	char *	In evasione degli ordini (sia in imm. che in rev. documenti) si invoca la funzione SIGLAPPSaveObject prima di lanciare la query di riempimento della form di esplosione delle testate ordine. Una personalizzazione può aggiungere alla clausola where una propria espressione. theObject è di tipo char *. In ingresso theObject contiene la clausola WHERE dello statement che il programma si appresta ad eseguire per riempire la finestra degli ordini. Per aggiungere un'espressione si deve far tornare alla funzione il valore FALSE, e l'espressione aggiuntiva deve essere sovrascritta in theObject. Per ulteriori informazioni consultare il file PERSDOC.DOC
210	DPMovMag	Dopo aver fatto doppio click su una riga della list box della finestra di revisione movimenti di magazzino viene invocata la funzione SaveObject per decidere se il movimento di

		magazzino può essere revisionato o meno. Se l'utente risponde FALSE allora viene dato un messaggio che il movimento non può essere revisionato e vengono disabilitati i pulsanti registra e elimina. Se viene riposto TRUE il movimento viene normalmente revisionato.

211	DPTeDoc	Dopo aver premuto il bottone di registrazione o di registrazione e contabilizzazione dall'ambiente Ciclo Passivo, viene invocata la funzione SaveObject per segnalare quali documenti sono stati presi in considerazione dal Ciclo Passivo (N.B.: la chiamata viene effettuata prima di salvare le eventuali modifiche alle righe dei documenti, cfr. theActionID=229). Se l'utente ha riepilogato n documenti fornitori, allora il programma Ciclo Passivo invoca la funzione SaveObject n+2 volte. La prima volta l'unico dato membro attualizzato della classe DPTeDoc è la variabile flag1 che contiene il carattere 'I'. Dopo di che' viene invocata n volte rendendo attuali le seguenti variabili:
		• tipoprotoc con il tipo protocollo
		• numeprotoc con il numero protocollo
		• dataprotoc con la data protocollo
		• numero con il numero, chiave della tabella TESDOCUM
		L'ultima chiamata che viene effettuata è analoga alla prima e l'unico dato membro attualizzato della classe DPTeDoc è la variabile flag1 che contiene il carattere 'F'. Una eventuale personalizzazione può utilizzare questa uscita per produrre un report dove si indicano le bolle che sono state prese in considerazione dal Ciclo Passivo. La prima chiamata consente di capire che il ciclo passivo iniziera' ad inviare le bolle, mentre l'ultima che le bolle visitate sono terminate.
		Es:
		DPTeDoc *theTeDoc;
		switch(theActionID) {
		case 211:
		theTeDoc= (DPTeDoc *) theObject;
		if(theTeDoc->flag1[0]=='I') {
		ApriFilesuDisco();
		}
		if(theTeDoc->flag1[0]==0) {
		AppendiFile(theTeDoc->tipoprotoc,theTeDoc->numeprotoc);
		}
		if(theTeDoc->flag1[0]=='F') {
		ChiudiFilesuDisco();
		}
		return(FALSE);
		}

212	DPTeDoc	In Immissione documenti viene segnalato l'evento uscita dal campo destinazione merce. E' possibile variare tutte le

		informazioni presenti in Testata1, Testata2, Testata3, Piede1, Piede2. Per maggiori dettagli vedere il documento PersDoc.doc
213	**Char ***	**In evasione degli ordini (sia in imm. che in rev. Documenti) si invoca la funzione SIGLAPPSaveObject prima di lanciare la query di riempimento della form di esplosione delle righe ordine. Una** **Personalizzazione può aggiungere alla clausola where una propria espressione. TheObject è di tipo char *.** **In ingresso theObject contiene il valore della colonna NUMERO dell'ordine che ci si appresta ad esplodere. Per aggiungere un'espressione si deve far tornare alla funzione il valore FALSE, e l'espressione aggiuntiva deve essere sovrascritta in theObject.** **Per ulterirori informazione consultare il documento PersDoc.doc**
214	**DPRiDis**	**Anagrafica distinta base: alla pressione del bottone aggiungi prima di aver inserito la riga (i dati della riga aggiunta sono in memoria). Viene invocata la funzione SIGLAPPSaveObject: theObject è di tipo DPRiDis i cui dati membro contengono tutte le informazioni della riga della distinta base.**
215	**DPRiDis**	**Anagrafica distinta base: alla pressione del bottone rimuovi prima aver eliminato la riga dalla listbox in memoria o alla pressione del bottone aggiungi se viene variato il codice articolo della riga (che sara' rimossa prima di aggiungere la nuova). Viene invocata la funzione SIGLAPPSaveObject: theObject è di tipo DPRiDis i cui dati membro contengono tutte le informazioni della riga della distinta base (se il dato membro quantità vale zero significa che la chiamata è stata effettuata alla pressione del bottone Aggiungi e la riga passata verrà sostituita).**
216	**DPTeDoc**	**In ciclo passivo, quando l'utente sceglie un documento da fatturare (doppio click su un documento da scegliere), è possibile per un programma esterno bloccare tale scelta. Si invoca la funzione SIGLAPPSaveObject dove theObject è di tipo DPTeDoc il cui dato membro numero contiene la chiave univoca per reperire il documento selezionato.** **Se una personalizzazione fa tornare il valore FALSE allora il documento scelto non viene caricato.**
217	**DPTeDoc**	**In Immissione documenti viene segnalato l'evento uscita dal campo FatturareA ed analogamente all'uscita dal campo cliente/fornitore, può essere raccolta l'uscita per variare tutte le informazioni presenti in Testata1, Testata2, Testata3, Piede1, Piede2.** **Per maggiori dettagli vedere il documento PersDoc.doc**
218	**char ***	**Prima di effettuare la query di ricerca sull'anagrafica articoli di magazzino da tutti gli ambienti. Lo scopo è di consentire la personalizzazione della stringa mostrata nella finestra di ricerca dell'anagrafica articoli di magazzino. Questa chiamata deve essere utilizzata per effettuare la selezione dei dati.** **Viene invocata la SIGLAPPSaveObject: theObject è puntatore a carattere (nei primi 2 caratteri contiene il numero del criterio di ricerca e nei successivi la chiave di ricerca).** **Se viene restituito il valore FALSE SIGLA non esegue la query**

		di ricerca e al momento di comporre la stringa da visualizzare nella finestra di ricerca effettua la chiamata successiva.
		Se il valore di ritorno TRUE viene eseguita la ricerca standard senza la successiva chiamata.
		
		Esempio di casting di theObject
		char *bufobj;
		switch(theActionID) {
		case 209:
		bufobj=(char *) theObject;

219	char *	Prima di comporre la stringa da inserire nella finestra di ricerca dell' anagrafica articoli se la chiamata precedente ha tornato il valore FALSE. Viene invocata la SIGLAPPSaveObject: theObject è puntatore a carattere che dovrà essere riempito con i dati da mostrare. Se viene restituito il valore TRUE si intende che terminato il blocco di righe relative allo stesso codice articolo; se viene restituito FALSE deve essere composta un'altra riga per lo stesso codice articolo. Per tutte le righe a partire dal carattere 80 DEVE essere inserito il codice dell'articolo.
220	char *	Questa uscita viene chiamata dopo che la 219 ha ritornato TRUE. Se torna TRUE vuol dire che ci sono altri articoli da inserire nella finestra di ricerca e viene nuovamente invocata la 219. Se ritorna FALSE il cursore è in EOF e quindi non ci sono più articoli da gestire. Esempio: L'uscita 218 serve alla personalizzazione per eseguire in proprio la query che recupera gli articoli da mostrare. E' possibile leggere il tipo di ricerca impostato dall'utente ("00" per descrizione, "01" per codice articolo ,"00" per codice articolo alternativo). Tornando FALSE si comunica a e-sigla++/Start++ che non deve eseguire la ricerca standard, ma semplicemente aprire la finestra e restare in attesa dell'uscita successiva (219). Nell'uscita 219 la personalizzazione dovrà costruire per ogni articolo reperito che si voglia mostrare, una stringa contente i dati da visualizzare restituendola a e-sigla++/Start++ e ritornare TRUE per passare all'uscita successiva (220). La stringa deve contenere a partire dal carattere 80 il codice articolo.Si vuole ottenere una visualizzazione che abbia piu righe per lo stesso articolo è possibile ritornare FALSE. L'uscita 219 sara' chiamata fintantoche' si torna FALSE. Nell'uscita 220 la personalizzazione scorre il cursore per mostrare il prossimo articolo e restituisce TRUE per per tornare all'uscita 219. Se non ci sono più articoli da mostrare si torna FALSE.
221	DPTeDoc	Immissione documenti alla pressione del bottone personalizzato del folder "Testata1". TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede digitati sino a quel momento. Questa uscita consente l'inserimento o la sostituzione dei valori della testata

		e piede.
222	**DPMovMag**	**Immissione documenti alla pressione del bottone personalizzato del folder "Righe1". TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga digitati sino a quel momento (N.B.: il codice destinazione merce è contenuto nel campo NUMCOLLEG). Questa uscita consente l'inserimento o la sostituzione dei valori della riga.**
223	**DPTeDoc**	**Immissione documenti alla pressione del bottone personalizzato del folder "Piede2". TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede digitati sino a quel momento. Questa uscita consente l'inserimento o la sostituzione dei valori della testata e piede.**
224	**DPTeDoc**	**Revisione documenti alla pressione del bottone personalizzato del folder "Testata1". TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede disponibili. Questa uscita consente unicamente l'inserimento o la sostituzione dei valori dei campi destinazione merce.**
225	**DPMovMag**	**Revisione documenti alla pressione del bottone personalizzato del folder "Righe1". TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con i dati della riga disponibili a video (N.B.: il codice destinazione merce è contenuto nel campo NUMCOLLEG). Questa uscita consente lettura e la sostituzione dei valori di rigas visualizzati al momento.**
226	**DPTeDoc**	**Revisione documenti alla pressione del bottone personalizzato del folder "Option". TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede disponibili. Questa uscita consente unicamente l'inserimento o la sostituzione dei valori dei campi destinazione merce.**
227	**DPMovMag**	**Questa uscita viene invocata in immissione documenti alla pressione del tasto NUOVO. TheObject è di tipo DPMovMag, il dato membro NUMERO contiene il numero dell'eventuale documento la cui immissione è annullata dalla pressione del bottone.**
228	**NULL**	**Questa uscita viene invocata dal programma di Configurazione al termine della procedura di ricalcalo dei saldi di magazzino.** **Il parametro theObject non è significativo.**
229	**NULL**	**Dopo aver premuto il bottone di registrazione o di registrazione e contabilizzazione dall'ambiente Ciclo Passivo, dopo il salvataggio dei documenti elaborati. I documenti interessati devono essere reperiti attraverso la chiamata con theActionID=211. Questa chiamata alla SIGLAPPSaveObject viene effettuata solo nel caso in cui sia stato premuto anche il bottone esplodi.** **Il parametro theObject non è significativo.**
230	**DPMovMag**	**Questa uscita viene invocata modalità di input da penna ottica dopo la lettura del codice a barre in fase di immissione di un documento. TheObject è di tipo DPMovMag, il dato membro ARTICOLO contiene il codice dell'articolo di magazzino. Viene invocata la SIGLAPPSaveObject, se il valore di ritorno è FALSE consente di impostare i campi quantità e quantità per confezione (attraverso i dati membro QUANTITA e QUANTCONFE del parametro theObject).**

231	**DPMovMag**	**Questa uscita viene invocata modalità di input da penna ottica dopo la lettura del codice a barre in fase di revisione di un documento. TheObject è di tipo DPMovMag, il dato membro ARTICOLO contiene il codice dell'articolo di magazzino. Viene invocata la SIGLAPPSaveObject, se il valore di ritorno è FALSE consente di impostare i campi quantità e quantità per confezione (attraverso i dati membro QUANTITA e QUANTCONFE del parametro theObject).**
232	**DPMovCo**	**Alla pressione del bottone 'Paga' in revisione distinte di pagamento fornitori.** **Le chiamate effettuate indicano quali scadenze vengono pagate (in questo caso viene chiamata la SIGLAPPDeleteObject) e i pagamenti effettuati (in questo caso viene chiamata la SIGLAPPSaveObject).** **La SIGLAPPDeleteObject() viene chiamata per capire quali scadenze sono state pagate: viene invocata una volta per ogni scadenza selezionata per il pagamento.** **La SIGLAPPSaveObject() viene chiamata per indicare l'inizio e la fine della elaborazione e per identificare il totale pagato per fornitore e sulla banca.** **La successione delle chiamate è la seguente:** **una chiamata alla SIGLAPPSaveObject per indicare l'inizio della procedura; i dati membro significativi dei theObject sono STATO, che vale "I", e CASO, che ha uno dei seguenti valori:** • **"1" BON, mov. unico banca SI** • **"2" BON, mov. unico banca NO** • **"3" DSP, mov. unico for. SI, mov. unico banca SI** • **"4" DSP, mov. unico for. SI, mov. unico banca NO** • **"5" DSP, mov. unico for. NO, mov. unico banca SI** • **"6" DSP, mov. unico for. NO, mov. unico banca NO** **le successive chiamate dipendono dal valore del dato membro CASO, e comunque sono la SIGLAPPDeleteObject per identificare le scadenze che vengono pagate (e che quindi saranno poi cancellate) e la SIGLAPPSaveObject per i movimenti che saranno registrati (sul fornitore e sulla banca);** **una chiamata alla SIGLAPPSaveObject per indicare la fine della procedura; l'unico dato membro significativo di theObject è STATO, che vale "F".** **Esempio di caso="3" (2 scadenze per il forn. F1, una scadenza per il forn. F2):** **SIGLAPPSaveObject, stato='I', caso='3'** **SIGLAPPDeleteObject per la prima scadenza di F1** **SIGLAPPDeleteObject per la seconda scadenza di F1** **SIGLAPPSaveObject per il totale pagato per F1** **SIGLAPPDeleteObject per la scadenza di F2** **SIGLAPPSaveObject per il totale pagato per F2** **SIGLAPPSaveObject per il totale pagato per la BANCA**

		8. SIGLAPPSaveObject, stato='F' **Per la chiamata alla SIGLAPPDeleteObject() il parametro theObject è di tipo DPMovCo, i dati membro significativi sono: numero, riga, datapresent, datadocum, numdocum, numdistint, sottoconto, c_f, sotdistint, segno, importo, codvaluta, impvaluta, impfattura, numrata, totrate, codbanca, codbancapr, descragg, deslingua, sospeso_sn, scadenz_sn, att_passiv, tipoeffett, stato, datavaluta, tiponumera, numrifsald, dataprotoc, numprotoc, contcorr, eimporto, eimpfattura, agente, capozona, datamatura, contropart.** **Per la chiamata alla SIGLAPPSaveObject() il parametro theObject è di tipo DPMovCo, i dati membro sono tutti valorrizati tranne gli importi nella valuta secondaria (lire o euro) rispetto alla valuta di stampa della distinta.**
233	**DPRiDis**	**Alla pressione del bottone 'Registra' in anagrafica distinta base prima di salvare.** **Vengono effettuate più chiamate successive alla SIGLAPPSaveObject per fornire i dati delle varie righe della distinta base mantenute in memoria.** **theObject è di tipo DPRiDis. Viene effettuata una prima chiamata in cui l'unico dato membro significativo è STATO che vale "I", se la chiamata torna FALSE non verranno effettuate altre chiamate e il salvataggio non viene effettuato. Se la chiamata precedente torna TRUE verranno effettuate n chiamate successive, dove n è il numero delle righe della distinta base; i dati membro significativi dell'oggetto theObject sono CODICE (codice della testata della d.b.), IDEVARIAN (identificativo della variante), CODART (codice del componente), QUANTITA e PERCRICARI. La sequenza termina con una chiamata alla SIGLAPPSaveObject, l'unico dato membro significativo è STATO che vale "F"; se la chiamata torna FALSE il salvataggio non viene effettuato.**
234	**DPMovCo**	**All'atto della contabilizzazione di qualsiasi documento da fatturare viene invocata la SIGLAPPSaveObject(). Il parametro theObject è di tipo DPMovCo l'unico dato membro significativo è NUMERO (contiene la chiave univoca per reperire il documento di magazzino, corrispondente cioè a TESDOCUM.NUMERO e MOVIMAG.NUMERO).** **Se la chiamata torna FALSE la contabilizzazione avviene con data di competenza uguale a quella impostata nel dato membro DATACOMPET (purchè non nulla e compresa tra le date di inizio e fine esercizio dell'esercizio selezionato).** **Da osservare che in caso di fatturazione riepilogativa la chiave univoca NUMERO individua il primo documento riepilogato in fattura.**
235	**DPMovCo**	**Alla registrazione dalla prima nota generale, prima di cancellare le scadenze pagate dal saldaconto viene invocata la SIGLAPPSaveObject().** **Il parametro theObject è di tipo DPMovCo i dati membro significativi sono NUMERO, RIGA e NUMRIFSALD (i primi due contengono la chiave univoca del movimento che sarà cancellato, numrifsald è il nuovo numero di riferimento della partita contabile che verrà attribuito).**
236	**DPTeDoc**	**Permette la rilevazione dello sconto pagamento e l'eventuale**

		sostituzione del valore a video durante l'immissione documenti. **Per ottenere la sostituzione è necessario ritornare FALSE.**
237	**DPTeDoc**	**Permette la rilevazione dello sconto cliente e l'eventuale sostituzione del valore a video durante l'immissione documenti.** **Per ottenere la sostituzione è necessario ritornare FALSE.**
238	**DPTeDoc**	**Permette la rilevazione dello sconto pagamento e l'eventuale sostituzione del valore a video durante la revisione documenti.** **Per ottenere la sostituzione è necessario ritornare FALSE.**
239	**DPTeDoc**	**Permette la rilevazione dello sconto cliente e l'eventuale sostituzione del valore a video durante la revisione documenti.** **Per ottenere la sostituzione è necessario ritornare FALSE.**
10021 **10035** **10064** **10152**	**DPPerRc**	**Inibisce la ricerca tradizionale di S++ per le tabelle :** **- Anagrafica di magazzino (ANAMAGA)(ID 10021,10035)** **- Banche (ABICAB) (ID 10064)** **- Testate Commesse (ID 10152)** **Tramite queste uscite è possibile lanciare una propria finestra di ricerca per le tabelle indicate precedentemente.** **Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** **- c_input (char[128]): contiene la stringa digitata sul campo da cui è partita la ricerca nel caso di Ricerca normale (freccettina rossa) in lettura;** **- c_output (char [128]): contiene il codice con cui S++ eseguirà la decodifica (da riempire a cura del programmatore);** **- c_tipo (double): contiene il criterio di ricerca richiesto dall'utente (1 ricerca normale - freccettine rosse - su codice; 0 ricerca normale - freccettine rosse - su descrizione; -1 per ricerca plus).** **Se la funzione torna TRUE continua a funzionare la ricerca tradizionale di S++.** **Se la funzione torna FALSE non sarà emessa nessuna ricerca da S++.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** **- C_INPUT** **- C_OUTPUT** **- C_TIPO** **ed hanno gli stessi significati esposti in precedenza.** **Esempio per la SIGPPDLL.DLL:** **#include "DPPerRc.h"** **BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject, int theActionID)** **{** **DPPerRc* thePers;** **BOOL aaa=FALSE;** **switch (theActionID)** **{** **...** **case 10021:// uscita alla pressione del tasto ricerca sul** **// campo codice o descrizione articolo**

		case 10035:
		thePers=(DPPerRc*) theObject;
		// Viene impostato fisso per esempio.
		strcpy(thePers->c_output,"CODICE ARTICOLO”);
		// si emette, per esempio, un messaggio windows
		// con la stringa digitata sul campo di ricerca.
		sprintf(buffer,"campo di ricerca %s",thePers->c_input);
		if(thePers->c_tipo>=0) MessageBox(NULL,
		buffer,"SIGPPDLL",
		MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
		aaa=FALSE;
		return(aaa);
		…
		case 10064:// ricerca personalizzata tabella ABICAB
		MessageBox(NULL, "RICERCA PERS", "SIGPPDLL",
		
		MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
		thePers=(DPPerRc*) theObject;
		strcpy(thePers->c_output,"00001.00001");
		sprintf(buffer,"campo di ricerca %s",thePers->c_input);
		if(thePers->c_tipo>=0) MessageBox(NULL, buffer,"SIGPPDLL",
		
		MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
		aaa=FALSE;
		return(aaa);
		...
		}
		return TRUE;

240 241 242	DPPerRc DPPerRc DPPerRc	Stampa Fatture Riepilogative Stampa Documenti Stampa da Immissione/revisione documenti Sostituisce la clausola ORDER BY nella fase di stampa dei documenti. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene la prima parte della stringa con il nuovo ordinamento (obbligatorio) - c_output (char [128]): contiene la parte finale dell’ordinamento (facoltativo). - c_tipo (double): vale 0 se si utilizza solo c_input vale 1 se si utilizza anche c_output Attenzione!!! La somma dei caratteri contenuti in c_input e c_output non deve superare 2040. Se la funzione torna TRUE l'ordinamento originale non sara' variato. Se la funzione torna FALSE sara' eseguito il nuovo ordinamento. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: • C_INPUT • C_OUTPUT • C_TIPO ed hanno gli stessi significati esposti in precedenza. ATTENZIONE !!! Queste uscite devono essere usate con cautela in quanto alcune funzioni del programma sono vincolate a un ben preciso ordinamento. Nel caso di stampa fatture riepilogative (ID=240) l'ordinamento è fondamentale per la costruzione del documento fattura riepilogativa. Quindi per i clienti che hanno in anagrafica l'opzione "Raggruppa bolle in fattura" è fondamentale che l'istruzione di ordinamento inizi :

		// con eurokit attivo
		ORDER BY MOVIMAG.SELEZIONE,TESDOCUM.VALCONT
		// senza eurokit attivo sulla ditta.
		ORDER BY MOVIMAG.SELEZIONE
		
		Inoltre se si utilizza anche il raggruppamento per indirizzi di spedizione:
		// con eurokit attivo
		ORDER BY MOVIMAG.SELEZIONE,TESDOCUM.VALCONT,TESDOCUM.CLI_FOR,TESDOCUM.LUOGO_DEST
		// senza eurokit attivo sulla ditta.
		ORDER BY MOVIMAG.SELEZIONE,TESDOCUM.CLI_FOR,TESDOCUM.LUOGO_DEST
		Nel caso di stampa documenti o SalvaEStampa da gestione documenti (ID=241,242),
		per garantire l'univocita' del documento stampato è necessaria la clausola:
		ORDER BY MOVIMAG.NUMERO

243	DPAnLot	Salvataggio anagrafica lotti (dopo la registrazione sulla tabella.)
244	DPMovMag	Immissione Documenti codice a barre sostituisce a video prezzi,sconti e maggiorazioni. Se la funzione torna FALSE sarnno presi in considerazione i valori dei campi: PRLORDO SCONTO1 se il numero di sconti configurato >0 SCONTO2 se il numero di sconti configurato >1 SCONTO3 se il numero di sconti configurato >2 SCONTO4 se il numero di sconti configurato >3 SCONTO5 se il numero di sconti configurato >4 MAGGIOR1 se maggiorazioni configurate >0 MAGGIOR2 se maggiorazioni configurate >1 Il campo prnetto sara' calcolato di conseguenza.
245	DPMovMag	Revisione Documenti immisione da codice a barre codice a barre sostituisce a video prezzi,sconti e maggiorazioni. (V.Uscita precedente.)
246	DPMovMag	Immissione Documenti. Se la funzione torna FALSE non sarà emessa la finestra delle note. Per la gestione esterna delle note sono messi a disposizione per la classe DPMovMag: Solo lettura (INPUT) • NUMERO: numero documento in immissione/modifica • RIGA: valore riga del riga in immissione/modifica • ESERCI: codice esercizio (per la SPPFrame il nome campo è ESERCIZIO). In lettura/scrittura (INPUT/OUTPUT): • NOTA_SN "S" se esiste già una nota, "N" altrimenti. Solo scrittura (OUTPUT): • DESARTICOL: ritorna la prima riga di note (max 40 caratteri).
247	DPMovMag	Revisione Documenti. Se la funzione torna FALSE non sara' emessa la finestra delle note. Per la gestione esterna delle note sono messi a disposizione per la classe DPMovMag i seguenti parametri: Solo lettura (INPUT) • NUMERO: numero documento in immissione/modifica • RIGA: valore riga del riga in immissione/modifica • ESERCI: codice esercizio (N.B.: per la SPPFrame il nome campo è ESERCIZIO) In Lettura/Scrittura (INPUT/OUTPUT): • NOTA_SN: "S" se esiste già una nota, "N" altrimenti.

		Solo Scrittura (OUTPUT): • **DESARTICOL: ritorna la prima riga di note (max 40 caratteri)** • **SOSPESO_SN: "S" segnala l'avvenuta modifica delle note e disabilita il bottone annulla sulla revisione documenti , "N" altrimenti.**
248	**DPOrdAut**	**Stampa pianificazione acquisti pressione del tasto Esegui.** **Il parametro theObject è di tipo DPOrdAut, i dati membro utilizzabili sono:** - **numero (char[10]): contiene il numero progressivo assegnato a tutti i record prodotti nella tabella per la generazione automatica dell'ordine fornitore per la stampa appena lanciata.**
249	**DPPerRc**	**Permette nella fase di stampa dei documenti di ottenere un uscita generica e nel caso in cui la funzione torni FALSE sara' possibile stampare un qualsiasi valore ad una determinata riga e colonna. Il campo da usare in configurazione moduli di stampa corrisponde nell'elenco dei campi possibili alla dicitura: "CALCOLATO DA PERSONALIZZAZIONE". Nella configurazione del campo sono utili le indicazioni di riga, colonna per posizionare il valore da stampare e una qualsiasi dicitura nell'apposito campo descrittivo che renda riconoscibile l'uscita in stampa.** **L'uscita è emessa durante il processo di composizione della riga in stampa per la riga e la colonna dove il campo è stato posizionato tramite la funzione di configurazione moduli di stampa.** **Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene la stringa descrittiva immessa in fase di configurazione moduli di stampa.** - **c_output (char [128]): contiene il valore di ritorno da stampare alle cordinate indicate in fase di configurazione moduli di stampa se il ritorno della funzione è FALSE.** **ATTENZIONE !!! per V4 il campo :** **"CALCOLATO DA PERSONALIZZAZIONE"** **È configurabile nella pagina "Controllo" del modulo di configurazione grafica stampe , scegliendo:** **"CALCOLATO "** **Il campo in stampa assume sempre il valore di c_output. Nel caso in cui c_output sia vuoto assume il valore di c_input.** **Quindi se si vuole ottenere come risultato di stampare un stringa vuota è necessario assegnare la stringa vuota anche a c_input.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
250	**DPPerRc**	**Permette nella fase di stampa dei documenti di ottenere un uscita generica e nel caso in cui la funzione torni FALSE sara' possibile stampare un qualsiasi valore ad una determinata riga e colonna. I campi per cui questa uscita è attiva in configurazione moduli di stampa corrispondono nell'elenco dei campi possibili alla dicitura:** • **CAMPO PERSONALIZZATO SU TESDOCUM** • **CAMPO PERSONALIZZATO SU CLIFO** • **CAMPO PERSONALIZZATO SU CFVEN** • **CAMPO PERSONALIZZATO SU ANAMAGA** • **CAMPO PERSONALIZZATO SU MOVIMAG** • **CAMPO PERSONALIZZATO PER LE NOTE** • **CAMPOPERSONALIZZATO PER MOVITAG.**

		• CAMPO PERSONALIZZATOVETTORI(CLIFO) • CAMPO PERSONALIZZATOVETTORI(CFVEN) Nella configurazione del campo sono utili le indicazioni di riga, colonna per posizionare il valore da stampare e una qualsiasi dicitura nell'apposito campo descrittivo che renda riconoscibile l'uscita in stampa. L'uscita è emessa durante il processo di composizione della riga in stampa con le coordinate per la riga e la colonna dove il campo è stato posizionato tramite la funzione di configurazione moduli di stampa. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene la stringa descrittiva immessa in fase di configurazione moduli di stampa che rappresenta il nome del campo per la tabella scelta. - c_output (char [128]): contiene il valore del campo configurato che al ritorno sara' stampato, alle cordinate indicate in fase di configurazione moduli di stampa, se il ritorno della funzione è FALSE. Può essere modificato. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
251	DPTeDoc	In fase di immissione di massa documenti di magazzino permette di aggiungere, ai controlli effettuati sui dati di testata di ogni documento, un controllo personalizzato. L'oggetto di classe DPTeDoc coniene tutti i valori letti per il documento in esame dal file d'importazione TDOCUM.TXT. Nel caso in cui la funzione ritorni FALSE il documento in esame sara scartato. In questo caso nel campo ANNOTAZION è possibile ritornare un messaggio che risultera' nel log stampato della procedura.
252	DPAnMag	Alla pressione del tasto Registra, prima della registrazione in anagrafica articoli di magazzino, permette la lettura dei dati dell'articolo in registrazione come erano prima delle eventuali modifiche apportate dall'utente. Il parametro theObject è di tipo DPAnMag, i dati membro contengono i dati prima della modifica. Questa chiamata viene effettuata anche in immissione di un nuovo articolo ma in questo caso caso i dati membro dell'oggetto DpanMag non sono valorizzati (in particolare il dato membro CODICE è una stringa nulla).
253	DPTaRLi	Alla registrazione del programma di immissione simultanea listini si ottiene un uscita dove l'oggetto di tipo DPTaRLi contiene i dati della riga di listino già inserita o modificata.
254	DPMovMa	In fase di immissione del movimento di magazzino l'uscita intercetta la pressione del tasto registra prima della registrazione della riga; se un'eventuale personalizzazione ritorna FALSE il movimento non viene registrato (in questo caso è anche possibile modificare il prezzo che verra' mostrato a video). Il parametro DPObject è di tipo DPMovMag i cui dati membro sono valorizzati con tutti i dati del movimento. E' possibile variare il prezzo (dati membro PRNETTO, EPRNETTO e PRVALUTA).
255	DPMovMa	In fase di revisione del movimento di magazzino l'uscita intercetta la pressione del tasto registra prima della registrazione della riga variata sulla tabella Movimag; se un'eventuale personalizzazione ritorna FALSE il movimento modificato non viene registrato (in questo caso è anche possibile modificare il prezzo che verra' mostrato a video). Il parametro DPObject è di tipo DPMovMag i cui dati membro sono valorizzati con tutti i dati del movimento. E' possibile variare il prezzo

		(dati membro PRNETTO, EPRNETTO e PRVALUTA).
256	DPAnMag	Alla pressione del tasto Registra, prima della registrazione in anagrafica articoli di magazzino pemette la lettura dei dati dell'articolo in registrazione. Se il ritorno è FALSE l'articolo non sara' registrato.
257	DPNote	Prima della stampa di ogni nota di un documento immediato o riepilogativo permette la lettura dei campi del record della tabella NOTE in stampa.
258	DPPerc	Prima della stampa di ogni nota di un documento immediato o riepilogativo permette di cambiare la riga di nota che verra' stampata. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): Contiene la riga in stampa con la nota. - c_output (char [128]): Il contenuto di questo campo sara' stampato se il ritorno della funzione è FALSE. N.B. poiche' la riga in stampa è 132 caratteri le posizioni di stampa da 129 a 132 risulteranno comunque sempre vuote. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
259	DPMovIva	Stampa registri iva: alla pressione del bottone esegui, prima di effettuare i controlli di sequenza dei protocolli; viene invocata la funzione SIGLAPPSaveObject. theObject è di tipo DPMovIva; i dati membro significativi sono registro (codice del registro iva che verrà stampato), mesecompet (mese di stampa), annocompet (anno di stampa), datareg (eventuale data finale di stampa, per default l'ultimo giorno del mese), stampat_sn ('N' se la stampa è di prova, 'S' se definitiva). Se la funzione torna FALSE viene controllato il valore del dato membro stato: • se stato='N' la stampa viene interdetta • se stato='S' l'elaborazione procede ma senza controllo di sequenza e senza evidenziazione delle anomalie nei protocolli • se stato='C' l'elaborazione procede senza controlli di sequenza ma con evidenziazione delle eventuali anomalie • per qualunque altro valore l'elaborazione procede come se il valore di ritorno della funzione fosse stato TRUE. Se la funzione torna TRUE l'elaborazione procede normalmente.
260	DPMovMag	In fase di immissione documenti impedisce l'aggiunta di una riga se il ritorno della funzione è FALSE. Nell'oggetto di classe DpMovMa sono contenuti i valori dei campi contenuti nei folder Riga1 e Riga 2.
261	DPMovMag	In fase di immissione documenti impedisce la rimozione di una riga se il ritorno della funzione è FALSE. Nell'oggetto di classe DpMovMa sono contenuti i valori dei campi contenuti nei folder Riga1 e Riga 2.
262	DPMovMag	In fase di revisone documenti impedisce l'aggiunta di una riga se il ritorno della funzione è FALSE. Nell'oggetto di classe DpMovMa sono contenuti i valori dei campi contenuti nei folder Riga1 e Riga 2.
263	DPMovMag	In fase di revisione documenti impedisce la rimozione di una riga se il ritorno della funzione è FALSE. Nell'oggetto di classe DpMovMa sono contenuti i valori dei campi contenuti nei folder Riga1 e Riga 2.
264	DPMovMag	In fase di revisione documenti impedisce la cancellazione del documento se il ritorno della funzione è FALSE.

		Nell'oggetto di classe DpMovMa sono contenuti i valori dei campi contenuti nei folder Riga1 e Riga 2.
265	**DPMovCo**	**In fase di revisione prima nota Iva impedisce il salvataggio del movimento se il ritorno della funzione è FALSE. Il parametro theObject è di tipo DPMovCo l'unico dato membro significativo è NUMERO che individua l'articolo contabile in revisione.**
266	**DPMovCo**	**In fase di revisione prima nota Iva impedisce la cancellazione del movimento se il ritorno della funzione è FALSE. Il parametro theObject è di tipo DPMovCo l'unico dato membro significativo è NUMERO che individua l'articolo contabile in revisione.**
267	**DPMovCo**	**In fase di revisione prima nota generale impedisce il salvataggio del movimento se il ritorno della funzione è FALSE. Il parametro theObject è di tipo DPMovCo l'unico dato membro significativo è NUMERO che individua l'articolo contabile in revisione.**
268	**DPMovCo**	**In fase di revisione prima nota generale impedisce la cancellazione del movimento se il ritorno della funzione è FALSE. Il parametro theObject è di tipo DPMovCo l'unico dato membro significativo è NUMERO che individua l'articolo contabile in revisione.**
269	**DPTeDoc**	**In fase di fatturazione riepilogativa termina la procedura se il ritorno della funzione è FALSE. Il parametro theObject è di tipo DPTeDoc i dati significativi sono :** • **CAUSCONTAB** • **NUMEPROTOC** • **DATAPROTOC** • **TIPOPROTOC** **N.B. Tramite la decodifica del codice riportato in CAUSCONTAB (causale contabile) è possibile reperire anche il codice del registro e del tipo documento iva con cui la fattura sara' eventualmente contabilizzata.**
270	**DPTeDoc**	**Immissione documenti dopo la scelta del codice valuta nel folder "Testata3". TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede digitati sino a quel momento. Se il ritorno della funzione è FALSE questa uscita consente l'inserimento o la sostituzione dei valori della testata e piede. Si ricorda che la valuta d'immissione si puo leggere nel campo VALUTA, nel caso in cui VALUTA è vuoto sara' il campo VALCONT ad indicare la valuta d'immissione (VALCONT='L' lire;VALCONT='E' EURO).**
271	**DPTeDoc**	**Immissione documenti Folder Righe1, dopo la decodifica del codice articolo. TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede digitati sino a quel momento. Se il ritorno della funzione è FALSE questa uscita consente l'inserimento o la sostituzione dei valori della testata e piede.**
272	**DPMovMag**	**Immissione documenti: in modalità di input da penna ottica viene invocata la SIGLAPPSaveObject; theObject è di tipo DPMovMag e il dato membro articolo contiene il codice a barre inserito. Se il valore di ritorno della funzione è FALSE, il codice a barre viene sostituito con quello restituito nel dato membro articolo dell'oggetto theObject.**
273	**DPMovMag**	**Revisione documenti: in modalità di input da penna ottica viene invocata la SIGLAPPSaveObject; theObject è di tipo DPMovMag e il dato membro articolo contiene il codice a barre inserito. Se il valore di ritorno della funzione è FALSE, il codice a barre viene sostituito con quello restituito nel dato membro articolo dell'oggetto theObject.**
274	**DPMovCo**	**Raggruppamento effetti, alla pressione del bottone Esegui nel caso di radiobox 'Definitiva' selezionato. Viene invocata una volta per ogni effetto che sarà raggruppato (e quindi cancellato) e una volta per ogni effetto raggruppato che verrà creato.**

		theObject è di tipo DPMovCo e suoi dati membro, eccetto numero e riga, sono attualizzati con tutti i dati degli effetti trattati. **In particolare la sequenza delle chiamate sarà composta da n chiamate dove il dato membro stato vale "C", una per ogni effetto che sarà raggruppato (e quindi cancellato) e una chiamata dove il dato membro stato vale "I", relativa all'effetto raggruppato che verrà creato. Il ciclo delle chiamate termina con una chiamata dove il dato membro stato vale "F" e tutti gli altri dati membro non sono significativi.** **ATTENZIONE: queste chiamate vengono effettuate all'interno della transazione pertanto eventuali accessi alle tabelle interessate dalla transazione (MOVCO, TASALSOT e DPCONFIG) potrebbero provocare blocchi e/o errori inaspettati.**
275	**DPTaClF**	**Immissione/revisione anagrafica clienti/fornitori. Se il ritorno della funzione è FALSE questa uscita consente di** **inibire la registrazione. L'oggetto contiene i dati in regsitrazione per la tabella CLIFO.**
276	**DPCFVen**	**Immissione/revisione anagrafica clienti/fornitori. Se il ritorno della funzione è FALSE questa uscita consente di** **inibire la registrazione. L'oggetto contiene i dati in registrazione per la tabella CFVEN.**
277	**DPPiacon**	**Immissione/revisione anagrafica clienti/fornitori. Se il ritorno della funzione è FALSE questa uscita consente di** **inibire la registrazione. L'oggetto contiene i dati in registrazione per la tabella PIACON.**
278	**NULL**	**CICLO PASSIVO: invocata alla pressione del bottone registra, theObject non è significativo**
279	**DPMovCo**	**CICLO PASSIVO: invocata alla pressione del bottone registra e contabilizza, theObject è di tipo DPMovCo e il dato membro numero contiene la chiave univoca per reperire il movimento contabile generato.**
280	**DPMovCo**	**Immissione/Revisione Prima Nota Iva: alla pressione del bottone registra viene invocata la funzione SIGLAPPSaveObject.** **theObject è di tipo DPMovCo e i dati membri significativi sono numero (sarà riempito solo in fase di revisione di una registrazione), datareg, esecompet, datacompet, dataprotoc, numprotoc (solo in revisione), datadocum, numdocum, sottoconto, c_f, contropart, causale, segno, importo, eimporto, codvaluta, impvaluta, regiva, tipodociva, caso e, dalla versione 4.11 mese di competenza IVA memorizzato nel campo ccosto.** **L'uscita non viene effettuata per le fatture extracee (caso=5) e per le fatture in sospensione di imposta (caso=2).** **Se il valore di ritorno della funzione è FALSE la registrazione non viene effettuata.** **N.B.: nel caso di Fatture Intracomunitarie (caso=1) i registri IVA interessati sono due e devono essere reperiti in base alla causale contabile.**
281	**DPTeDoc**	**Durante la funzione "Rispristino bolle", viene invocata prima di cancellare ogni eventuale Fattura/Nota Credito (creata dalla funzione "Creazione Fatture Raggruppate"). I dati disponibili riferiti al documento in cancellazione sono:** - **NUMERO** - **ESERPROTOC** - **NUMEPROTOC** - **TIPOPROTOC** - **DATAPROTOC** **ATTENZIONE: questa chiamata è effettuata all'interno di una transazione di cancellazione, pertanto eventuali accessi alle tabelle**

		interessate dalla transazione (TESDOCUM,MOVIMAG,NOTE) potrebbero provocare blocchi e/o errori inaspettati.
282	NULL	L'uscita viene emessa alla fine del processo di creazione fatture raggruppate. Non si passa nessun parametro valido.
283	DPPerRc	Durante la funzione "Rispristino bolle", viene invocata prima di ripristinare la bolla trovata. I dati disponibili riferiti al documento in ripristino sono contenuti nei dati membro : - c_input (char[128]): Contiene la chiave univoca della bolla (NUMERO) - c_output (char [128]): "S" se si siamo in è stato selezionato "Disattiva bolle selelzionate". Contiene "N" se è stato selezionato "Riattiva bolle selezionate". Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT ATTENZIONE: questa chiamata è effettuata all'interno di una transazione di aggiornamento, pertanto eventuali accessi alle tabelle interessate dalla transazione (TESDOCUM,MOVIMAG,NOTE) potrebbero provocare blocchi e/o errori inaspettati.
284	NULL	L'uscita viene emessa alla fine del processo di ripristino bolle. Non si passa nessun parametro valido.
285	DPPerRc	Durante la funzione "Elimina fatture riepilogative" viene invocata prima della riattivazione di ogni bolla collegata alla fattura in cancellazione. I dati disponibili riferiti al documento in ripristino sono contenuti nei dati membro : - c_input (char[128]): Contiene la chiave univoca della bolla (NUMERO) Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT ATTENZIONE: questa chiamata è effettuata all'interno di una transazione di aggiornamento, pertanto eventuali accessi alle tabelle interessate dalla transazione (TESDOCUM,MOVCO,MOVIVA,PROVAGEN,INTRA) potrebbero provocare blocchi e/o errori inaspettati.
286	DPMovCo	Raggruppamento scadenze passive, alla pressione del bottone Esegui nel caso di radiobox 'Definitiva' selezionato. Viene invocata una volta per ogni scadenza che sarà raggruppata (e quindi cancellata) e una volta per ogni scadenza raggruppata che verrà creata. theObject è di tipo DPMovCo e suoi dati membro, eccetto numero e riga, sono attualizzati con tutti i dati delle scadenze trattate. In particolare la sequenza delle chiamate sarà composta da n chiamate dove il dato membro stato vale "C", una per ogni scadenza che sarà raggruppata (e quindi cancellata) e una chiamata dove il dato membro stato vale "I", relativa alla scadenza raggruppata che verrà creata. Il ciclo delle chiamate termina con una chiamata dove il dato membro stato vale "F" e tutti gli altri dati membro non sono significativi. ATTENZIONE: queste chiamate vengono effettuate all'interno della transazione pertanto eventuali accessi alle tabelle interessate dalla transazione stessa (MOVCO e DPCONFIG) potrebbero provocare blocchi e/o errori inaspettati.
287	DPMovMag	CICLO PASSIVO: invocata alla pressione del bottone aggiungi nella pagina delle righe del documento.

		theObject è di tipo DPMovMag e i dati membro significativi sono numero, riga, articolo, contropart, quantita, sconto1, sconto2 , sconto3, sconto4, sconto5, maggior1, maggior2, codiva, eprlordo o se il documento è in valuta prvaluta, codvaluta, ecambio e datavaluta. **N.B.: ad esclusione di numero e riga gli altri dati disponibili sono quelli che l'utente può modificare e che saranno resi definitivi al momento della registrazione.**
288	**DPPerRc**	**Viene invocata alla pressione del tasto taglie delle seguenti funzioni:** - **Immissione/Revisione movimenti di magazzino** - **Immissione/Revisione documenti di magazzino** - **Gestione Ciclo Passivo** - **Evasione Ordini.** **Se il valore di ritorno della funzione è FALSE permette di parametrizzare il numero di taglie presenti nella finestra taglie. Il numero di taglie da visualizzare deve essere contenuto nel dato membro:** - **c_input (char[128]): (Numero di taglie da visualizzare).** **Nell'uso da SPPFRAME il nome campo con cui effettuare la SetField e':** - **C_INPUT**
289	**DPMovTag**	**Viene invocata alla pressione del tasto taglie durante l'immissione documenti.** **Se il valore di ritorno della funzione è FALSE la finestra taglie standard non vienen emessa, permettendone la sostituzione con una personalizzata. I valori di ritorno delle quantità per taglia immesse dall'utente devono essere specificati nei dati membro: qnttag[0], qnttag[1], qnttag[2], …. , qnttag[19].** **Nell'uso da SPPFRAME il nome campo con cui effettuare la SetField e':** - **QNTTAG01** - **QNTTAG02** - **QNTTAG03** - **…** - **QNTTAG20** **E' possibile usare lo stesso tipo di tecnica per sostituire la finestra taglie nelle altre funzioni dove è richiesta l'immissione/revisione per taglia.** **In dettaglio la funzione:** - **Immissione movimenti magazzino ha l'uscita 291** - **Revisione movimenti magazzino ha l'uscita 292** - **Revisione documenti magazzino ha l'uscita 293** - **Gestione Ciclo Passivo ha l'uscita 295** - **Evasione Righe Ordini in immissione documenti ha l'uscita 296.** - **Evasione Righe Ordini in revisione documenti ha l'uscita 294.** - **Evasione Lotti ha l'uscita 297** - **Immissione Documenti bottone Taglie per Quantita Evasa ha l'uscita 298.** - **Revisione Documenti bottone Taglie per Quantita evasa ha l'uscita 299.**
290	**DPMovTag**	**Viene invocata alla pressione del tasto taglie delle seguenti funzioni:** - **Immissione/Revisione movimenti di magazzino** - **Immissione/Revisione documenti di magazzino** - **Gestione Ciclo Passivo** - **Evasione Ordini.** **Se il valore di ritorno della funzione è FALSE, abilita l'immissione della quantita per taglia, solo per i campi taglia settati dalla personalizzazione, impendendo l'immissione per gli altri. Per abilitare all'immissione il primo campo taglia della finestra si deve porre a uno (1) il dato membro:**

		- qnttag[0] Per abilitare all'immissione il secondo campo taglia della finestra si deve porre a uno (1) i dato membro: - qnttag[1] Allo stesso modo si deve procedere per gli altri campi. Per i campi taglia della finestra che non si vogliono abilitati il corrispondente elemento di qnttag[] dovrà essere impostato a zero (0). Nell'uso da SPPFRAME il nome campo con cui effettuare la SetField e': - QNTTAG01 - QNTTAG02 - QNTTAG03 - … - QNTTAG20
291	DPMovTag	Viene invocata alla pressione del tasto taglie durante l'immissione movimenti di magazzino. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
292	DPMovTag	Viene invocata alla pressione del tasto taglie durante la revisione movimenti di magazzino. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
293	DPMovTag	Viene invocata alla pressione del tasto taglie durante Revisione Documenti di magazzino. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
294	DPMovTag	Viene invocata alla pressione del tasto taglie durante Evasione Ordini in Revisione Documenti di magazzino. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
295	DPMovTag	Viene invocata alla pressione del tasto taglie durante la Gestione Ciclo Passivo. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
296	DPMovTag	Viene invocata alla pressione del tasto taglie durante l' Evasione Ordini in Immissione Documenti di magazzino. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
297	DPMovTag	Viene invocata alla pressione del tasto taglie durante la gestione lotti di magazzino. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
298	DPMovTag	Viene invocata alla pressione del tasto taglie per le QUANTITA' EVASE durante l'immissione documenti. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
299	DPMovTag	Viene invocata alla pressione del tasto taglie per le QUANTITA' EVASE durante la revisione documenti. Ha la stessa funzione e le stesse specifiche dell'uscita 289.
300	DPMovCo	Registrazione Distinte effetti attivi: dopo la registrazione del giroconto automatico. theObject è di tipo DPMovCo, il dato membro NUMERO contiene la chiave per reperire il movimento.
301	DPPiacon	Visualizzazione schede contabili: invocata su decodifica del sottoconto. Il parametro theObject è di tipo DPPiaCon. Se il valore di ritorno è FALSE l'esecuzione resta bloccata sull'inserimento del codice sottoconto, se invece è TRUE (valore di default) prosegue normalmente.
302	DPPiacon	Visualizzazione schede contabili: dopo la pressione del tasto esegui. Il parametro theObject è di tipo DPPiaCon. Se il valore di ritorno è FALSE l'esecuzione resta bloccata sull'inserimento del codice sottoconto, se invece è TRUE (valore di default) prosegue normalmente.
303	DPTabUb	Anagrafica Ubicazioni: alla pressione tasto CANCELLA si invoca la funzione SIGLAPPDeleteObject con theObject di tipo DPTabUb prima di cancellare il record. Se la funzione ritorna TRUE (default) il record sarà cancellato se ritorna FALSE il record non sarà

		cancellato.
304	DPAnLot	Anagrafica Lotti: alla pressione tasto CANCELLA si invoca la funzione SIGLAPPDeleteObject con theObject di tipo DPAnLot prima di cancellare il record. Se la funzione ritorna TRUE il record sarà cancellato se ritorna FALSE il record non sarà cancellato.
305	char *	Permette di aggiungere una parte personalizzata nella clausola WHERE alla pressione del bottone AGGIUNGI della finestra di evasione ordini(testate). Se vuole aggiungere delle espressioni alla clausola, si deve ritornare FALSE e scrivere le espressioni aggiuntive nel parametro theObject (vedere anche uscita 209).
306	DPTeDoc	Viene invocata alla pressione del tasto "Copia" nella funzione: - Copia Documenti Piu' precisamente l'uscita viene invocata dopo la creazione del documento copia e prima dell'eventuale lancio della revisione del suddetto. L'oggetto passato contiene tutti i dati del documento copia, in più nel campo numecolleg viene passato il valore del campo numero del documento di cui è stata effettuata la copia.
307	char *	Viene invocata prima di lanciare la query di reperimento dei movimenti dalla configurazione S++: L'oggetto passato contiene la clausola where della query.
308	char *	Viene invocata prima di lanciare la query di reperimento dei movimenti dalla stampa del brogliaccio con pressione del bottone “Esporta” L'oggetto passato contiene la clausola where della query.
309	char *	Viene invocata dopo aver effettuato l'esportazione dei movimenti tramite la pressione del tasto “Esporta”. L'oggetto passato corrisponde al primo numero registrato su c:\XMOVCO.txt
310	char *	Viene invocata prima di lanciare la query di reperimento dei dati nella visualizzazione schede articolo di S++ (pressione del tasto ESEGUI). L’oggetto passato contiene la clausola where della query.
311	char *	Viene invocata prima di lanciare la query di reperimento dei dati nella revisione documenti di S++ (pressione del tasto ESEGUI). L’oggetto passato contiene la clausola where della query.
312	char *	Viene invocata prima di lanciare la query di reperimento dei dati nella revisione movimenti di magazzino di S++ (pressione del tasto ESEGUI). L’oggetto passato contiene la clausola where della query.
313	char *	Viene invocata prima di lanciare la query di reperimento dei dati dalla configurazione di S++ (pressione del tasto ESEGUI nella videata ESPORTAZIONE DOCUMENTI/MOVIMENTI). L’oggetto passato contiene la clausola where della query.
314	char *	Viene invocata durante l'esportazione dei movimenti tramite la pressione del tasto “Esporta” nella videata di REVISIONE DOCUMENTI per ogni documento esportato. L'oggetto passato corrisponde al numero registrato su c:\TDOCUM.txt
315	char *	Viene invocata durante l'esportazione dei movimenti tramite la pressione del tasto “Esporta” nella videata di REVISIONE MOVIMENTI DI MAGAZZINO per ogni documento/movimento esportato. L'oggetto passato corrisponde al numero registrato su c:\TDOCUM.txt/c:\RMOVMA.txt.
316	char *	Viene invocata durante l'esportazione dei movimenti tramite la pressione del tasto “Esporta” nella videata di VISUALIZZAZIONE SCHEDE ARTICOLO per ogni documento/movimento esportato. L'oggetto passato corrisponde al numero registrato su

		c:\TDOCUM.txt e c:\RMOVMA.txt.
317	DPMovMag	Viene invocata in fase di immissione documenti all'uscita dal campo Quantità per confezione, dopo aver effettuato tutte le operazioni relative al suddetto campo. TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga disponibili. Questa uscita consente l'inserimento o la sostituzione dei valori della riga.
318	DPMovMag	Viene invocata in fase di revisione documenti all'uscita dal campo Quantità per confezione, dopo aver effettuato tutte le operazioni relative al suddetto campo. TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga disponibili. Questa uscita consente l'inserimento o la sostituzione dei valori della riga.
319	DPMovMag	Viene invocata in fase di immissione documenti alla pressione del tasto Aggiungi prima di effettuare tutti i controlli di validità. Il parametro theObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga disponibili. Questa uscita consente l'inserimento o la sostituzione dei valori della riga (vale solo per quelli mostrati a video).
320	DPMovMag	Viene invocata in fase di revisione documenti alla pressione del tasto Aggiungi prima di effettuare tutti i controlli di validità. Il parametro theObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga disponibili. Questa uscita consente l'inserimento o la sostituzione dei valori della riga (vale solo per quelli mostrati a video).
321	DPTeDoc	Viene invocata in fase di immissione documenti alla pressione del tasto registra solo in caso di documento collegato. Il parametro theObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con le informazioni relative al documento principale e non al documento collegato.
322	NULL	Viene invocata in fase di immissione documenti alla uscita dal campo tipo documento. Se la funzione ritorna FALSE l'immissione è bloccata ed il focus mantenuto sul campo tipo documento. Se invece il valore di ritorno è TRUE (default) l'elaborazione procede regolarmente.

323	Char *	SOLO PER SIGLA (non per START).
		In Immissione/Revisione Documenti vene invocata alla pressione del tasto "Documenti" del folder "Righe1". Permette ad una personalizzazione di modificare il tipo di ordinamento con cui vengono presentate le testate dei documenti da evadere nell'apposita videata. Se l'uscita ritorna il valore TRUE l'ordinamento non viene modificato, altrimenti viene utilizzato come clausola di ordinamento la stringa contenuta il valore di ritorno contenuto in theObject.
		Esempio di casting di theObject.
		`char *bufobj;`
		`switch(theActionID) {`
		`case 323: .....`
		`bufobj=(char *) theObject;`
		`strcpy(bufobj," ORDER BY DPTEDOC.DATAPREV"`
		`}`

324	DPTeDoc	In Immissione Documenti al SalvaeStampa del documento se si ritorna FALSE permette di emettere un ulteriore pannellino di riepilogo, prima di effettuare la stampa del documento. Sono disponibili in lettura i valori di testata del documento.
325	DPTeDoc	In Revisione Documenti stessa funzione dell'uscita 324.

326	DPTeDoc	Durante la Stampa delle Fatture Riepilogative se si ritorna FALSE permette di attivare la stampa della destinazione merce insieme al riferimento bolla per ogni documento riepilogato in fattura, anche se non l'opzione non era stata selezionata dalla finestra di lancio (folder opzioni).

327	Char *	In immissione documenti viene invocata alla pressione del tasto MAIL. Se si torna FALSE permette di indicare il contenuto di theObject come destinatario nell'invio mail con deltacomm.
		In input theObject contiene l'indirizzo mail attuale per l'invio.
		Esempio :
		char *bufobj;
		switch(theActionID) {
		case 327:
		bufobj=(char *) theObject;
		strcpy(bufobj," cliente@info.it"
		}

328	Char *	In immissione documenti viene invocata alla pressione del tasto MAIL. Se si torna FALSE permette di indicare il contenuto di theObject come destinatario di posta nell'invio con il client di posta. In input theObject contiene l'indirizzo mail attuale per l'invio.
329	Char *	In Revisione Documenti viene invocata alla pressione del tasto MAIL. Se si torna FALSE permette di indicare il contenuto di theObject come destinatario di posta nell'invio mail con deltacomm. In input theObject contiene l'indirizzo mail attuale per l'invio.
330	Char *	In Revisione Documenti viene invocata alla pressione del tasto MAIL. Se si torna FALSE permette di indicare il contenuto di theObject come destinatario di posta nell'invio con il client di posta. In input theObject contiene l'indirizzo mail attuale per l'invio.
331	DPMovMag	In Immissione Documenti nella finestra di evasione delle righe documento, viene invocata alla pressione del tasto aggiungi, se la funzione ritorna FALSE, il campo quantità in evasione a video viene sostituito con il valore di theMovMag->quantita.
332	DPMovMag	In Revisione Documenti nella finestra di evasione delle righe documento, viene invocata alla pressione del tasto aggiungi, se la funzione ritorna FALSE, il campo quantità in evasione a video viene sostituito con il valore di theMovMag->quantita.
333	DPMovCo	Contabilizzazione RID attivi/passivi: dopo la pressione del bottone "Registra&Contabilizza". Il parametro theObject è di tipo DPMovCo, il dato membro numero contiene il numero del raggruppamento di record che compongono il pagamento/incasso dei RID, il dato membro att_passiv contiene il valore 'A' per i RID attivi o 'P' per i RID passivi. Attenzione: in base alle opzioni disponibili è possibile che questa uscita sia ripetuta più volte (viene comunque eseguita dopo la commit).
334	char *	Si invoca la SIGLAPPSaveObject alla selezione delle stampe bloccate prima di visualizzare o eseguire la ristampa. Il parametro theObject contiene il nome del file, comprensivo del percorso completo, contenente la stampa. Se la funzione ritorna FALSE la stampa non sarà eseguita.
335	char *	Si invoca la SIGLAPPSAveObject alla selezione delle stampe bloccate prima di visualizzare o eseguire la ristampa. Il parametro theObject contiene il titolo della stampa (vedere anche ID=334).

336	char *	Immissione/Revisione documenti. Se la funzione torna FALSE permette di definire nella pagina Righe2 “Prev.evas.” (data prevista evasione) come primo campo della pagina. In questo modo alla selezione della pagina Righe2 il cursore e il fuoco saranno attribuiti a questo campo.
337	char *	Permette di leggere da theObject la "WHERE" di selezione documenti che sarà usata dalla Stampa riepilogativa fatture.
338	char *	Permette di leggere da theObject la "WHERE" di selezione documenti che sarà usata dalla Ristampa riepilogativa fatture.
339	char *	Permette di leggere da theObject la "WHERE" di selezione documenti che sarà usata dalla stampa differita documenti.
340	DPMovMa	In fase di immissione documenti impedisce l' “Aggiungi Sopra” di una riga se il ritorno della funzione è FALSE. In theObject sono contenuti i valori dei campi delle pagine Riga1 e Riga 2.
341	DPMovMa	Revisione Documenti “Aggiungi Sopra” (V.340)
342	DPMovMa	Immissione documenti: in fase di evasione dell’ordine, sia parziale che totale, viene consentita la variazione dei dati ricavati dal documento evaso in particolare del prezzo.
343	DPPerRc	Anagrafica clienti/fornitori: l’uscita viene invocata all’avvenuto salvataggio di un cliente il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il codice del cliente fornitore appena registrato. - c_output (char [128]): contiene il codice introdotto dall’utente con cui verrà inizializzato il campo codice per una nuova immissione. - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output.
344	DPMovCo	Prima nota generale/saldaconto: alla pressione del tasto Rimuovi. L’oggetto e DPMovCo rende disponibili NUMERO e RIGA della scadenza cancellata. DOPO LA CANCELLAZIONE e IL SALVATAGGIO NELLA TABELLA SCADENZE
345	DPMovCo	Prima nota generale/saldaconto: alla pressione del tasto Collega, se la partita risultante ha saldo zero e l’utente richiede la cancellazione delle scadenze sospese residue, si esegue un uscita per ogni scadenza cancellata. L’oggetto e DPMovCo rende disponibili NUMERO e RIGA. DOPO LA CANCELLAZIONE e IL SALVATAGGIO NELLA TABELLA SCADENZE
346	DPIntrS	Stampa modelli Intra (CONTEGGIO FRONTESPIZIO DI UNA RIGA CON IMPORTO A 0): se si restituisce FALSE le righe con importo a 0 vengono stampate; restituendo TRUE(default) le righe a 0 non vengono stampate.
347	DPIntrS	Stampa modelli Intra (stampa di una riga con importo a 0): se si restituisce FALSE le righe con importo a 0 vengono stampate; restituendo TRUE(default) le righe a 0 non vengono stampate.
348	DPIntrS	Stampa modelli Intra (MINIDISCO DI UNA RIGA CON IMPORTO A 0): se si restituisce FALSE le righe con importo a 0 vengono stampate; restituendo TRUE(default) le righe a 0 non vengono stampate.
349	DPTiDoc	Viene invocata in fase di immissione documenti alla uscita dal campo tipo documento. Il parametro theObject è di tipo DPTiDoc e i suoi dati membro sono attualizzati con le informazioni relative al tipo documento selezionato. Se la funzione ritorna FALSE il focus è bloccato

		sul campo tipo documento. Se invece il valore di ritorno è TRUE (default) l'elaborazione procede regolarmente.
350	DPMovCo	Prima nota generale/saldaconto: alla pressione del tasto Rimuovi. L'oggetto e DPMovCo rende disponibili NUMERO e RIGA della scadenza cancellata PRIMA DELLA CANCELLAZIONE.
351	DPMovCo	Prima nota generale/saldaconto: alla pressione del tasto Collega, se la partita risultante ha saldo zero e l'utente richiede la cancellazione delle scadenze sospese residue, si esegue un uscita per ogni scadenza cancellata. L'oggetto e DPMovCo rende disponibili NUMERO e RIGA. PRIMA DELLA CANCELLAZIONE.
352	DPMovCo	Prima nota generale PRIMA DELLA CANCELLAZIONE di un pagamento collegato a scadenze salvate nella tabella SCADENZE. L'oggetto e DPMovCo rende disponibili NUMERO e RIGA del pagamento in cancellazione. Le scadenze sono leggibili tramite lo statment : SELECT * FROM SCADENZE WHERE NUMEROPAGA= DPMovCo.NUMERO AND RIGAPAGA=DPMovCo. RIGA
353	DPMovCo	Visualizzazione schede contabili a partite. Alla pressione del tasto Collega, l'oggetto di tipo DPMovCo contiene come dati validi in sola lettura: MOVCO.NUMRIFSALD e MOVCO.SOTTOCONTO della partita di riferimento. L'uscita interviene prima della rilettura dei movimenti contabili e del conseguente aggiornamento del video.
354	DPPerRc	Pemette di sostituire la clausola ORDER BY nella fase di Ristampa delle fatture riepilogative. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene la prima parte della stringa con il nuovo ordinamento (obbligatorio) - c_output (char [128]): contiene la parte finale dell'ordinamento (facoltativo). - c_tipo (double): vale 0 se si utilizza solo c_input vale 1 se si utilizza anche c_output Attenzione!!! La somma dei caratteri contenuti in c_input e c_output non deve superare 2040. Se la funzione torna TRUE l'ordinamento originale non sara' variato. Se la funzione torna FALSE sara' eseguito il nuovo ordinamento. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT - C_TIPO ed hanno gli stessi significati esposti in precedenza. ATTENZIONE !!! Questa uscita deve essere usata con cautela in quanto alcune funzioni del programma sono vincolate a un ben preciso ordinamento. Nella ristampa fatture riepilogative l'ordinamento è fondamentale per la costruzione del documento fattura riepilogativa. Quindi per i clienti che hanno in anagrafica l'opzione "Raggruppa bolle in fattura" è fondamentale che l'istruzione di ordinamento inizi con: ORDER BY TESDOCUM.NUMEDEFINI,TESDOCUM.DATADEFINI,MOVIMAG.SELEZIONE Inoltre se si utilizza anche il raggruppamento per indirizzi di spedizione: ORDER BY TESDOCUM.NUMEDEFINI,TESDOCUM.DATADEFINI,MOVIMAG.SELEZIONE,TESDOCUM.CLI_FOR,TESDOCUM.LUOGO_DEST
355	DPTeDoc	Revisione dei documenti DOPO LA CANCELLAZIONE DI UN DOCUMENTO l'oggetto di classe DPTeDoc rende disponibili i dati di testata del documento cancellato.

356	DPAnMag	Revisione articoli di magazzino. Alla decodifica del codice articolo, permette di leggere i dati. Se la funzione ritorna TRUE i dati letti dall'oggetto di classe DPAnMag saranno riportati a video.
357	DPTeDoc	In fase di stampa fatture riepilogative fornisce il tipo delle bolle selezionate (variabile tipodefini), il tipo di fattura riepilogativa scelta (variabile tipoprotoc) e il periodo di fatturazione (variabile tipocolleg). Il caso variabili blank indica "Tutti i tipi documento" (per tipoprotoc e tipodefini) o "Nessun periodo" (per tipocolleg).
358	DPMovMag	Copia Documenti in fase di generazione di ciascuna riga del documento copia viene invocata l'uscita, l'oggetto contiene i dati della riga generata che possono essere letti ma non modificati se non tramite una update.
359	DPCauCo	Contabilità/Immissione Prima Nota, dopo l'immissione della causale contabile e aver premuto il pulsante "Esegui", viene invocata l'uscita che consente di testare la causale contabile immessa. theObject è un oggetto di classe DPCauCo (tabella Causali Contabili.). Se la funzione ritorna il valore FALSE gli ambienti di immissione prima nota (genere o iva) non saranno lanciati.
360	Char*	Magazzino/Stampa Statistiche.Dopo l'immissione dei parametri di esecuzione e aver premuto il pulsante "Esegui", viene invocata l'uscita che consente di testare il magazzino iniziale e finale immesso. theObject è puntatore a carattere che contiene i codici di magazzino nella seguente forma "XXX YYY" (fra XXX e YYY c'è uno spazio bianco) dove: XXX = codice magazzino iniziale YYY = codice magazzino finale es: 001 111 in cui il magazzino iniziale è 001 e quello finale 111.
361	Char*	Magazzino/Visualizzazione Schede Articolo. Dopo l'immissione dei parametri di esecuzione e aver premuto il pulsante "Esegui", viene invocata l'uscita che consente di testare il magazzino corrente selezionato. theObject è puntatore a carattere che contiene il codice del magazzino selezionato.
330 362	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo "Causale" nella pagina "Generali" . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall'utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
363	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo "Causale" nella pagina "Saldac." . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall'utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo codice del primo folder

		viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
364	**DPPerRc**	**Prima nota generale: viene invocata alla decodifica del campo "Data extra contab." nella pagina "Altri" . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene il valore del campo digitato dall'utente.** - **c_output (char [128]): vuoto** - **c_tipo (double): Non significativo.** **Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
365	**DPPerRc**	**Prima nota generale: viene invocata alla decodifica del campo "Data registraz." nella pagina "Generali" . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene il valore del campo digitato dall'utente.** - **c_output (char [128]): vuoto** - **c_tipo (double): Non significativo.** **Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
366	**DPPerRc**	**Prima nota generale: viene invocata alla decodifica del campo "Data compet." nella pagina "Generali" . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene il valore del campo digitato dall'utente.** - **c_output (char [128]): vuoto** - **c_tipo (double): Non significativo.** **Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
367	**DPDist**	**Creazione del minidisco Ri.Ba./R.I.D./Bonifico: viene invocata alla pressione del bottone "Esegui" prima della creazione del file. Il parametro theObject è di tipo DPDist i cui dati membro contengono tutte le informazioni sulla distinta selezionata (in particolare - per individuare il tipo di distinta - il dato membro TIPOEFFETT, valorizzato solo per le distinte *attive*, vale '7' per le Ri.Ba. e 'F' per i R.I.D., mentre il dato membro MEZZODIPAG, valorizzato solo per le distinte *passive*, vale 'B' per il bonifico).** **Se la funzione ritorna il valore FALSE la procedura viene bloccata (N.B.:**

		non si procede alla generazione del file).
368	DPDist	Creazione del minidisco Ri.Ba./R.I.D./Bonifico: viene invocata alla fine della creazione del file immediatamente prima di mostrare la finestra di riepilogo con il nome del file e la richiesta di copiarlo su floppy. Il parametro theObject è di tipo DPDist i cui dati membro contengono tutte le informazioni sulla distinta selezionata (in particolare - per individuare il tipo di distinta - il dato membro TIPOEFFETT, valorizzato solo per le distinte *attive*, vale '7' per le Ri.Ba. e 'F' per i R.I.D., mentre il dato membro MEZZODIPAG, valorizzato solo per le distinte *passive*, vale 'B' per il bonifico). Il nome del file generato viene comunicato dalla uscita con theActionID=412 che è invocata immediatamente prima di questa uscita (il valore di default per il nome del file generato si trova nella cartella per i file di word, excel ecc. - ricavabile dal record con codice= 90010 della tabella DPCONFIG - e il nome è ottenuto premettendo al numero della distinta - dato membro NUMDISTINT - il carattere 'R' per le riba, 'D' per i RID o 'B' per i bonifici). Se la funzione ritorna il valore FALSE non viene mostrata la finestra di riepilogo.
369	DPDist	Presentazione Pagamenti: nel caso di pagamento a mezzo assegno di C/C viene invocata all'uscita del campo contenente la causale contabile e all'uscita del campo contenente il numero del primo assegno. Il parametro theObject è di tipo DPDist, gli unici dati membro significativi sono SOTTOCONTO (contiene il codice della banca di presentazione), STATO (vale 'C' se la funzione è stat invocata all'uscita del campo causale contabile, vale 'A' se invocata all'uscita del campo numero assegno) e NEFFETTI (contiene il numero dell'assegno eventualmente presente nel campo a video). Se la funzione ritorna il valore FALSE il contenuto del dato membro NEFFETTI viene inserito nel campo contenente il numero del primo assegno.
370	DPMovCo	Presentazione Pagamenti: nel caso di pagamento a mezzo assegno di C/C viene invocata alla fine della procedura, se eseguita in forma definitiva. Viene eseguita una chiamata per ogni pagamento generato più una ulteriore per indicare la fine della sequenza. Il parametro theObject è di tipo DPMovCo, gli unici dati membro significativi sono NUMERO (contiene la chiave per reperire la registrazione) e STATO (vale 'I' finchè NUMERO contiene un valore significativo, cioè dalla prima chiamata fino alla penultima, 'F' quando la sequenza delle chiamate termina, in questo caso NUMERO non contiene alcun valore significativo). Se alla prima chiamata la funzione torna il valore FALSE vengono eseguite anche le successive, altrimenti nessuna ulteriore chiamata viene eseguita.
371	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo "Dare" nella pagina "Generali" . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall'utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo "Dare" del della pagina "Generali" viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT

372	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo “Avere” nella pagina “Generali” . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall’utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo “Avere” della pagina “Generali” viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
373	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo “Sottoconto” nella pagina “Saldac.” . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall’utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo “Sottoconto” della pagina “Saldac.” viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
374	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo “Cliente/Fornitore ….” nella pagina “Selezion.” . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall’utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo “Cliente/Fornitore ….” nella pagina “Selezion.” viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
375	DPPerRc	Prima nota generale: viene invocata alla decodifica del campo “Avvocato” nella pagina “Insoluti” . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall’utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE il campo campo “Avvocato” nella pagina “Insoluti” viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
376	DPPerRc	Prima nota generale: viene invocata nella finestra “Insoluto” conseguente alla pressione del bottone “Insoluto” nella pagina

		"Saldac.", alla decodifica del campo "Banca". Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - **c_input (char[128]): contiene il valore del campo digitato dall'utente.** - **c_output (char [128]): vuoto** - **c_tipo (double): Non significativo.** **Se la funzione ritorna il valore FALSE il campo "Banca" viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
377	**DPMovMag**	**Viene invocata in fase di immissione documenti all'uscita dal campo Prezzo nella pagina righe1, dopo aver effettuato tutte le operazioni relative al suddetto campo. TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga disponibili. Questa uscita consente l'inserimento o la sostituzione dei valori della riga.**
378	**DPMovMag**	**Viene invocata in fase di revisione documenti all'uscita dal campo Prezzo nella pagina righe1, dopo aver effettuato tutte le operazioni relative al suddetto campo. TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con tutti i dati della riga disponibili. Questa uscita consente l'inserimento o la sostituzione dei valori della riga.**
379	**DPTaClF**	**Gestione ciclo passivo: si invoca la funzione SIGLAPPSaveObject all'uscita del campo codice fornitore.** **TheObject è di tipo DPTaClF e i suoi dati membro sono attualizzati con tutti i dati del fornitore selezionato.**
380	**DPPerRc**	**Gestione ciclo passivo: si invoca la funzione SIGLAPPSaveObject prima della esecuzione della query di selezione dei dati in modo da poter aggiungere una ulteriore condizione alla clausola WHERE dello statement.** **TheObject è di tipo DPPerRc e i sui dati membro non contengono dati significativi. Se la funzione torna FALSE il contenuto del dato membro C_OUTPUT sarà aggiunto alla fine della clausola where.** **N.B.: potranno essere aggiunte condizioni soltanto sui campi delle tabelle TESDOCUM e TIPODOCUM.**
381	**DPPerRc**	**Gestione ciclo passivo: si invoca la funzione SIGLAPPSaveObject prima della esecuzione della query di selezione dei dati in modo da poter sostituire la clausola ORDER BY dello statement.** **TheObject è di tipo DPPerRc, il dato membro C_INPUT contiene la clausola order by standard. Se la funzione torna FALSE il contenuto del dato membro C_OUTPUT viene utilizzato come clausola order by.** **N.B.: nella orber by personalizzata potranno essere utilizzati soltanto i campi delle tabelle TESDOCUM e TIPODOCUM.**
382	**DPAnLot**	**Immissione/Revisione lotti in documenti di carico. Se la funzione torna FALSE consente, nella fase di generazione/modifica di un lotto, di passare un codice lotto nuovo o già esistente, data di creazione di scadenza e descrizioni libere allegate ai lotti all'ambiente di gestione. Il codice passato dalla personalizzazione sarà editabile nell'ambiente di gestione lotti. L'uscita è valida solo per documenti di carico del lotto: bolla carico o nota credito intestata ad un cliente.**
383	**DPPerRc**	**Finestra per la scelta dei prezzi in immissione documenti. Invocata in fase di creazione della finestra consente ad una personalizzazione di indicare in essa un prezzo e proporlo di default.** **Nella classe DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene la didascalia che descrive il**

		prezzo personalizzato nella finestra d'immssione di scelta dei prezzi. - c_output (char [128]): contiene il valore del prezzo (i decimali sono separati da un ".") nella valuta d'immissione del documento. - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE la descrizione indicata in c_input e il prezzo indicato in c_output saranno mostrati nella finestra d'immssione prezzi. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
384	DPPerRc	Finestra per la scelta dei prezzi in revisione documenti. Invocata in fase di creazione della finestra consente ad una personalizzazione di indicare in essa un prezzo e proporlo di default. Nella classe DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene la didascalia che descrive il prezzo personalizzato nella finestra d'immssione di scelta dei prezzi. - c_output (char [128]): contiene il valore del prezzo (i decimali sono separati da un ".") nella valuta d'immissione del documento. - c_tipo (double): Non significativo. Se la funzione ritorna il valore FALSE la descrizione indicata in c_input e il prezzo indicato in c_output saranno mostrati nella finestra d'immssione prezzi. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - C_INPUT - C_OUTPUT
Da 385 a 392	DPPerRc	In fase di stampa lettere di sollecito scadenze permette di modificare le righe di intestazione della lettera. Nella classe DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene la riga di intestazione che sta per essere stampata. - c_output (char [128]): il contenuto di questa variabile può essere modificato e verrà stampato nel caso la funzione ritorni FALSE. - c_tipo (double): Non significativo. Le uscite da 387 a 389 vengono invocate se nell'anagrafica Cli/For è stato indicato un indirizzo per la spedizione dei documenti.
393	Char *	Invio mail documenti di magazzino con deltacomm. Se la funzione torna FALSE permette ad una personalizzazione di modificare il corpo della mail inviata tramite deltacomm. Il messaggio personalizzato da inserire in char * theObject può essere al massimo 190 caratteri.
394	DPImmAz	Durante la stampa documenti/fatture riepilogative permette di leggere e modificare i dati di archiviazione del documento/fattura riepilogativa in stampa nelle immagini aziendali prima della registrazione.
395	DPImmAz	Durante la stampa documenti/fatture riepilogative permette di leggere i dati di archiviazione nelle immagini aziendali dopo la registrazione.
396	DPPerRc	Prima nota iva: viene invocata alla decodifica del campo "Sottoconto" nella pagina "Generali" . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il valore del campo digitato dall'utente. - c_output (char [128]): vuoto - c_tipo (double): Non significativo.

		Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto. **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
397	**DPPerRc**	**Prima nota iva: viene invocata alla decodifica del campo "Sottoconto" nella pagina "Controp." . Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene il valore del campo digitato dall'utente.** - **c_output (char [128]): vuoto** - **c_tipo (double): Non significativo.** **Se la funzione ritorna il valore FALSE il campo codice del primo folder viene inizializzato con il valore del parametro c_output se e solo se c_output non è vuoto.** **Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono:** - **C_INPUT** - **C_OUTPUT**
398	**DPMovCo**	**In fase di stampa lettere di sollecito permette di individuare se la stampa effettuata è in lingua.** **La variabile da utilizzare, in sola lettura, è:** **- theMovCo->stato,** **che nel caso specifico contiene il codice della lingua selezionata per la stampa.**
399	**DPTeDoc**	**In fase di immissione da fonte dati esterna consente di leggere i dati della testata del documento già inseriti, i dati significativi sono: magazzino principale, magazzino collegato, tipo protocollo, data protocollo, numero documento, data documento, tipo pagamento, sconto pagamento, sconto cliente/fornitore, cliente/fornitore. Se la personalizzazione ritorna il valore FALSE l'immissione da fonte dati esterna viene bloccata.**
400	**DPMovMag**	**In immissione documenti è invocata alla pressione del bottone personalizzato del folder "Righe1" e permette di lanciare la revisione di una riga del documento. TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con i valori della riga in immissione/revisione al momento. Se la personalizzazione torna FALSE e il dato membro DPMovMag.riga contiene un numero di riga valido per il documento (>=0 e <numero di righe del documento) la riga indicata in DPMovMag.riga sarà selezionata per la revisione.** **Nell'uso da SPPFRAME il nome campo con cui effettuare la SetField/GetField è: RIGA.**
401	**DPMovMag**	**In revisione documenti è invocata alla pressione del bottone personalizzato del folder "Righe1" e permette di lanciare la revisione di una riga del documento. TheObject è di tipo DPMovMag e i suoi dati membro sono attualizzati con i valori della riga in immissione/revisione al momento. Se la personalizzazione torna FALSE e il dato membro DPMovMag.riga contiene un numero di riga valido per il documento (>=0 e <numero di righe del documento) la riga indicata in DPMovMag.riga sarà selezionata per la revisione. Nell'uso da SPPFRAME il nome campo con cui effettuare la SetField/GetField è: RIGA**
402	**DPTeDoc**	**Immissione documenti alla pressione del bottone Registra o SalvaEStampa prima di effettuare la registrazione del documento. TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con tutti i dati di testata e piede digitati sino a quel momento. Se il ritorno**

		della funzione è FALSE i valori modificati in DPTeDoc saranno riportati a video nei rispettivi campi di testata e/o piede.
403	DPTeDoc	Revisione documenti alla pressione del bottone Registra o SalvaEStampa prima di effettuare la registrazione del documento. TheObject è di tipo DPTeDoc e i suoi dati membro sono attualizzati con i dati di testata e piede digitati sino a quel momento. Se il ritorno della funzione è FALSE solo i campi destinazione merce sono modificati in DPTeDoc.
404	DPTeDoc	Immissione documenti invocata alla chiusura della finestra.
406	NULL	Inizio stampa solleciti scadenze. Se la funzione torna FALSE non si stampa.
407	NULL	Fine stampa solleciti scadenze.
408	DPMovMag	Immissione di massa documenti di magazzino. È invocata alla fine del processo. In theObject i dati membro utilizzabili sono: ▪ DPMovMag.numero contiene il NUMERO assegnato dal programma al primo documento importato. Riempito solo l'immissione non è di prova. ▪ DPMovMag.numerodoc contiene il valore della chiave univoca NUMERO assegnato dal programma dell'ULTIMO documento importato. Riempito solo l'immissione non è di prova. ▪ DPMovMag.tipo contiene il valore stringa OK se la immissione torna il messaggio "Immissione completata, senza errori". DPMovMag.tipo risulta vuoto se l'immissione torna il messaggio "Immissione terminata in maniera anomala" Se la funzione torna TRUE nel file di log sarà aggiunto un messaggio con il testo ricavato dal campo DPMovMag.descragg. Nell'uso da SPPFRAME i nomi campo con cui effettuare la SetField/GetField sono: - NUMERO - NUMERODOC - TIPO - DESCRAGG
409	DPPerRc	Imissione da fonte dati esterna da immissione documenti. Se la funzione torna FALSE permette di impostare il valore dell'ubicazione del documento collegato. La classe del parametro theObject è di tipo DPPerRc. Se la funzione torna FALSE i dati membro utilizzabili sono: - c_output (char [128]): Deve contenere il codice dell'ubicazione collegata. ATTENZIONE !!! NON VIENE ESEGUITO NESSUN CONTROLLO CIRCA LA VALIDITÀ DEL CODICE PASSATO. Nell'uso da SPPFRAME il nomi campo con cui effettuare la SetField è: - C_OUTPUT
410	DPPerRc	Generazione del documento collegato da immissione documenti. Se la funzione torna FALSE permette di impostare il valore del magazzino di riga del documento collegato. La classe del parametro theObject è di tipo DPPerRc. Il parametro c_input dell'oggetto viene impostato con il codice articolo di magazzino della riga. Se la funzione torna FALSE i dati membro utilizzabili sono: - c_output (char [128]): Deve contenere il codice del magazzino di riga del documento collegato. ATTENZIONE !!! NON VIENE ESEGUITO NESSUN CONTROLLO CIRCA LA VALIDITÀ DEL CODICE RESTITUITO. Nell'uso da SPPFRAME il nomi campo con cui effettuare la SetField è: - C_OUTPUT
411	DPTeDoc	Copia documenti. Viene invocata alla pressione del tasto copia. Se la funzione ritorna il valore FALSE la duplicazione del documento viene

		bloccata e la procedura rimane in attesa di nuovi comandi. La classe del parametro theObject è di tipo DPTeDoc. Il parametro numero dell'oggetto viene impostato con il numero del documento copia. Il campo VALCONT dell'oggetto riporta il valore 'S' se è stato attivato il checkbox mantieni prezzi del documento di origine 'N' altrimenti.
412	char *	Creazione del minidisco Ri.Ba./R.I.D./Bonifico: viene invocata subito prima della 368 (alla fine della creazione del file immediatamente prima di mostrare la finestra di riepilogo con il nome del file e la richiesta di copiarlo su floppy). Il parametro theObject è di tipo char * e contiene il nome del file
418	DPAnLot	Anagrafica lotti viene invocata al momento della pressione del tasto registra e consente di bloccare il salvataggio del codice lotto inserito nel caso che venga ritornato il valore FALSE.
419	DPAnLot	Finestra di immissione dei lotti. Viene invocata sui documenti di carico al momento della pressione del tasto registra. Consente di bloccare il salvataggio del codice lotto inserito nel caso che venga ritornato il valore FALSE. I campi lotto, articolo e fornitore sono sempre valorizzati con i dati del lotto corrente anche nel caso che questo non sia esistente in anagrafica.
420	DPAnLot	Finestra di immissione dei lotti. Viene invocata sui documenti di scarico al momento della pressione del tasto aggiungi. Consente di bloccare il salvataggio del codice lotto inserito nel caso che venga ritornato il valore FALSE.
421	DPTeDoc *	In revisione documenti permette di leggere i valori di testata del documento dopo la pressione del tasto Annulla. Se l'uscita torna FALSE il tasto Annulla non ha nessun effetto.
422	Char *	Durante l'invio MAIL da DELTACOM permette di leggere il nome del file INF completo del percorso. Ricordiamo che il file TXT associato ha lo stesso nome e stesso percorso. Il file INF per l'invio mail contiene una serie di sezioni e tags la cui documentazione in dettaglio può essere trovata nel manuale del deltacom.
423	Char *	Durante l'invio FAX da DELTACOM permette di leggere il nome del file INF completo del percorso. Ricordiamo che il file TXT associato ha lo stesso nome e stesso percorso. Il file INF per l'invio mail contiene una serie di sezioni e tags la cui documentazione in dettaglio può essere trovata nel manuale del deltacom.
424	DPTmpMat	Immissione documenti, consente, ritornando il valore FALSE, di sostituire il codice matricola generato in automatico da SIGLA con uno personalizzato posto in: DPTmpMat.matricola (SIGPPDLL.DLL) TMPCAMAT.MATRICOLA (SPPFRAME)
425	DPAnMat	Finestra di inserimento delle matricole, consente, ritornando il valore FALSE, di sostiruire il codice matricola generato in automatico da SIGLA con uno personalizzato posto in: DPAnMat.matricola (SIGPPDLL.DLL) ANAMATR.MATRICOLA (SPPFRAME)
426	DPAnMat	Revisione documenti consente, ritornando il valore FALSE, di sostiruire il codice matricola generato in automatico da SIGLA con uno personalizzato posto in: DPAnMat.matricola (SIGPPDLL.DLL) ANAMATR.MATRICOLA (SPPFRAME)
427	DPMovMag	In immissione documenti sul campo quantità, permette di leggere tutti i valori a video sui folder "Righe1" e "Righe2" e di modificarli nel caso si

		ritorni il valore FALSE.
428	DPTeDoc	In immissione documenti sul campo quantità, permette di leggere tutti i valori a video sui folder “Testata1”, “Testata2”, “Piede1” e “Piede2” e di modificarli nel caso si ritorni il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 427).
429	DPMovMag	In immissione documenti sul campo quantità, permette di bloccare la normale prosecuzione del programma ritornando il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 428).
430	DPMovMag	In immissione documenti sul campo quantità per confezione, permette di leggere tutti i valori a video sui folder “Righe1” e “Righe2” e di modificarli nel caso si ritorni il valore FALSE.
431	DPTeDoc	In immissione documenti sul campo quantità per confezione, permette di leggere tutti i valori a video sui folder “Testata1”, “Testata2”, “Piede1” e “Piede2” e di modificarli nel caso si ritorni il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 430).
432	DPMovMag	In immissione documenti sul campo quantità per confezione, permette di bloccare la normale prosecuzione del programma ritornando il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 431).
433	DPMovMag	In revisione documenti sul campo quantità, permette di leggere tutti i valori a video sui folder “Righe1” e “Righe2” e di modificarli nel caso si ritorni il valore FALSE.
434	DPTeDoc	In revisione documenti sul campo quantità, permette di leggere tutti i valori a video sui folder “Testata1”, “Testata2”, “Piede1” e “Piede2” e di modificarli nel caso si ritorni il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 433).
435	DPMovMag	In revisione documenti sul campo quantità, permette di bloccare la normale prosecuzione del programma ritornando il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 434).
436	DPMovMag	In revisione documenti sul campo quantità per confezione, permette di leggere tutti i valori a video sui folder “Righe1” e “Righe2” e di modificarli nel caso si ritorni il valore FALSE.
437	DPTeDoc	In revisione documenti sul campo quantità per confezione, permette di leggere tutti i valori a video sui folder “Testata1”, “Testata2”, “Piede1” e “Piede2” e di modificarli nel caso si ritorni il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 436).
438	DPMovMag	In revisione documenti sul campo quantità per confezione, permette di bloccare la normale prosecuzione del programma ritornando il valore FALSE. Questa uscita è eseguita dopo la precedente (Id 437).
439	DPPiacon	In immissione sottoconti sulla pressione del tasto registra, permette di bloccare la registrazione del sottoconto se si ritorna il valore FALSE.
440	DPTaClF	In anagrafica clienti/fornitori/prospect alla decodifica del codice permette di bloccare la revisione ritornando FALSE.
441	Char*	Magazzino/Stampa Situazioni. Dopo l’immissione dei parametri di esecuzione e aver premuto il pulsante “Esegui”, viene invocata l’uscita che consente di testare il magazzino corrente selezionato. theObject è puntatore a carattere che contiene il codice del magazzino selezionato. Se la scelta è su tutti i magazzini theObject riferisce una stringa composta da spazi bianchi.
442	DPAnLot	Carico prodotto finito. Alla pressione del tasto esegui, consente di intercettare i nuovi lotti creati prima che vengano registrati nella tabella. E’ quindi possibile modificarli in modo personalizzato.
443	Char *	Invocata durante la stampa di un documento in cui è configurato il campo LOTTI COMULATI (V.cfgstampe.xls). Il parametro passato è una stringa null terminated contenente la riga di stampa relativa al lotto comulato. Se l’uscita torna FALSE le modifiche effettuate al parametro passato saranno riportate in stampa.

<table>
<tr><td>444</td><td>DPMovCo</td><td>Invocata in fase di immissione prima nota, alla pressione del bottone Eesegui. Permette di leggere i dati immessi a video dall'utente. Se la funzione ritorna il valore FALSE gli ambienti di immissione prima nota (genere o iva) non saranno lanciati. I dati immessi a video dall'utente copiati nella classe DPMovCo sono:
<table>
<tr><th>Dato membro classe DPMovCo</th><th>Descrizione</th><th>Nome simbolico per SPPFRAME</th></tr>
<tr><td>causale</td><td>Causale contabile</td><td>CAUSALE</td></tr>
<tr><td>datareg</td><td>Data registrazione</td><td>DATAREG</td></tr>
<tr><td>datacompet</td><td>Data competenza</td><td>DATACOMPET</td></tr>
<tr><td>regiva</td><td>Registro iva</td><td>REGIVA</td></tr>
<tr><td>tipodociva</td><td>Registro vendite</td><td>TIPODOCIVA</td></tr>
<tr><td>sottoconto</td><td>Codice modello di registrazione</td><td>SOTTOCONTO</td></tr>
</table></td></tr>
<tr><td>445</td><td>DPMovMag</td><td>Invocata in Immissione documenti se è stata selezionata una riga con un doppio click e si preme il secondo bottone di personalizzazione in Righe1 o Righe 2. TheObject è di tipo DPMovMag, i suoi dati membro sono attualizzati con i dati della riga a video digitati sino a quel momento (N.B.: il codice destinazione merce è contenuto nel campo NUMCOLLEG). Se l'uscita ritorna FALSE i valori a video della riga in revisione sono sostituiti con quelli di DPMovMag.</td></tr>
<tr><td>446</td><td>DPMovMag</td><td>Invocata in Immissione documenti alla pressione dei nuovi bottoni personalizzati del folder "Righe1" e "Righe2"solo nel caso che NON sia stata selezionata una riga per la revisione. TheObject è di tipo DPMovMag, se esiste una riga evidenziata con un singolo click, nei suoi in dati membro è possibile leggere:
<table>
<tr><th>Dato membro classe DPMovMag</th><th>Descrizione</th><th>Nome simbolico per SPPFRAME</th></tr>
<tr><td>numero</td><td>Numero univoco del documento</td><td>NUMERO</td></tr>
<tr><td>riga</td><td>Numero di riga attuale(soggetto arinumerazione alla registrazione)</td><td>RIGA</td></tr>
</table>
N.B. Nel caso che il campo riga riporti il valore -1 significa che non esiste nessuna riga immessa o non è stata indicata nessuna riga con il singolo click.</td></tr>
<tr><td>447</td><td>DPMovMag</td><td>Invocata in Revisone documenti se è stata selezionata una riga con un doppio click e si preme il secondo bottone di personalizzazione in Righe1 o Righe 2. Questa uscita consente di leggere e/o sostituire i valori a video della riga in revisione. TheObject è di tipo DPMovMag, i suoi dati membro sono attualizzati con i dati della riga a video digitati sino a quel momento (N.B.: il codice destinazione merce è contenuto nel campo NUMCOLLEG). Se l'uscita torna FALSE i valori a video della riga in revisione saranno sostituiti con quelli contenuti in DPMovMag.</td></tr>
</table>

<table>
<tr><td>448</td><td>DPMovMag</td><td>Invocata in Revisione documenti alla pressione dei nuovi bottoni personalizzati del folder “Righe1” e “Righe2”solo nel caso che NON sia stata selezionata una riga per la revisione. TheObject è di tipo DPMovMag, se esiste una riga evidenziata con un singolo click, nei suoi in dati membro è possibile leggere:
<table>
<tr><th>Dato membro classe DPMovMag</th><th>Descrizione</th><th>Nome simbolico per SPPFRAME</th></tr>
<tr><td>numero</td><td>Numero univoco del documento</td><td>NUMERO</td></tr>
<tr><td>riga</td><td>Numero di riga attuale(soggetto a rinumerazione alla registrazione)</td><td>RIGA</td></tr>
</table>
N.B. Nel caso che il campo riga riporti il valore -1 significa che non esiste nessuna riga immessa o non è stata indicata nessuna riga con il singolo click.</td></tr>
<tr><td>449</td><td>DPCpg</td><td>Invocata in stampa documenti all’inizio del calcolo delle scadenze per il documento in stampa se il documento stesso ha indicata un codice di pagamento. TheObject è di classe DPCpg (testata codice di pagamento) e contiene i dati di testata del codice di pagamento.</td></tr>
<tr><td>450</td><td>Char *</td><td>Invocata in stampa documenti durante il calcolo delle scadenze se il documento in stampa ha indicata un codice di pagamento. TheObject è di classe (char *) e contiene la data calcolata che può essere sostituita se la routine ritorna FALSE. ATTENZIONE in caso di sostituzione la data deve essere indicata nel formato ANSI AAAAMMGG in modo che il numero di caratteri che costituiscono la data stessa sia esattamente 8.</td></tr>
<tr><td>451</td><td>Char *</td><td>Invocata in stampa documenti durante il calcolo degli importi delle rate di scadenza se il documento in stampa ha indicata un codice di pagamento. TheObject è di classe (char *) e contiene l’importo della rata calcolata che può essere sostituita se la routine ritorna FALSE. ATTENZIONE in caso di sostituzione la rata deve essere indicata con i decimali separati dal “.” (punto).</td></tr>
<tr><td>452</td><td>Char *</td><td>Identica all’uscita 451 tranne che per il fatto che importi sono IN VALUTA NON DI CONTO. Lire se la stampa è in euro e viceversa.</td></tr>
<tr><td>453</td><td>Char *</td><td>Identica all’uscita 451 tranne che per il fatto che importi sono IN VALUTA di stampa (Dollari,Yen,etc.etc.) se il documento non è in valuta gli importi passati sono nulli.</td></tr>
<tr><td>454</td><td>DPTeDoc</td><td>Revisione documenti (finestra di scelta del documento): invocata alla pressione del bottone “Stampa” (solo per l’opzione di stampa del documento) prima di eseguire qualsiasi operazione. TheObject è di tipo DPTeDoc, i suoi dati membro sono attualizzati con i dati della testata del documento.
Se la funzione ritorna il valore FALSE non viene eseguita la stampa del documento</td></tr>
</table>

455	**Char ***	**Pressione del tasto Stampa in anteprima di stampa. Viene restituito il nome completo di path del file temporaneo che contiene la stampa.**
456	**NULL**	**Immissione documenti (finestra di evasione righe ordini) invocata alla pressione del bottone ANNULLA.**
457	**NULL**	**Revisione documenti (finestra di evasione righe ordini) invocata alla pressione del bottone ANNULLA.**
458	**DPDist**	**Ristampa distinte attive e passive invocata alla pressione del tasto esegui nel caso di stampa di prova. Interviene alla fine del processo di stampa. TheObject è di classe DPDist distinta di pagamento.**
459	**DPDist**	**Ristampa distinte attive e passive invocata alla pressione del tasto esegui nel caso di stampa definitiva. Interviene alla fine del processo di stampa. TheObject è di classe DPDist distinta di pagamento.**
460	**DPTeDoc**	**Invocata in fase di revisione documenti durante il salvataggio. Consente di leggere il valore sul quale vengono calcolate le spese in percentuale. L'unico dato significativo dell'oggetto è TESDOCUM.ETOTIMPORT in cui è memorizzato il valore menzionato.**
461	**DPBaCod**	**Invocata in fase di cancellazione di un codice a barre. Consente di bloccare, ritornando il valore FALSE, l'eliminazione del codice.**
462	**DPStaEs**	**Invocata in fase di cancellazione di uno stato estero. Consente di bloccare, ritornando il valore FALSE, l'eliminazione dello stato.**
463	**DPTMarc**	**Invocata in fase di cancellazione di un marchio. Consente di bloccare, ritornando il valore FALSE, l'eliminazione del marchio.**
464	**DPVetto**	**Invocata in fase di cancellazione di un vettore. Consente di bloccare, ritornando il valore FALSE, l'eliminazione del vettore.**
465	**DPUMenu**	**Invocata in fase di creazione dei menù personalizzati per SPPFrame.dll. Se la funzione ritorna FALSE la voce di menù non viene inserita nel corrispondente sottomenù del menù principale.** **TheObject è di classe DPUMenu e contiene i dati delle voci di menù (l'uscita non viene emessa per i sottomenù del menù principale ma solo per le singole voci di menù).** **Per i dettagli sull'utilizzo della tabella UMENU si rimanda al manuale operativo della SPPFrame.** **N.B.: questa uscita è utilizzabile sono nel caso in cui sia presente la SPPFrame.dll.**
466	**DPAnMag**	**Navigatore Articoli: si invoca la funzione SIGLAPPSaveObject alla decodifica dell'articolo.** **TheObject è di classe DPAnMag e i suoi dati membro sono attualizzati con i valori dell'articolo selezionato.**
467	**Char ***	**Invocata durante la stampa documenti se è attiva l'opzione "Archiviazione st.doc. per Web++ / E-Ware++"** **in Configurazione / Applicazione / Magazzino / Anagraf.)**

<table>
<tr><td></td><td></td><td>TheObject è di classe (char *) e contiene il nome del file PDF memorizzato completo del percorso.</td></tr>
<tr><td>468</td><td>DPTaEle</td><td>Invocata dopo aver riempito la tabella ELECF e prima di produrre il file per l'invio telematico. Unico campo valorizzato della tabella è il campo ANNO con l'anno della dichiarazione degli allegati.</td></tr>
<tr><td>469</td><td>DPRicLot</td><td>Invocata in IMMISSIONE DOCUMENTI nella finestra di evasione delle righe ordine alla selezione di ogni riga ordine se e solo se la riga non ha un codice lotto attribuito.

Sono disponibili alla lettura:
• Codice Lotto Interno
• Codice Fornitore
• Codice Articolo
• Codice Lotto

I valori precedente sono Identificati dai seguenti dati membro della classe DPRicLot (si ricorda che gli identificativi sono da intendersi sensibili alle maiuscole/minuscole)
<table>
<tr><th>Id.Classe</th><th>Tipo</th><th>Lun</th><th>Id. Frame</th><th>Tipo Frame</th></tr>
<tr><td>codice</td><td>char</td><td>10</td><td>CODICE</td><td>string</td></tr>
<tr><td>fornitore</td><td>char</td><td>10</td><td>FORNITORE</td><td>string</td></tr>
<tr><td>articolo</td><td>char</td><td>30</td><td>ARTICOLO</td><td>string</td></tr>
<tr><td>lotto</td><td>char</td><td>20</td><td>LOTTO</td><td>string</td></tr>
</table>
Se la personalizzazione ritorna il valore FALSE è possibile ritornare all'applicazione valori personalizzati diversi da quelli letti.</td></tr>
<tr><td>470</td><td>DPRicLot</td><td>Invocata in REVISIONE DOCUMENTI nella finestra di evasione delle righe ordine alla selezione di ogni riga ordine se e solo se la riga non ha un codice lotto attribuito.

Sono disponibili alla lettura:
• Codice Lotto Interno
• Codice Fornitore
• Codice Articolo
• Codice Lotto

I valori precedente sono Identificati dai seguenti dati membro della classe DPRicLot (si ricorda che gli identificativi sono da intendersi sensibili alle maiuscole/minuscole)
<table>
<tr><th>Id.Classe</th><th>Tipo</th><th>Lun</th><th>Id. Frame</th><th>Tipo Frame</th></tr>
<tr><td>codice</td><td>char</td><td>10</td><td>CODICE</td><td>string</td></tr>
<tr><td>fornitore</td><td>char</td><td>10</td><td>FORNITORE</td><td>string</td></tr>
<tr><td>articolo</td><td>char</td><td>30</td><td>ARTICOLO</td><td>string</td></tr>
<tr><td>lotto</td><td>char</td><td>20</td><td>LOTTO</td><td>string</td></tr>
</table>
Se la personalizzazione ritorna il valore FALSE è possibile ritornare all'applicazione valori personalizzati diversi da quelli letti.</td></tr>
</table>

471	DPTeDoc	Invocata in fase di immissione documenti durante il salvataggio. Consente di leggere il valore del netto a pagare. L'unico dato significativo dell'oggetto è etotimporto in cui è memorizzato il valore menzionato.
472	char *	Viene invocata prima di lanciare la query di reperimento degli ordini nel navigatore articoli di SIGLA. L'oggetto passato contiene la clausola where della query.
473	char *	Viene invocata prima di lanciare la query di reperimento dei movimenti di carico scarico nel navigatore articoli di SIGLA. L'oggetto passato contiene la clausola where della query.
474	DPMovCo	Alla pressione del tasto registra nella gestione della prima nota contabile, consente ritornando il valore false di non lanciare la videata di gestione dei compensi a terzi. L'oggetto passato contiene come dato significativo il valore del campo numero della registrazione contabile. Questa uscita viene invocata solo nel caso di immissione di un documento abilitato per la gestione dei compensi a terzi.
475	DPMovMag	Alla pressione del tasto fido in immissione documenti, consente ritornando il valore false di non lanciare la finestra di visualizzazione dei dati del fido. L'oggetto passato contiene tutti i dati del documento di magazzino che si stà registrando immessi fino a quel momento.
476	DPMovMag	Alla pressione del tasto fido in revisione documenti, consente ritornando il valore false di non lanciare la finestra di visualizzazione dei dati del fido. L'oggetto passato contiene tutti i dati del documento di magazzino che si stà registrando immessi fino a quel momento.
477	char *	Permette di leggere da theObject la "WHERE" di selezione documenti che sarà usata dalla funzione di creazione fatture raggruppate.
478	char *	Durante la Stampa Fatture Riepilogative permette di leggere e cambiare la stringa del riferimento fattura.
479	char *	Durante la Stampa Fatture Riepilogative permette di leggere dalla testata dalla bolla i dati di destinazione merce(ragione sociale).
480	DPMovMat	Durante la Stampa Documenti / Stampa Fatture Riepilogative. Emessa solo è configurato il campo MATRICOLE CUMULATE permette di leggere le matricole collegate alla riga documento. L'uscita è ripetuta per ogni matricola legata alla riga documento.
481	DPMovMat	Durante la Stampa Documenti / Stampa Fatture Riepilogative. Emessa solo NON è configurato il campo MATRICOLE CUMULATE ed è configurato almeno uno dei campi relativi alle matricole. Permette di leggere la matricola collegata alla riga documento. L'uscita è messa una sola volta.
483	char *	Durante la Stampa Fatture Riepilogative permette di leggere dalla testata dalla bolla i dati di destinazione merce(indirizzo).
484	DPSalda	In fase di registrazione del pagamento nel modulo compensi a terzi consente, ritornando il valore FALSE, di bloccare la registrazione del pagamento in contabilità.
486	DPCashF *	Permette ad un programma di immettere flussi esterni a SIGLA nell'analisi dei flussi di cassa. L'uscita è eseguita alla pressione del tasto Esegui nella funzione Flussi Di Cassa e continua ad essere chiamata fintatochè il programma di personalizzazione ritorna FALSE. La Classe DPCashF (per SIGPPDLL) corrisponde alla tabella CASHFLOW (per SPPFRAME). Il flusso all'interno del Cash Flow è identificato come "Reg.Esterne" Le date devono essere specificate in formato ansi (AAAAMMGG) I dati numerici per SIGPPDLL sono double, per SPPFRAME sono stringhe PCHAR (in questo caso l'unico separatore dei decimali ammesso è il

punto ovvero il carattere ascii 46 “.”)
I codici che ammettono decodifica (causale contabile e sottoconto) possono essere anche diversi da quelli di SIGLA in quanto non è effettuato nessun controllo / decodifica.

Nome Campo (nome dato membro classe DPCashF)	Lun	Tipo	Descrizione
DATA (data)	8	Char	Data movimento di cash flow Formato ANSI AAAAMMGG OBBLIGATORIO.
SICUREZZA (sicurezza)	1	Char	Grado di attendibilità dell’uscita/entrata. 1) Contanti (Cassa valori assegni predatati, cedolini carte di credito, etc.) 2) Effetti/RID clienti portati in banca e non ancora esitati. Tasse da pagare (iva, ritenute d’acconto, irpef, inps,) 3) Debiti da/verso clienti (scadenziario clienti) 4) Debiti da/verso Fornitori OBBLIGATORIO.
DARE (dare)		Num	Importo movimento di entrata
AVERE (avere)		Num	Importo movimento di uscita
DESCRIZION (descrizion)	30	Char	Descrizione movimento
DESLINGUA (deslingua)	30	Char	Descrizione Aggiuntiva
SOTTOCONTO (sottoconto)	10	Char	Codice sottoconto a cui il movimento è riferito.
RAGIONESOC (ragionesoc)	50	Char	Descrizione sottoconto
NUMERODOC (numerodoc)	7	Char	Numero documento
DATADOC (datadoc)	8	Char	Data documento (formato ANSI AAAAMMGG)
TIPODOC (tipodoc)	2	Char	Tipo documento
NUMEPROT (numeroprot)	7	Char	Numero protocollo
DATAPROT (dataprot)	8	Char	Data protocollo
TIPOEFFETT (tipoeffett)	1	Char	Tipo scadenza. Vale "0" - rimessa diretta o contanti "1" - tratta "2" - ricevuta bancaria "3" - cessione credito "4" - pagherò

			"5" - lettera di credito "6" - tratta accettata "7" - RiBa "8" - ritardato pagamento "9" - cambiale "A" - altro pagamento "B" - bonifico bancario "F" – RID
C_F_A (c_f_a)	1	Char	Vale "C","F","A"
TIPOTASSA (tipotassa)	1	Char	"0" Nessuna Tassa "1" Versamento IVA "2" Versamento IRPEF "3" Versamento INPS "4" Versamento ENASARCO
CAUSALE (causale)	3	Char	Codice causale contabile
DCAUSALE (descausale)	40	Char	Descrizione causale
NUMERO	7	Char	Chiave registrazione esterna
RIGA	5	Char	Chiave registrazione esterna

DISPONIBILE SOLO PER VERSIONE 4

487 — **char ***

Raccoglie il DoubleClick fatta su una riga di dettaglio della grid dei flussi di cassa (cash flow). Se torna FALSE inibisce la visualizzazione dell'elemento SIGLA collegato alla registrazione di cash flow.
Il parametro passato è di classe char * e contiene il valore di campi :

- **Tipo Flusso**
- **Numero**
- **Riga**

della riga di Cash Flow cliccata. I campi sono separati da virgole nella forma :
T,NNNNNNN,RRRRR
Il codice Tipo Flusso vale

Tipo Flusso	Significato	Numero	Riga
1	Registrazioni Contabili	Numero in MOVCO	Riga in MOVCO
2	Registrazioni Periodiche	Primi sette caratteri del codice modello registrazione contabile.	Utimi 3 caratteri del codice modello registrazione contabile.
3	Registrazioni Previste	Numero in ALLERTE.	Vuoto.
4	Registrazioni Esterne	Valore inserito da una personalizzazione con l'uscita 486. Altrimenti progressivo della registrazione manuale inserita dall'utente nella finestra Flussi di Cassa.	Valore inserito da una personalizzazione con l'uscita 486. Altrimenti progressivo della registrazion

		<table><tr><td></td><td></td><td></td><td>e manuale inserita dall'utente nella finestra Flussi di Cassa.</td></tr><tr><td>5</td><td>Ordini Clienti</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr><tr><td>6</td><td>Bolle Clienti</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr><tr><td>7</td><td>Resi Clienti</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr><tr><td>8</td><td>Ordini Fornitori</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr><tr><td>9</td><td>Ciclo Passivo</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr><tr><td>A</td><td>Ft.Agenti</td><td>Numero in PROVAGEN.</td><td>Vuoto.</td></tr><tr><td>B</td><td>Interessi Clienti</td><td>Numero in MOVCO</td><td>Vuoto.</td></tr><tr><td>C</td><td>Preventivi Clienti</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr><tr><td>D</td><td>Preventivi Fornitori</td><td>Numero in TESDOCUM.</td><td>Vuoto.</td></tr></table> **DISPONIBILE SOLO PER VERSIONE 4**
488	**DPPerRc**	**Ritornando il valore FALSE permette nella fase di evasione ordini in immissione documenti di sostituire, solo nel caso di ricerca sul codice articolo, il codice di ricerca utilizzato sostituendolo con uno ritornato dalla personalizzazione.** **Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene il codice di ricerca immesso a video.** - **c_output (char [128]): contiene il valore di ritorno da sostituire al codice di ricerca immesso a video se il ritorno della funzione è FALSE.**
489	**DPPerRc**	**Ritornando il valore FALSE permette nella fase di evasione ordini in revisione documenti di sostituire, solo nel caso di ricerca sul codice articolo, il codice di ricerca utilizzato sostituendolo con uno ritornato dalla personalizzazione.** **Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:** - **c_input (char[128]): contiene il codice di ricerca immesso a video.** - **c_output (char [128]): contiene il valore di ritorno da sostituire al codice di ricerca immesso a video se il ritorno della funzione è FALSE.**
490	**Char ***	**Invocata in stampa documenti durante il calcolo degli importi delle rate di scadenza se il documento in stampa ha indicata un codice di pagamento.** **TheObject è di classe (char *) e contiene l'importo delle SPESE INCASSO legate al codice di pagamento. L'importo può essere sostituito se la routine ritorna FALSE.** **ATTENZIONE in caso di sostituzione la rata deve essere indicata con i decimali separati dal "." (punto).**
491	**Char ***	**Invocata, se è attiva la gestione grafica delle stampe, immediatamente prima di mostrare l'anteprima di stampa (o immediatamente prima**

		della stampa se non è configurata l'anteprima di stampa). TheObject è di tipo Char * e contiene il nome del file xml contenente il dataset della stampa completo del percorso. Il dataset contiene tutti i dati della stampa in oggetto e, nel caso della stampa dei documenti contiene i dati di tutti i documenti stampati. Il file è cancellato automaticamente dopo che è stato elaborato dalla procedura. Il nome del file inizia con "$S", se contiene i dati della stampa di un report, o "$X", nel caso in cui contenga i dati della stampa di una serie di documenti, l'estensione è ".tmp" (pur essendo comunque un file XML). Per una descrizione dello schema del file xml si veda l'apposita appendice di questo documento. DISPONIBILE SOLO PER VERSIONE 4
492	Char *	Invocata, se è attiva la gestione grafica delle stampe dei documenti e se è attiva l'archiviazione automatica delle stampe dei documenti nelle immagini aziendali o per interazione con web++, durante la fase di elaborazione dei documenti prima della uscita precedente (theActionID=491). TheObject è di tipo Char * e contiene il nome del file xml contenente il dataset del singolo documento (completo del percorso). Questa uscita viene invocata per ogni singolo documento interessato dalla sessione di stampa e soltanto se la stampa del documento è di tipo definitivo (cioè non si tratta di una stampa di prova). Il dataset contiene i dati di un solo documento. Il file è cancellato automaticamente dopo che il documento è stato archiviato in formato PDF dalla procedura. Il nome del file inizia con "$P" e l'estensione è ".tmp" (pur essendo comunque un file XML). Per una descrizione dello schema del file xml si veda l'apposita appendice di questo documento. Supponendo di eseguire una fatturazione riepilogativa non di prova che produca 3 fatture la sequenza delle uscite emesse sarà: 1. uscita 492 per i dati della sola fattura n.1 2. uscita 492 per i dati della sola fattura n.2 3. uscita 492 per i dati della sola fattura n.3 4. uscita 491 per i dati di tutte e tre le fatture Supponendo di eseguire una fatturazione riepilogativa di prova che produca 3 fatture la sequenza delle uscite emesse sarà: 1. uscita 491 per i dati di tutte e tre le fatture Prima di mostrare l'anteprima di stampa (o di eseguire la stampa se l'anteprima non è configurata) viene anche prodotto il file PDF di ogni singolo documento. Il nome completo del percorso di tale file è contenuto in un apposito elemento del file xml (per i dettagli si veda la descrizione dello schema in appendice a questo documento). DISPONIBILE SOLO PER VERSIONE 4
493	Char *	Invocata, se è attiva la gestione grafica delle stampe, alla pressione del bottone "Archivia" in anteprima di stampa (anteprima della stampa "grafica"). Ritornando il valore FALSE viene inibita l'archiviazione nel modulo immagini aziendali. TheObject è di tipo Char * e contiene il nome del file PDF della stampa completo del percorso, il titolo della stampa e il nome del file XML contenente il dataset del singolo documento completo del percorso (con una virgola per separare questi tre elementi). Il nome file xml contenente il dataset della stampa è sicuramente già stato comunicato dalla 491 che ha preceduto questa uscita. Attenzione quando viene emessa questa uscita (493) il file xml del dataset è già

		stato cancellato, quindi se lo si vuole utilizzare è necessario elaborarlo tramite l'uscita 491. Supponendo di eseguire la stampa del bilancio di verifica e che sia premuto il bottone "Archivia" in anteprima, la sequenza delle uscite emesse sarà: 1. uscita 491 che fornisce il file xml del dataset 2. uscita 493 che fornisce il nome del file PDF della stampa, il titolo della stampa e lo stesso nome del file xml del dataset DISPONIBILE SOLO PER VERSIONE 4
494	DPAnLot	Immissione/Revisione lotti in documenti di carico. Se la funzione torna FALSE consente, nella fase di generazione/modifica di un lotto, di impostare alcuni dati del lotto: data di creazione, data di scadenza e descrizioni libere (accertarsi di avere inizializzato tutte le 20 descrizioni libere). I dati impostati dalla personalizzazione saranno visualizzati e editabili nell'ambiente di gestione lotti. L'uscita è valida solo per documenti di carico: bolla carico o nota credito intestata ad un cliente.
495	DPUser	Invocata, se è attiva la gestione degli utenti, prima della registrazione dall'ambiente di gestione utenti. The Object è di tipo DPUser * e i dati membro contengono le informazioni dell'utente compresa la password dell'utente in forma criptata. Ritornando il valore FALSE viene inibita la registrazione. Questa uscita viene emessa sia operando dalla funzione di cambio password utente presente in SIGLA che dal menù Gestione Utenti del programma di Configurazione.
496	DPServiz	Invocata alla pressione del pulsante registra restituisce l'oggetto theServiz
497	DPIntra	Stampa modelli Intra (CONTEGGIO FRONTESPIZIO DI UNA RIGA CON IMPORTO A 0): se si restituisce FALSE le righe con importo a 0 vengono stampate; restituendo TRUE(default) le righe a 0 non vengono stampate.
498	DPIntrN	Stampa modelli Intra (stampa di una riga con importo a 0): se si restituisce FALSE le righe con importo a 0 vengono stampate; restituendo TRUE(default) le righe a 0 non vengono stampate.
499	DPIntrN	Stampa modelli Intra (MINIDISCO DI UNA RIGA CON IMPORTO A 0): se si restituisce FALSE le righe con importo a 0 vengono stampate; restituendo TRUE(default) le righe a 0 non vengono stampate.
501	DPTedoc	In fase di immissione da fonte dati esterna sulla revisione documenti consente di leggere i dati della testata del documento già inseriti, i dati significativi sono: magazzino principale, magazzino collegato, tipo protocollo, data protocollo, numero documento, data documento, tipo pagamento, sconto pagamento, sconto cliente/fornitore, cliente/fornitore. Se la personalizzazione ritorna il valore FALSE l'immissione da fonte dati esterna viene bloccata.
502	DPMovMag	Revisione documenti: finche' la funzione torna FALSE, i valori restituiti nell'oggetto DPMovMag opportunamente riempito da un programma esterno,vengono aggiunte alla List Box delle righe. Tale programma esterno non deve preoccuparsi di aggiornare le giacenze che vengono aggiornate automaticamente da SIGLA. I campi dell'oggetto DPMovMag vengono riempiti con i valori digitati a video sino a quel momento. La procedura si preoccupa anche di aggiungere la riga a Movimag senza bisogno che se ne occupi il programma esterno.
507	DPMovCo	In revisione Prima Nota Iva alla pressione del tasto scanner consente di bloccare l'avvio della funzione SCANNER ritornando il valore FALSE. Gli unici dati significativi sono NUMERO e FILE di Movco.
508	DPMovCo	In revisione Prima Nota Iva alla pressione del tasto viewer consente di bloccare l'avvio della funzione VIEWER ritornando il valore FALSE. Gli unici dati significativi sono NUMERO e FILE di Movco.

509	DPMovIva	Stampa liquidazione periodica IVA: invocata alla pressione del tasto "Stampa" (solo per liquidazione periodica). TheObject è di tipo DPMovIva e gli unici dati membro significativi sono: • MESECOMPET (mese o trimestre di stampa); • ANNOCOMPET (anno di stampa); • DATAREG (eventuale data registrazione finale); • STLIQ_SN ("N" per stampa di prova, "S" per stampa definitiva). Se la funzione torna FALSE la stampa è interdetta.
510	DPMovCo	Stampa giornale contabile: invocata alla pressione del tasto "Stampa". TheObject è di tipo DPMovCo e gli unici dati membro significativi sono: • DATAINIZIO (data di stampa iniziale); • DATAFINE (data di stampa finale); • STAMPAT_SN ("N" per stampa di prova, "S" per stampa definitiva). Se la funzione torna FALSE la stampa è interdetta.
511	DPMovMa	Immissione documenti: in fase di evasione dell'ordine, sia parziale che totale, viene consentita la variazione dei dati ricavati dal documento evaso in particolare del prezzo.

512	DPPerc	E' invocata in prima nota generale alla pressione del tasto Aggiungi. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono:
		• c_input (char[128]): contiene la descrizione della causale contabile.
		• c_output (char [128]): contiene il valore della descrizione aggiuntiva a video. Se la funzione ritorna FALSE è usato per sostituire il valore la descrizione aggiuntiva a video.
		• c_tipo (double): vale 0 per l'immissione di un nuovo movimento contabile; vale 1 in revisione di un movimento già immesso.
		Se la funzione ritorna FALSE il campo c_output contiene il valore da attribuire alla descrizione aggiuntiva a video.
		
		Nell'uso da SPPFRAME i nomi campo con cui effettuare la GetField sono:
		• C_INPUT
		• C_OUTPUT
		Esempio:
		// …
		#include "DPPerRc.h"
		// …
		BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
		{
		DPPerRc * thePerRc;
		char buffer[128];
		switch(theActionID) {

		case 512:
		thePerRc = (DPPerRc *) theObject;
		sprintf(buffer,"In:%s Out:%s Tipo:%i",thePerRc->c_input,thePerRc->c_output,(int) thePerRc->c_tipo);
		MessageBox(NULL,buffer,"SIGPPDLL",MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
		if((int) thePerRc->c_tipo==0) {
		strcpy(thePerRc->c_output,thePerRc->c_input);
		return(FALSE);
		}
		break;
		}
		return TRUE;
		}

513	DPTeDoc	Pressione del tasto Registra/Salva in immissione documenti Permette di leggere e, se la funzione torna FALSE, di sostituire i dati di testata e piede a video.
514	DPTeDoc	Pressione del tasto Registra/Salva in revisione documenti. Permette di leggere e, se la funzione torna FALSE, di sostituire i dati di testata e piede a video.
517	DPMo770	IMMISSIONE/REVISIONE COMPENSI A TERZI: Viene invocata alla pressione del pulsante AGGIUNGI. TheObject è di tipo DPMo770, unici dati validi EIRPEF e EIMPONIRPE. Ritornanto FALSE l'operazione è interdetta.
518	DPMovCo	CICLO PASSIVO: invocata alla pressione del bottone registra e contabilizza prima di procedere nella contabilizzazione. TheObject è di tipo DPMovCo, i dati membro significativi sono: • DATAREG • DATACOMPET, ESECOMPET • DATAPROTOC • DATADOCUM, NUMDOCUM • SOTTOCONTO • CAUSALE, TIPODOCIVA • CODPAGAM • AGENTE, CAPOZONA e contengono i valori inseriti a video. Se la funzione torna FALSE l'operazione è interdetta.
519	DPMovCo	IMMISSIONE/REVISIONE PRIMA NOTA IVA: viene invocata alla pressione del pulsante SALVA in fase di registrazione di una fattura extracee. TheObject è di tipo DPMovCo, i dati membro significativi sono: • NUMERO (in revisione) • DATAREG

		• DATACOMPET, ESECOMPET • SOTTOCONTO, C_F • EIMPORTO, EIMPFATTUR • CODVALUTA, IMPVALUTA (se in valuta estera) • CAUSALE • CODPAGAM • AGENTE, CAPOZONA • UTENTE, ULT_AGG Ritornanto FALSE l'operazione è interdetta.
520	char *	REVISIONE PRIMA NOTA IVA: è invocata alla pressione del bottone Aggiungi nella pagina Scadenz. se la scadenza è già inserita in una distinta. TheObject è di tipo char * e contiene il valore del campo NUMERO di MOVCO (corrispondente alla fattura). Ritornando FALSE l'operazione è interdetta.
521	DPTeDoc	In immissione documenti dopo salvataggio (con o senza stampa) permette di impostare a video i dati di testata/piede.
522	DPTeDoc	In immissione documenti dopo salvataggio (con o senza stampa) permette di impostare a video i dati di Righe1/Righe2 testata/piede. Lo scopo dell'uscita è posizionarsi dopo il salvataggio in Righe1 piuttosto che rimanere in Testata1.
523	DPPerRc	REVISIONE DOCUMENTI: ritornando il valore FALSE permette nella fase di riempimento della listbox revisione documenti di aggiungere, informazioni personalizzate in coda. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene una stringa così composta campo numero di tesdocum primi 7 caratteri, campo annotazioni di Tesdocum altri 60 caratteri. - c_output (char [128]): contiene le informazioni da visualizzare.
524	DPPerRc	IMMISSIONE DOCUMENTI: ritornando il valore FALSE permette, nella fase di riempimento della listbox di selezione degli ordini da evadere, di visualizzare informazioni personalizzate in coda. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il campo numero di tesdocum. - c_output (char [128]): contiene le informazioni da visualizzare vengono visualizzati solo i primi 25 caratteri nel caso che venga richiesto l'ordinamento standard degli ordini, 40 nel caso che venga richiesto l'ordinamento per indirizzo di spedizione.
525	DPPerRc	SELEZIONE ORDINI IN REV. DOC.: ritornando il valore FALSE permette, nella fase di riempimento della listbox di selezione degli ordini da evadere in revisione documenti, aggiungere informazioni personalizzate in coda. Il parametro theObject è di tipo DPPerRc, i dati membro utilizzabili sono: - c_input (char[128]): contiene il campo numero di tesdocum. - c_output (char [128]): contiene le informazioni da visualizzare vengono visualizzati solo i primi 25 caratteri nel caso che venga richiesto l'ordinamento standard degli ordini, 40 nel caso che

		venga richiesto l'ordinamento per indirizzo di spedizione.
526	DPMovIva	In Immissione/Revisione prima nota IVA invocata alla pressione del bottone Aggiungi del folder IVA. TheObject è di tipo DPMovIva, i dati membro significativi sono: • NUMERO (solo in revisione) • DATAREG • REGISTRO • CODIVA • EIMPONIBIL, EIMPOSTA (operando in euro) • VIMPONIBIL, VIMPOSTA (operando in valuta) • SOTTOCONTO • C_F • UTENTE • ULT_AGG e contengono i valori inseriti a video. Se la funzione torna FALSE l'operazione è interdetta.
527	DPMovIva	Stampa liquidazione periodica IVA: invocata alla fine della stampa (solo per liquidazione periodica). TheObject è di tipo DPMovIva e gli unici dati membro significativi sono: • MESECOMPET (mese o trimestre di stampa); • ANNOCOMPET (anno di stampa); • DATAREG (eventuale data registrazione finale); • STLIQ_SN ("N" per stampa di prova, "S" per stampa definitiva).
528	char *	REVISIONE PRIMA NOTA IVA: viene invocata alla pressione del bottone Rimuovi nella pagina Scadenz. se la scadenza è già inserita in una distinta. TheObject è di tipo char * e contiene il valore del campo NUMERO di MOVCO (corrispondente alla fattura). Se la funzione torna FALSE l'operazione è interdetta.
530	Char *	in STAMPA SOLLECITI INSOLUTI permette di leggere il nome del file INF di deltacom. Il nome contiene anche il percoso (cartella di polling).
532	Char *	in STAMPA LETTER SOLLECITO SCADENZE permette di leggere il nome del file INF di deltacom. Il nome contiene anche il percoso (cartella di polling).
533	Char *	in STAMPA SCHEDE CONTABILI (Opzione "Stampa lettere a cliente") permette di leggere il nome del file INF di deltacom. Il nome contiene anche il percoso (cartella di polling).
534	DPMovCo	CICLO PASSIVO: invocata alla pressione del bottone Registra e Contabilizza prima di procedere nella contabilizzazione. TheObject è di tipo DPMovCo, i dati membro significativi sono: • DATAREG • DATACOMPET, ESECOMPET • DATAPROTOC • DATADOCUM, NUMDOCUM • SOTTOCONTO • CAUSALE, TIPODOCIVA • CODPAGAM • AGENTE, CAPOZONA • EIMPFATTUR (se la registrazione è in euro) o IMPVALUTA e CODVALUTA se la registrazione è in valuta estera

		e contengono i valori inseriti a video. Il totale della registrazione è contenuto in EIMPFATTUR se la registrazione è eseguita in euro, mentre se la registrazione è eseguita in valuta estera è contenuto in IMPVALUTA (ovviamente in questo caso CODVALUTA contiene il codice della valuta estera). **Se la funzione torna FALSE il codice causale contabile e il codice del tipo pagamento saranno sostituiti con quelli modificati dalla personalizzazione. Dopodiché la procedura procede con tutti i controlli e se non ci sono errori con la contabilizzazione (l'uscita viene emessa come prima operazione subito dopo la pressione del bottone Registra&Contabilizza).**
535	**DPMovMag**	**GENERAZIONE ORDINI FORNITORE: invocata durante il salvataggio degli ordini, permette di variare i dati delle singole righe d'ordine generate in automatico dalla procedura.**
536	**DPIntra**	**In stampa definitiva documenti fiscali (fatture/fatture riepilogative/note credito etc.) dopo l'inserimento di ogni movimento intra. TheObject è di classe DPintra e i dati sono in solo lettura. Per la modifica è necessario aggiornare il movimento utilizzando la chiave univoca NUMERO (v.tecnotes.chm).**

SIGLAPPDeleteObject

SCOPO: SIGLA chiama questa funzione in determinati punti dell'applicazione DOPO aver proceduto alla cancellazione di un dato. L'identificatore theActionID consente di stabilire quale procedura di SIGLA ha invocato la funzione mentre theObject fornisce le informazioni relative ai dati da salvare.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPDeleteObject(DPObject *theObject**,int** theActionID**)**

Il puntatore theObject deve essere castato sull'oggetto appropriato (vedere le macro SPPSAVE_ che mappano alcuni valori possibili per theActionID).

Gli header files per effettuare il cast dell'oggetto passato come parametro sono distribuiti con il sistema di sviluppo di SIGLA. Poiché i nomi delle variabili membro delle classi di oggetti di SIGLA mappano, nella maggior parte dei casi, i nomi dei campi delle tabelle del database la ricerca fra i dati membro delle informazioni necessarie è piuttosto semplice.

La tabella che mappa i valori possibili per il parametro theActionID e riporta le specifiche dell'oggetto theObject il cui reference viene passato è la stessa già dettagliata per la funzione SIGLAPPSaveObject().

SIGLAPPSetSconti

SCOPO: SIGLA chiama questa funzione durante l'immissione/revisione dei documenti di magazzino (ordini, bolle, fatture etc.) immediatamente PRIMA di valutare secondo il criterio standard gli sconti e le maggiorazioni da applicare sulla riga. Se la funzione torna FALSE la procedura si incarica di determinare gli sconti/maggiorazioni, se torna TRUE i valori impostati nei parametri sc*, mg*vengono assunti come default e quindi presentati a video.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPSetSconti(fDPObject * theAnamag**, DPObject *** theCliFor**, DPObject *** theMovMag**, int** theTipoDoc**, double far *** sc1**, double far *** sc2**, double far *** sc3**, double far *** sc4**, double far *** sc5**, double far *** mg1**, double far *** mg2**)**

Il parametro theTipoDoc consente di individuare il tipo effettivo del documento elaborato (vedere le macro SPPTIDOC_* nel file SIGPPDLL.H).

Il significato dei parametri che vengono passati alla funzione è il seguente:

* *theAnamag* puntatore ad un oggetto DPAnMag (la cui classe è definita nell'header file DPAnMag.h). L'oggetto contiene, nei suoi dati membro, le informazioni relative all'articolo di magazzino che si sta trattando (codice articolo etc.)
* *theCliFor* puntatore ad un oggetto DPTaClF (la cui classe è definita nell'header DPTaClF.h). L'oggetto contiene, nei suoi dati membro, le informazioni relative al cliente/fornitore su cui si sta operando.
* *theMovMag* puntatore ad un oggetto DPMovMag (la cui classe è definita nell'header DPMovMag.h). L'oggetto contiene, nei suoi dati membro, tutte le informazioni presenti sulla riga di documento attualmente trattata (ma non ancora inserita nel database).
* *theTipoDoc* intero che individua il tipo effettivo di documento su cui si sta operando (vedere le macro SPPTIDOC_* nel file SIGPPDLL.H).
* *sc1,sc2,sc3,sc4,sc5* puntatori a variabili double nelle quali devono essere tornati gli sconti riga da applicare
* *mg1,mg2* puntatori a variabili double nelle quali devono essere tornate le maggiorazioni riga da applicare

Valore theTipoDoc	Descrizione
TheTipoDoc=-1	**Se esco dallo sconto 1 in immissione doc., in questo caso sc1 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto1.**
TheTipoDoc=-2	**Se esco dallo sconto 2 in immissione doc., in questo caso sc2 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto2.**
TheTipoDoc=-3	**Se esco dallo sconto 3 in immissione doc., in questo caso sc3 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto3.**
TheTipoDoc=-4	**Se esco dallo sconto 4 in immissione doc., in questo caso sc4 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto4.**
TheTipoDoc=-5	**Se esco dallo sconto 5 in immissione doc., in questo caso sc5 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto5.**
TheTipoDoc=-6	**Se esco dalla maggiorazione 1 in immissione doc., in questo caso mg1 contiene in ingresso la maggiorazione contenuta a video e in uscita deve eventualmente contenere la maggiorazione che la personalizzazione intende riscrivere nel campo maggiorazione1.**
TheTipoDoc=-7	**Se esco dalla maggiorazione 2 in immissione doc., in questo caso mg2 contiene in ingresso la maggiorazione contenuta a video e in uscita deve eventualmente contenere la maggiorazione che la personalizzazione intende riscrivere nel campo maggiorazione2.**

TheTipoDoc=-8	**Se esco dallo sconto 1 in revisione doc., in questo caso sc1 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto1.**
TheTipoDoc=-9	**Se esco dallo sconto 2 in revisione doc., in questo caso sc2 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto2.**
TheTipoDoc=-10	**Se esco dallo sconto 3 in revisione doc., in questo caso sc3 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto3.**
TheTipoDoc=-11	**Se esco dallo sconto 4 in revisione doc., in questo caso sc4 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto4.**
TheTipoDoc=-12	**Se esco dallo sconto 5 in revisione doc., in questo caso sc5 contiene in ingresso lo sconto contenuto a video e in uscita deve eventualmente contenere lo sconto che la personalizzazione intende riscrivere nel campo sconto5.**
TheTipoDoc=-13	**Se esco dalla maggiorazione 1 in revisione doc., in questo caso mg1 contiene in ingresso la maggiorazione contenuta a video e in uscita deve eventualmente contenere la maggiorazione che la personalizzazione intende riscrivere nel campo maggiorazione1.**
TheTipoDoc=-14	**Se esco dalla maggiorazione 2 in revisione doc., in questo caso mg2 contiene in ingresso la maggiorazione contenuta a video e in uscita deve eventualmente contenere la maggiorazione che la personalizzazione intende riscrivere nel campo maggiorazione2.**

L'esempio che segue mostra come si debba modificare la funzione per determinare gli sconti in modo alternativo rispetto allo standard:

```
BOOL DLLCALL SIGLAPPSetSconti(DPObject *theAnamag, DPObject *theCliFor, DPObject *theMovMag, int theTipoDoc, double far *sc1, double far *sc2, double far *sc3, double far *sc4, double far *sc5, double far *mg1, double far *mg2)
{
        *sc1=6.5;
        *sc2=10.9;
        *sc3=0.;
        *sc4=0.;
        *sc5=0.;
        *mg1=0.;
        *mg2=0.;
        return TRUE;
}
```

SIGLAPPSetProvvigioni

SCOPO: SIGLA chiama questa funzione durante l'immissione/revisione dei documenti di magazzino (ordini,bolle,fatture etc.) immediatamente PRIMA di valutare secondo il criterio standard le provvigioni agente/capozona da applicare sulla riga. Se la funzione torna FALSE la procedura si incarica di determinare le provvigioni, se torna TRUE i valori impostati nei parametri pragente e prcapozona vengono assunti come default e quindi presentati a video.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPSetProvvigioni(DPObject ***theAnamag, DPObject ***theCliFor**,**
DPObject ***theMovMag, int** theTipoDoc**, double far *** pragente**, double far *** prcapozona**)**

Il parametro theTipoDoc consente di individuare il tipo effettivo del documento elaborato (vedere le macro SPPTIDOC_* nel file SIGPPDLL.H).

Il significato dei parametri che vengono passati alla funzione è il seguente:

* *theAnamag* puntatore ad un oggetto DPAnMag (la cui classe è definita nell'header file DPAnMag.h). L'oggetto contiene, nei suoi dati membro, le informazioni relative all'articolo di magazzino che si sta trattando (codice articolo etc.)
* *theCliFor* puntatore ad un oggetto DPTaClF (la cui classe è definita nell'header DPTaClF.h). L'oggetto contiene, nei suoi dati membro, le informazioni relative al cliente/fornitore su cui si sta operando.
* *theMovMag* puntatore ad un oggetto DPMovMag (la cui classe è definita nell'header DPMovMag.h). L'oggetto contiene, nei suoi dati membro, tutte le informazioni presenti sulla riga di documento attualmente trattata (ma non ancora inserita nel database).
* *theTipoDoc* intero che individua il tipo effettivo di documento su cui si sta operando (vedere le macro SPPTIDOC_* nel file SIGPPDLL.H).
* *pragente,prcapozona* puntatori a variabili double nelle quali devono essere tornate le provvigioni da proporre a video all'utente

SIGLAPPSetPrezzo

SCOPO: SIGLA chiama questa funzione durante l'immissione/revisione dei documenti di magazzino (ordini, bolle, fatture etc.) immediatamente PRIMA di valutare secondo il criterio standard il prezzo da applicare sulla riga. Se la funzione torna FALSE la procedura si incarica di determinare il prezzo, se torna TRUE il valore impostato nel parametro thePrezzo viene assunto come default e quindi presentato a video.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPSetPrezzo(DPObject ***theAnamag, DPObject ***theCliFor**,**
DPObject ***theMovMag, int** theTipoDoc**, double far *** thePrezzo**)**

Il parametro theTipoDoc consente di individuare il tipo effettivo del documento elaborato (vedere le macro SPPTIDOC_* nel file SIGPPDLL.H).

Il significato dei parametri che vengono passati alla funzione è il seguente:

* *theAnamag* puntatore ad un oggetto DPAnMag (la cui classe è definita nell'header file DPAnMag.h). L'oggetto contiene, nei suoi dati membro, le informazioni relative all'articolo di magazzino che si sta trattando (codice articolo etc.)
* *theCliFor* puntatore ad un oggetto DPTaClF (la cui classe è definita nell'header DPTaClF.h). L'oggetto contiene, nei suoi dati membro, le informazioni relative al cliente/fornitore su cui si sta operando.
* *theMovMag* puntatore ad un oggetto DPMovMag (la cui classe è definita nell'header DPMovMag.h). L'oggetto contiene, nei suoi dati membro, tutte le informazioni presenti sulla riga di documento attualmente trattata (ma non ancora inserita nel database).
* *theTipoDoc* intero che individua il tipo effettivo di documento su cui si sta operando (vedere le macro SPPTIDOC_* nel file SIGPPDLL.H).

* *thePrezzo* puntatore a variabile double nella quale deve essere tornato il prezzo da proporre a video all'utente

Questa funzione viene chiamata durante l'immissione/revisione dei documenti di magazzino (ordini, bolle, fatture etc.) anche all'uscita del campo video 'codice articolo'. In questo caso il parametro theTipoDoc ha valore -1 e se la funzione torna TRUE il valore contenuto nel parametro thePrezzo viene assegnato al campo video quantità.

Questa funzione viene chiamata durante l'immissione/revisione dei documenti di magazzino (ordini, bolle, fatture etc.) dalla finestra di esplosione righe ordini dopo che l'utente ha eseguito un click sulla riga d'interesse. In questo caso il parametro theTipoDoc ha valore -2 e se la funzione torna TRUE il valore contenuto nel parametro thePrezzo viene assegnato al campo video quantità della finestra esplosione ordini. Il parametro theMovMag mappa la riga di movimento di magazzino (riga dell'ordine che si sta importando). I parametri theAnmag e theCliFor non sono passati. La riga ordine non è ancora stata importata sul documento.

Questa funzione viene chiamata durante l'immissione/revisione dei documenti di magazzino (ordini, bolle, fatture etc.) quando l'utente esegue un doppio click su una riga già inserita. In questo caso il parametro theTipoDoc ha valore -3 e se la funzione torna TRUE il valore contenuto nel parametro thePrezzo viene assegnato al campo video quantità. Il parametro theMovMag mappa la riga di movimento di magazzino.

SIGLAPPExecuteSQL

SCOPO: consente l'esecuzione immediata di uno statement SQL utilizzando la stessa connessione che usa SIGLA.

La sintassi della funzione è la seguente:

BOOL DLLCALL SIGLAPPExecuteSQL(DPDbase *theDB**, LPSTR** SQLStm**)**

N.B.: la funzione non deve essere modificata ma viene fornita per essere usata così com'è.

Per chiamare la funzione è necessario impostare i parametri richiesti che sono:

* *theDB* puntatore ad un oggetto DPDbase: deve essere usato uno dei due puntatori che sono passati dalla funzione SIGLAPPInit() che quindi dovranno essere salvati in variabili globali.
* *SQLStm* puntatore ad un array di caratteri null-terminated contenente lo statement SQL da lanciare in esecuzione

Ad esempio, volendo azzerare il campo 'Prezzo Standard' nella tabella anagrafica di magazzino si dovrà chiamare la funzione nel seguente modo:

SIGLAPPExecuteSQL(Ditta,"UPDATE ANAMAGA SET PRSTANDARD=0");

Personalizzazione delle finestre

Le versioni Win32 (Windows95/NT) di SIGLA/START consentono di operare l'aggiunta di campi di input su alcune delle finestre gestite. I campi di input aggiuntivi possono essere *normali* (la gestione della memorizzazione e del recupero dal database sono in questo caso demandate al personalizzatore che dovrà operare una reimplementazione della funzione SIGLAPPSaveObject contenuta nella DLL SIGPPDLL) o *automatici.*

I campi automatici consentono di operare delle aggiunte di informazione alle finestre di SIGLA/START che vengono automaticamente memorizzate e recuperate nelle tabelle del database.

L'operazione avviene definendo in un file ASCII, il cui nome varia in funzione della finestra che si desidera personalizzare, un insieme di istruzioni in grado di descrivere i campi da aggiungere. La funzione SIGLAPPSaveObject contenuta nella libreria a link dinamico SIGPPDLL consente, se opportunamente reimplementata, di effettuare controlli sui campi aggiunti e/o di acquisirne/modificarne il valore a video.

Definizione dei campi aggiuntivi

Per le finestre per le quali è attiva la possibilità di gestire campi aggiuntivi è necessario creare, se si intende sfruttare questa funzionalità, un file ASCII contenente le specifiche dei campi da aggiungere. Il nome del file è definito nella tabella riportata successivamente ed è univoco per ogni finestra. Per tutti i files il postfisso da usare è **.USR**. I files **devono** essere posizionati nell'indirizzario contenente le procedure eseguibili di SIGLA/START.

Sintassi dei files di personalizzazione

I files di descrizione dei campi aggiuntivi sono composti da una serie di righe ognuna delle quali descrive un oggetto da aggiungere alla finestra. La struttura della riga è **posizionale** e le posizioni devono essere rigidamente rispettate. Il primo carattere della riga stabilisce il tipo di oggetto da aggiungere mentre la restante parte della riga riporta i parametri necessari per la definizione del campo stesso. **Tutte le coordinate necessarie per il posizionamento di un oggetto video sono espresse in caratteri** (a.e. la coordinata orizzontale 10 rappresenta il carattere a colonna 10). Le coordinate dell'angolo alto sinistro della finestra sono x=00,y=00 (intendendo con x l'asse orizzontale e con y l'asse verticale). Il file può contenere delle righe di commento che devono iniziare con il carattere <**;>** (punto e virgola)

```
; Questo è un commento
;
; Ogni riga di commento può contenere qualunque cosa
```

Definizione della label del folder aggiuntivo

Tutti i campi aggiuntivi devono essere contenuti all'interno di un folder che va ad aggiungersi a quelli normalmente gestiti nella finestra su cui si sta operando. **La prima istruzione diversa da un commento che deve essere riportata all'interno del file deve essere quella necessaria ad impostare il titolo del folder aggiuntivo.**

La riga di comando per definire la label del folder aggiuntivo e':

Ldescrizione

dove:

- **L** in prima posizione individua il tipo di azione da intraprendere (definizione label folder aggiuntivo)
- **descrizione** è il testo della label che individua il folder. **N.B.: il testo non deve superare gli 8 caratteri di ampiezza**

```
; definizione della label del Folder Aggiuntivo
LNote
```

Disegno di un riquadro nel folder aggiuntivo

E' possibile procedere alla definizione di un contorno 3D all'interno del folder aggiuntivo.

La riga di comando per definire un contorno e':

AxxyyXXYY

dove:

- **A** in prima posizione individua il tipo di azione da intraprendere (disegno di un riquadro nel folder aggiuntivo)
- **xx,yy** rappresentano le coordinate in caratteri dell'angolo alto sinistro del riquadro
- **XX,YY** rappresentano le coordinate in caratteri dell'angolo basso destro del riquadro

```
; definizione della label del Folder Aggiuntivo
LNote
; definizione di un riquadro nel Folder Aggiuntivo
A02025009
```

N.B.: tutti i riquadri devono essere definiti DOPO la definizione della label del folder aggiuntivo e PRIMA della definizione degli altri oggetti video.

Definizione di una didascalia

E' possibile procedere alla definizione di una didascalia all'interno del folder aggiuntivo.

La riga di comando per definire una didascalia e':

TxxyyTesto

dove:

- **T** in prima posizione individua il tipo di azione da intraprendere (definizione di una didascalia nel folder aggiuntivo)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte della didascalia
- **Testo** è la stringa di caratteri che deve comparire sul video come didascalia

```
; definizione della label del Folder Aggiuntivo
LNote
; definizione di un riquadro nel Folder Aggiuntivo
A02025009
; definzione delle didascalie
T0303Data
T0305Importo
T0307Descrizione
```

Definizione di un campo d'immissione numerico abilitato all'input

La riga di comando per definire un campo d'immissione numerico e':

INlldxxyy

dove:

- **I** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)

- **N** in seconda posizione individua il tipo di campo d'immissione (campo numerico abilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (compreso il punto decimale e la parte decimale)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (massimo 5)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

IN1322005

Definizione di un campo d'immissione numerico disabilitato all'input

La riga di comando per definire un campo d'immissione numerico disabilitato e':

Inlldxxyy

dove:

- **I** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)
- **n** in seconda posizione individua il tipo di campo d'immissione (campo numerico disabilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (compreso il punto decimale e la parte decimale)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (massimo 5)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

In1322005

Definizione di un campo d'immissione data abilitato all'input

La riga di comando per definire un campo d'immissione data e':

IDlldxxyy

dove:

- **I** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)
- **D** in seconda posizione individua il tipo di campo d'immissione (campo data abilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (per i campi data deve essere 08)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (per i campi data deve essere 0)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

ID0802003

Definizione di un campo d'immissione data disabilitato all'input

La riga di comando per definire un campo d'immissione data disabilitato e':

Idlldxxyy

dove:

- **I** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)
- **d** in seconda posizione individua il tipo di campo d'immissione (campo data disabilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (per i campi data deve essere 08)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (per i campi data deve essere 0)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

Id0802003

Definizione di un campo d'immissione testo abilitato all'input

La riga di comando per definire un campo d'immissione testo e':

ITIIdxxyyppp…p

dove:

- **I** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)
- **T** in seconda posizione individua il tipo di campo d'immissione (campo testo abilitato all'input)
- **II** rappresenta, su due digit, l'ampiezza del campo (per i campi testo deve essere minore o uguale a 60)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (per i campi testo deve essere 0)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione
- **ppp…p** è una stringa di caratteri che viene usata come picture di lettura (obbligatoria) che deve avere la stessa lunghezza del campo

Le picture per i campi testo possono essere una combinazione dei seguenti caratteri:

- **X** carattere alfanumerico case sensitive
- **!** carattere alfanumerico convertito in maiuscolo
- **A** carattere alfabetico
- **9** carattere numerico

La picture di lettura per un campo testo di 16 caratteri destinato a contenere un codice fiscale potrebbe essere impostato come:

!!!!!!99!99!999!

o

XXXXXX99X99X999X

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

IT2002007XXXXXXXXXXXXXXXXXXX

Definizione di un campo d'immissione testo disabilitato all'input

La riga di comando per definire un campo d'immissione testo disabilitato e':

ltlldxxyyppp…p

dove:

- **l** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)
- **t** in seconda posizione individua il tipo di campo d'immissione (campo testo disabilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (per i campi testo deve essere minore o uguale a 60)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (per i campi testo deve essere 0)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione
- **ppp…p** è una stringa di caratteri che viene usata come picture di lettura (obbligatoria) che deve avere la stessa lunghezza del campo

Le picture per i campi testo possono essere una combinazione dei seguenti caratteri:

- **X** carattere alfanumerico case sensitive
- **!** carattere alfanumerico convertito in maiuscolo
- **A** carattere alfabetico
- **9** carattere numerico

La picture di lettura per un campo testo di 16 caratteri destinato a contenere un codice fiscale potrebbe essere impostato come:

!!!!!!99!99!999!

o

XXXXXX99X99X999X

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

It2002007XXXXXXXXXXXXXXXXXXX

Definizione di un campo d'immissione numerico automatico

La riga di comando per definire un campo d'immissione numerico automatico e':

DNlldxxyytfffffffffff

dove:

- **D** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione automatico nel folder aggiuntivo)
- **N o n** in seconda posizione individua il tipo di campo d'immissione (campo numerico abilitato/diabilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (compreso il punto decimale e la parte decimale)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (massimo 5)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione
- **t** rappresenta su un digit il numero della tabella del database in cui è posizionato il campo da gestire. Il numero della tabella è un intero di un byte il cui valore minimo è 1. I numeri possibili variano da finestra a finestra e sono mappati nella tabella che segue
- **fffffffffff** stringa di 11 caratteri che individua il nome del campo da aggiornare nella tabella definita al punto precedente

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

DN13220051USR0001

Definizione di un campo d'immissione data automatico

La riga di comando per definire un campo d'immissione data automatico e':

DDlldxxyytfffffffffff

dove:

- **D** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione automatico nel folder aggiuntivo)
- **D o d** in seconda posizione individua il tipo di campo d'immissione (campo data abilitato/disabilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (per i campi data deve essere 08)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (per i campi data deve essere 0)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione
- **t** rappresenta su un digit il numero della tabella del database in cui è posizionato il campo da gestire. Il numero della tabella è un intero di un byte il cui valore minimo è 1. I numeri possibili variano da finestra a finestra e sono mappati nella tabella che segue
- **fffffffffff** stringa di 11 caratteri che individua il nome del campo da aggiornare nella tabella definita al punto precedente

; definizione della label del Folder Aggiuntivo

LNote

; definizione di un riquadro nel Folder Aggiuntivo

A02025009

; definzione delle didascalie

T0303Data

T0305Importo

T0307Descrizione

; definizione dei campi d'immissione

DD08020031USR0002

Definizione di un campo d'immissione testo automatico

La riga di comando per definire un campo d'immissione testo automatico e':

DTlldxxyytfffffffffffppp…p

dove:

- **D** in prima posizione individua il tipo di azione da intraprendere (definizione di un campo d'immissione nel folder aggiuntivo)
- **T o t** in seconda posizione individua il tipo di campo d'immissione (campo testo abilitato/disabilitato all'input)
- **ll** rappresenta, su due digit, l'ampiezza del campo (per i campi testo deve essere minore o uguale a 60)
- **d** rappresenta il numero di digit decimali da gestire dopo la virgola (per i campi testo deve essere 0)
- **xx,yy** rappresentano le coordinate in caratteri del primo byte del campo d'immissione
- **t** rappresenta su un digit il numero della tabella del database in cui è posizionato il campo da gestire. Il numero della tabella è un intero di un byte il cui valore minimo è 1. I numeri possibili variano da finestra a finestra e sono mappati nella tabella che segue

- **fffffffffff** stringa di 11 caratteri che individua il nome del campo da aggiornare nella tabella definita al punto precedente
- **ppp…p** è una stringa di caratteri che viene usata come picture di lettura (obbligatoria) che deve avere la stessa lunghezza del campo

Le picture per i campi testo possono essere una combinazione dei seguenti caratteri:

- **X** carattere alfanumerico case sensitive
- **!** carattere alfanumerico convertito in maiuscolo
- **A** carattere alfabetico
- **9** carattere numerico

La picture di lettura per un campo testo di 16 caratteri destinato a contenere un codice fiscale potrebbe essere impostato come:

!!!!!!99!99!999!

o

XXXXXX99X99X999X

```
; definizione della label del Folder Aggiuntivo
LNote
; definizione di un riquadro nel Folder Aggiuntivo
A02025009
; definzione delle didascalie
T0303Data
T0305Importo
T0307Descrizione
; definizione dei campi d'immissione
DT20020071USR0003    XXXXXXXXXXXXXXXXXXX
```

Chiamate a SIGLAPPSaveObject

Se è stata attivata la gestione di campi video aggiuntivi SIGLA/START eseguono una serie di chiamate alla funzione SIGLAPPSaveObject contenuta nella libreria a link dinamico SIGPPDLL.

Le chiamate generate dalla gestione dei campi video aggiuntivi sono contraddistinte dal valore –1000 del parametro theActionID.

Il parametro **theObject** deve essere in tal caso castato sulla classe DPUsrFld. L'header file che definisce la classe DPUsrFld è distribuito assieme al sistema di sviluppo.

La funzione SIGLAPPSaveObject viene chiamata a seguito dei seguenti eventi:

- **durante la fase di cleanup dei campi video della finestra** (ad esempio quando l'utente preme il push button "Nuovo"). In tal caso il campo **actualfld** dell'oggetto DPUsrFld assume il valore **–4**. La chiamata può essere sfruttata per inizializzare il contenuto dei campi aggiuntivi ad un valore di default. I campi automatici vengono sbiancati senza che sia necessario alcun intervento di programmazione.
- **immediatamente dopo la determinazione della chiave univoca per la finestra su cui si sta operando** (ad esempio dopo che l'utente ha individuato il codice univoco). In tal caso il campo **actualfld** dell'oggetto DPUsrFld assume il valore **–1** e il campo **keyfld** contiene il valore della chiave univoca.

- **immediatamente dopo la.memorizzazione del record su cui si sta operando** (ad esempio quando l'utente preme il tasto "Registra"). In tal caso il campo **actualfld** dell'oggetto DPUsrFld assume il valore -**2**
- **immediatamente dopo la.cancellazione del record su cui si sta operando** (ad esempio quando l'utente preme il tasto "Cancella"). In tal caso il campo **actualfld** dell'oggetto DPUsrFld assume il valore -**3**
- **all'uscita di ogni campo aggiuntivo, comunque determinata.** In tal caso il campo **actualfld** dell'oggetto DPUsrFld assume un valore maggiore o uguale a zero che individua il numero progressivo del campo da cui si è usciti. Il primo campo aggiuntivo di input è individuato dal valore 0, il secondo dal valore 1 e così via. Se la funzione torna il valore logico FALSE SIGLA emette un segnale acustico e il controllo di input resta assegnato al campo attuale
- **immediatamente dopo la pressione del tasto "RicercaPlus" quando il controllo di input è assegnato ad un campo aggiuntivo.** In questo caso la variabile booliana **ricercaplus** assume il valore TRUE. E' quindi possibile, per il personalizzatore, sviluppare una adeguata funzione di ricerca.

In ogni caso all'uscita dalla chiamata il pacchetto controlla il valore contenuto nei buffer dei campi video (vedi descrizione della classe DPUsrFld) e riporta sullo schermo quelli variati. Si ha in questo modo la possibilità di operare modifiche sui campi video.

La classe DPUsrFld

La classe DPUsrFld definisce un insieme di dati membro che consentono al personalizzatore di operare sui campi video aggiuntivi. I dati d'interesse sono i seguenti:

- char **videoname[64]** contiene il nome del file di personalizzazione della finestra (senza il postfisso .USR). Il dato consente di determinare quale finestra personalizzata ha generato la chiamata a SIGLAPPSaveObject
- int **actualfld** contiene un intero che determina o il campo video che detiene il controllo di input (valori maggiori o uguali a zero) o il motivo della chiamata alla funzione SIGLAPPSaveObject (valori negativi)
- BOOL **ricercaplus** assume normalmente valore FALSE. E' impostato a TRUE se l'utente ha premuto il pulsante "RicercaPlus". In tal caso la variabile **actualfld** individua il campo che detiene il controllo dell'input.
- char ***usrfld[60]** vettore di 60 componenti ognuna delle quali individua il buffer di un campo video. Il buffer del primo campo video definito è contenuto in usrfld[0], quello del secondo in usrfld[1] e così via. I buffer associati a campi d'immissione di tipo data contengono l'informazione nel formato ANSI (AAAAMMGG), quelli associati a campi numerici contengono il dato sottoforma di stringa ASCII. In tutti i casi la stringa e null terminated.

Finestre modificabili

Finestra	*Nome file .USR*	*Tabelle disponibili*
Agenti	**AGENTI**	**1 - AGENTI**
Aliquote Iva	**TABIVA**	**1 - TABIVA**
Anagrafica Magazzino	**ANAMAG**	**1 - ANAMAGA**
Causali Contabili	**CAUSCO**	**1 - CAUSCONT**
Centri di Costo	**CENCOS**	**1 – CENTRCOS**
Clienti/Fornitori	**CLIFOR**	**1 – CLIFO**
		2 - CFVEN
Famiglie Merceologiche	**FAMMER**	**1 - GRUPMERC**
Gruppi Merceologici	**GRUMER**	**1 - GRUPMERC**
Magazzini	**TABMAG**	**1 - TABMAGAZ**
Marchi	**MARCHI**	**1 - TMARCHI**

Finestra	*Nome file .USR*	*Tabelle disponibili*
Raggruppamento Magazzini	**RAGMAG**	**1 - RAGGRMAG**
Sottoconti	**PIACON**	**1 - PIACON**
Sottofamiglie Merceologiche	**SFAMER**	**1 - GRUPMERC**
Vettori	**VETTOR**	**1 - VETTORI**
Imballi	**IMBAL**	**1 – IMBALLI**
Indirizzi di spedizione	**INDSPED**	**1 – INDSPED**
Anagrafica Cespiti	**ANACES**	**1 – CESPITI**

Esempi

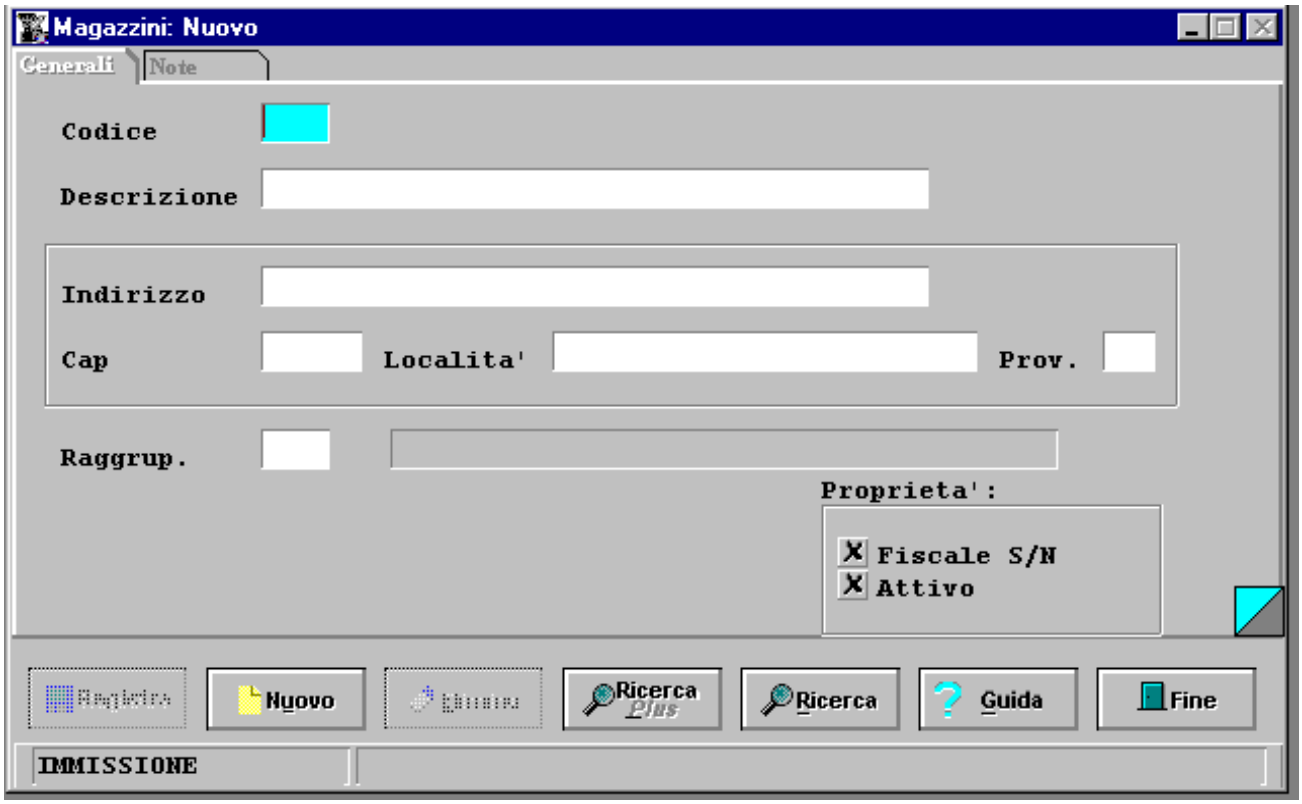
Aggiunta di informazioni sulla finestra tabella magazzini gestite automaticamente

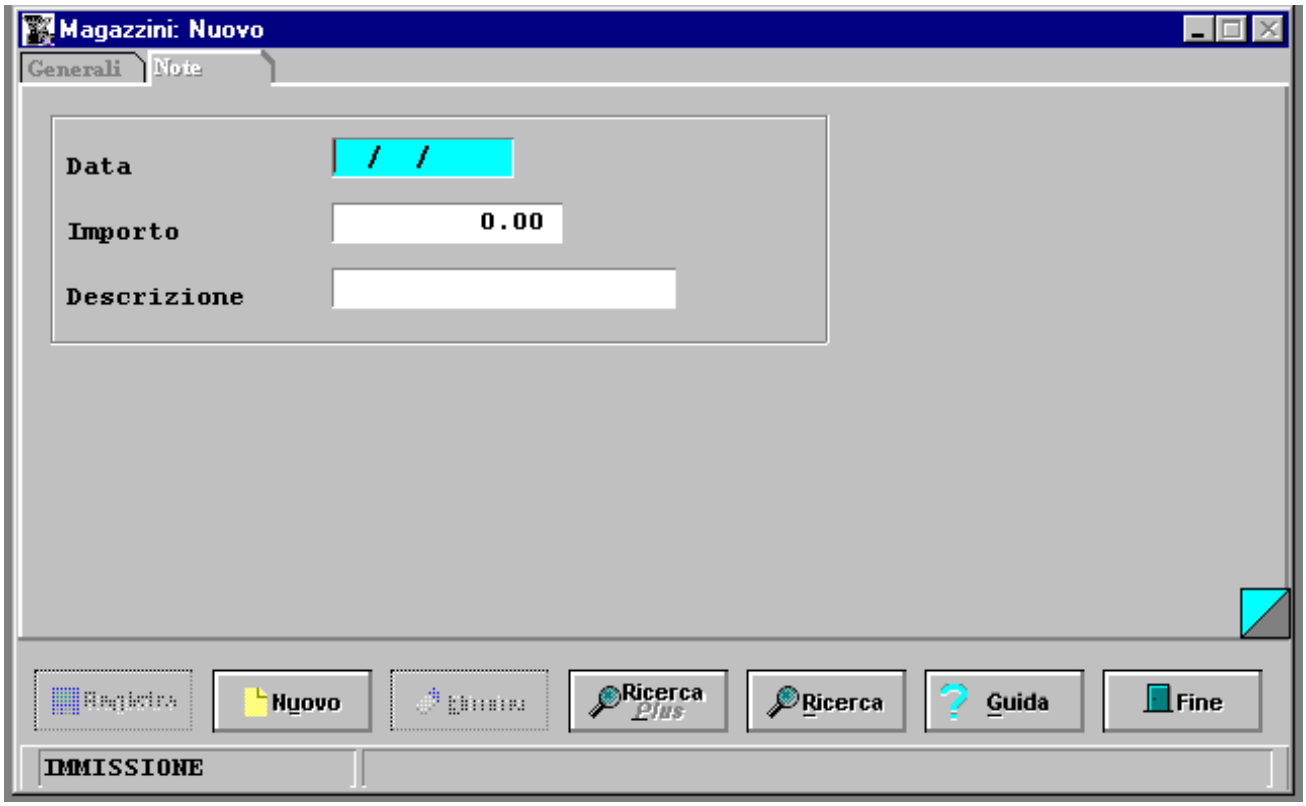
Supponendo di voler aggiungere tre campi (una data, un numero e un testo) alla tabella in questione, è necessario generare con un comune editor il file ASCII **TABMAG.USR** definito come segue (il file deve essere posizionato nell'indirizzario contenente gli applicativi SIGLA):

```
; definizione della label del Folder Aggiuntivo
LNote
; definizione di un riquadro nel Folder Aggiuntivo
A02025009
; definzione delle didascalie
T0303Data
T0305Importo
T0307Descrizione
; definizione dei campi d'immissione
DD08020031USR0001
DN13220051USR0002
DT20020071USR0003    XXXXXXXXXXXXXXXXXXX
```

La struttura della tabella TABMAGAZ nel database dati aziendali deve essere ovviamente modificata con l'aggiunta dei tre campi 'USR0001', 'USR0002' e 'USR0003' (questa operazione può essere effettuata reimplementando la funzione SIGLAPPCreateDitta).

L'aspetto della finestra modificata è illustrato nelle due figure che seguono.


Magazzini: Nuovo
Generali
Note
Codice
Descrizione
Indirizzo
Cap
Localita'
Prov.
Raggrup.
Proprieta':
Fiscale S/N
Attivo
Registra
Nuovo
Elimina
Ricerca Plus
Ricerca
Guida
Fine
IMMISSIONE



Reimplementazione della funzione SIGLAPPSaveObject

L'esempio che segue illustra come ci si debba comportare per reimplementare la funzione SIGLAPPSaveObject allo scopo di acquisire il controllo dei campi aggiuntivi. Facendo riferimento all'esempio precedente, si suppone di voler verificare che la data presentata a video sia essere compresa fra 01/01/98 e 30/06/98 e che, in corrispondenza alla digitazione di una data ammessa, si voglia proporre automaticamente nel campo "Descrizione" la decodifica del mese:

```
BOOL _declspec(dllexport) SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        DPUsrFld *param;
        if(theActionID==-1000) {
                param=(DPUsrFld *) theObject;
                // controlla il nome della finestra
                if(strcmp(param->videoname,"TABMAG")==0) {
                        switch(param->actualfld) {
                                // controlla il campo 0 (data)
                                case 0:
                                        // torna FALSE se la data non è valida
                                        if(strcmp(param->usrfld[0],"19980101")<0 || strcmp(param-
>usrfld[0],"19980630")>0)  return(FALSE);
                                        // controlla il mese (5^ carattere della data)
                                        switch(param->usrfld[0][5]) {
                                                case '1':
                                                  sprintf(param->usrfld[2],"gennaio");
                                                  break;
                                                case '2':
                                                  sprintf(param->usrfld[2],"febbraio");
                                                  break;
                                                case '3':
                                                  sprintf(param->usrfld[2],"marzo");
                                                  break;
                                                case '4':
                                                  sprintf(param->usrfld[2],"aprile");
                                                  break;
                                                case '5':
                                                  sprintf(param->usrfld[2],"maggio");
                                                  break;
                                                case '6':
                                                  sprintf(param->usrfld[2],"giugno");
                                                  break;
                                        }
                        }
                }
        }
        return(TRUE);
}
```

Dalla release 1.02 di e-SIGLA (release 2.14 di SIGLA) le chiamate con actionid -1000 possono essere attivate anche per SPPFrame ed intercettate nella funzione SPPFrameIntSaveObject della SPPFrOld.Per l'attivazione è necessario implementare l'actionid -999 nella funzione SPPFrameIntSaveObject in maniera che il valore di ritorno della funzione sia False(V.documentazione FRAME per maggiori dettagli). Da notare che, nel caso sia stato utilizzato per lo sviluppo il metodo misto FRAME+SIGPPDLL e non si intenda migrare le implementazioni fatte per SIGPPDLL a FRAME, non occorre nessuna variazione, poiché se non si implementa l'actionid -999 per la SPPFrameIntSaveObject o se il ritorno è True le successive uscite -1000 continueranno ad essere indirizzate alla SIGPPDLL.DLL

La libreria a link dinamico SPPSERV.DLL

La DLL SPPSERV.DLL contiene un elenco di funzioni di utilità mirate a consentire ad uno sviluppatore una facile implementazione di azioni di inserimento/cancellazione negli archivi di SIGLA. La libreria **NON** è utilizzata direttamente dall'applicazione ed è stata espressamente concepita per agevolare il lavoro di customizzazione del prodotto.

La DLL viene distribuita per il solo ambiente MS Windows a 32 bit nella versione ODBC e nella versione in grado di operare sugli archivi di SIGLA Start Suite.

Le funzioni fornite sono state implementate **utilizzando direttamente gli oggetti che sono serviti allo sviluppo del pacchetto standard**. Ne consengue che la manutenzione della libreria è direttamente collegata alla manutenzione di SIGLA: tutte le variazioni operative eventualmente apportate al pacchetto saranno immediatamente disponibili in SPPSERV.DLL con lo stesso livello di test dell'applicazione.

N.B.: con la versione 2.05 di SIGLA sono state implementate nuove funzionalità relative alla gestione del magazzino a lotti/ubicazioni. La versione 12 della libreria SPPSERV aggiorna automaticamente la giacenza per ubicazione degli articoli movimentati a magazzino (se il codice ubicazione è correttamente impostato sul movimento di magazzino). La versione 13 della libreria contiene le API necessarie per la gestione dei movimenti per lotto con aggiornamento automatico della giacenza.

ATTENZIONE: a partire dalla release 22 della DLL nelle funzioni che effettuano le connessioni ai dati comuni ed ai dati ditta è stato inserito il controllo di release degli archivi stessi. Pertanto le versioni 22 e successive della DLL potranno funzionare esclusivamente con archivi allineati alla corrispondente versione di SIGLA.

Prerequisiti e avvertenze

SPPSERV.DLL utilizza alcune delle DLL di base distribuite assieme a SIGLA. Tali librerie vengono automaticamente caricate nella cartella di installazione di SIGLA dalla procedura di SETUP. Ne consegue che SPPSERV.DLL deve trovarsi nella stessa cartella d'installazione di SIGLA per funzionare correttamente. Se così non è e si vuole copiare la SPPSERV.DLL in una qualsiasi cartella è obbligatorio copiare in questa cartella tutte le DLL e il file dei messaggi (SPPMSG.ITA) che si trovano nella cartella d'installazione di SIGLA.

La versione ODBC della libreria stabilisce due ulteriori connessioni al database server (8 se il DBMS è SQL Server). Questo fatto deve essere considerato durante la configurazione del DBMS. Se, ad esempio, si prevede di usare SPPSERV.DLL durante la normale attività con SIGLA e si sta usando SQL Server è necessario prevedere almeno 16 connessioni al database per ogni posto di lavoro.

Il file SPPSERV.H

Allo scopo di facilitare il compito ai programmatori C/C++ viene fornito assieme alla DLL l'header file SPPSERV.H contenente la definizione dei prototipi delle funzioni disponibili.

A partire dalla release 4 della DLL le funzioni implementate sono dichiarate come FAR PASCAL nella versione a 16 bit e come _stdcall nella versione a 32 bit.

Elenco funzioni contenute in SPPSERV.DLL per numero rilascio

Release 1

SPPSRVActivateTransaction
SPPSRVAppendMovcoRecord
SPPSRVAppendMovIvaRecord
SPPSRVAppendMovmagRecord
SPPSRVCheckMovcoRecord
SPPSRVCheckMovIvaRecord
SPPSRVComuniCommit
SPPSRVComuniConnect
SPPSRVComuniExecuteSQL
SPPSRVComuniRollBack
SPPSRVDeleteMovcoGroup
SPPSRVDeleteMovcoRecord
SPPSRVDeleteMovIvaGroup
SPPSRVDeleteMovmagRecord
SPPSRVDittaCommit
SPPSRVDittaConnect
SPPSRVDittaExecuteSQL
SPPSRVDittaRollBack
SPPSRVEndMovcoTransaction
SPPSRVEndMovmagTransaction
SPPSRVExit
SPPSRVGetComuniDbmsType
SPPSRVGetDittaDbmsType
SPPSRVGetNum
SPPSRVInit
SPPSRVInitMovcoTransaction
SPPSRVInitMovmagTransaction
SPPSRVSetEsercizio
SPPSRVSetMovcoField
SPPSRVSetMovIvaField
SPPSRVSetMovmagEvaso
SPPSRVSetMovmagField
SPPSRVSetMovmagQnt

SPPSRVSetMovTagEvaso
SPPSRVSetMovTagQnt
SPPSRVSetTodayDate
SPPSRVSetUtente
SPPSRVVersion

Release 3
SPPSRVReadMovmagNumber

Release 4
SPPSRVGetMovcoField
SPPSRVGetMovIvaField
SPPSRVIsMovcoEOF
SPPSRVIsMovIvaEOF
SPPSRVMovcoNext
SPPSRVMovcoSelect
SPPSRVMovIvaNext
SPPSRVMovIvaSelect
SPPSRVSendMail
SPPSRVCalcolaScadenze
SPPSRVReadScadenza

Release 5
SPPSRVDeleteMovmagSet

Release 7
SPPSRVSetTesDocumField
SPPSRVAppendTesDocumRecord

Release 8
SPPSRVInternalInit
SPPSRVInternalExit
SPPSRVInternalComuniConnect
SPPSRVInternalDittaConnect
SPPSRVDoTelephonCall

Release 9
SPPSRVSetEuro

Release 10
SPPSRVGetSerial[18]

Release 11
SPPSRVSetMovmagNumero
SPPSRVSetMovmagRiga

Release 13
SPPSRVInitMovLotTransaction
SPPSRVInitMovLotTransactionPlus
SPPSRVSetMovLotArticolo
SPPSRVSetMovLotTipoEffettivo
SPPSRVSetMovLotField
SPPSRVSetMovLotQnt
SPPSRVSetMovLotQntTag
SPPSRVSetMovLotEvaso
SPPSRVSetMovLotEvasoTag
SPPSRVAppendMovLotRecord
SPPSRVEndMovLotTransaction
SPPSRVDeleteMovLotRecord
SPPSRVDelAllMovLotperMovMag
SPPSRVReadMovLotNumber
SPPSRVSetMovLotNumero

[18] Solo versioni a 32 bit

SPPSRVGetNewCodiceAnLot

Release 26
SPPSRVAppendMovmagRecordEx

Release 32
SPPSRVInitMovMatTransaction
SPPSRVInitMovMatTransactionPlus
SPPSRVSetMovMatField
SPPSRVReadMovMatNumber
SPPSRVSetMovMatNumero
SPPSRVSetMovMatRiga
SPPSRVSetMovMatArticolo
SPPSRVAppendMovMatRecord
SPPSRVEndMovMatTransaction
SPPSRVGetNewCodiceAnMat

Release 34
SPPSRVDittaRollBackEx

Release 41
SPPSRVInitRicMagazzinoTransaction
SPPSRVRicalcolaMagazzino
SPPSRVEndRicMagazzinoTransaction
SPPSRVInitRicContabilitaTransaction
SPPSRVRicalcolaContabilita
SPPSRVEndRicContabilitaTransaction

Release 43
SPPSRVCalcolaPrezzi
SPPSRVCalcolaPrezziEx
SPPSRVReadPrezzo
SPPSRVReadSconto
SPPSRVInitImpMovMagazzinoTransaction
SPPSRVImportazioneMovMagazzino
SPPSRVEndImpMovMagazzinoTransaction
SPPSRVInitImpDocumentiTransaction
SPPSRVImportazioneDocumenti
SPPSRVEndImpDocumentiTransaction
SPPSRVInitImpMovContabiliTransaction
SPPSRVImportazioneMovContabili
SPPSRVEndImpMovContabiliTransaction

Release 56
SPPSRVInitRicLottiTransaction
SPPSRVRicalcolaLotti
SPPSRVEndRicLottiTransaction
SPPSRVInitRicFidoTransaction
SPPSRVRicalcolaFido
SPPSRVEndRicFidoTransaction

Release 65
SPPSRVCalcolaFidoDocumento

Funzioni contenute in SPPSERV.DLL: funzioni generali

int SPPSRVInit(HWND hMainWnd)

Release 1 **SCOPO:** SPPSRVInit() inizializza la DLL e deve essere chiamata **prima** di usare qualunque altra funzione contenuta nella libreria. Il tentativo di usare altre funzioni prima di aver chiamato SPPSRVInit() può comportare la generazione di errori. La funzione deve essere chiamata una sola volta.

PARAMETRI:

* **HWND hMainWnd**: handle della finestra principale dell'applicazione chiamante. Questo valore viene utilizzato durante l'attività di connessione del driver ODBC al database. La sua errata indicazione può generare errori fatali.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **2** DLL già inizializzata

int SPPSRVExit()

Release 1

SCOPO: termina l'attività della DLL ed esegue le azioni di cleanup necessarie. Il tentativo di usare altre funzioni dopo aver chiamato SPPSRVExit() può comportare la generazione di errori.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non predentemente inizializzata

int SPPSRVVersion()

Release 1

SCOPO: la funzione ritorna il numero di versione della DLL come intero. L'implementazione attuale torna un valore costante. A partire dal rilascio 4.12.0/3.29.0 il valore tornato è **71.**

PARAMETRI: nessuno

VALORI DI RITORNO: numero di versione della DLL

int SPPSRVComuniConnect(LPSTR szComuniConnectString)

Release 1

SCOPO: stabilisce la connessione al database dati comuni. La funzione deve essere chiamata **prima** di usare le funzioni operative.

PARAMETRI:

* **LPSTR szComuniConnectString:** per la versione ODBC il parametro è il puntatore al buffer contenente la stringa di connessione ODBC per il database dati comuni; per la versione Start Suite il parametro è il puntatore al buffer contenente la path dell'indirizzario all'interno del quale sono memorizzate le tabelle del database dati comuni.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-3** connessione non stabilita perché la versione degli archivi non corrisponde
* **3** connesione al database già stabilita in precedenza
* **4** errore durante la connessione
* **6** DLL non inizializzata

Esempio:

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()!=1) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
#endif
....
SPPSRVExit();
```

int SPPSRVDittaConnect(LPSTR szDittaConnectString)

Release 1

SCOPO: stabilisce la connessione al database aziendale. La funzione deve essere chiamata **prima** di usare le funzioni operative. SPPSRVDittaConnect può essere chiamata più volte nel corso della stessa applicazione. **La connessione in vigore è quella relativa all'ultima chiamata effettuata.**

La funzione procede alla lettura della tabella di configurazione della ditta (DPCONFIG) e adotta i parametri in essa impostati per l'attività sul database aziendale (lunghezza della chiave di magazzino, lunghezza codice piano dei conti etc.)

La connessione al database dati comuni **deve** essere effettuata **prima** di chiamare questa funzione.

PARAMETRI:

* **LPSTR szDittaConnectString:** per la versione ODBC il parametro è il puntatore al buffer contenente la stringa di connessione ODBC per il database aziendale; per la versione Start Suite il parametro è il puntatore al buffer contenente la path dell'indirizzario all'interno del quale sono memorizzate le tabelle del database aziendale.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-3** connessione non stabilita perché la versione degli archivi non corrisponde
* **4** errore durante la connessione
* **6** DLL non inizializzata
* **50** connessione al database dati comuni non eseguita
* **51** dati necessari all'uso dell'Eurokit non impostati

Esempio:

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()!=1) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
....
SPPSRVExit();
```

int SPPSRVSetEsercizio(LPSTR szEsercizio)

Release 1 **SCOPO:** imposta l'esercizio corrente. La funzione può essere chiamata più volte all'interno della stessa applicazione per cambiare l'esercizio di lavoro. **E' necessario chiamare almeno una volta SPPSRVSetEsercizio prima di procedere ad una azione operativa (registrazione/cancellazione).**

SPPSERV.DLL non effettua alcun tipo di controllo sull'esistenza negli archivi dell'esercizio impostato.

PARAMETRI:

* **szEsercizio:** puntatore al buffer contenente il codice dell'esercizio da impostare (4 caratteri, NULL terminated)

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza parametro non corretta
* **6** DLL non inizializzata

int SPPSRVSetUtente(LPSTR szUtente)

Release 1 **SCOPO:** imposta il codice dell'utente corrente. La funzione può essere chiamata più volte all'interno della stessa applicazione per cambiare il codice utente. **E' necessario chiamare almeno una volta SPPSRVSetUtente prima di procedere ad una azione operativa (registrazione/cancellazione).** Il codice utente viene utilizzato per impostare il campo **UTENTE** dei record trattati. Se la gestione utenti non è stata impostata in SIGLA è comunque conveniente impostare il codice utente al valore "SIGLA " (tre blank finali) per uniformarsi alle caratteristiche di funzionamento del pacchetto base.

SPPSERV.DLL non effettua alcun tipo di controllo sull'esistenza negli archivi del codice utente impostato.

PARAMETRI:

* **szUtente:** puntatore al buffer contenente il codice dell'utente da impostare (8 caratteri, NULL terminated)

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza parametro non corretta
* **6** DLL non inizializzata

int SPPSRVSetTodayDate(LPSTR szTodayDate)

Release 1 **SCOPO:** imposta il valore della data odierna. La funzione può essere chiamata più volte all'interno della stessa applicazione per cambiare la data odierna. **E' necessario chiamare almeno una volta SPPSRVSetTodayDate prima di procedere ad una azione operativa (registrazione/cancellazione).** Il valore impostato viene utilizzato per aggiornare il campo ULT_AGG nei record memorizzati.

SPPSERV.DLL non effettua alcun tipo di controllo di congruenza sul valore impostato.

PARAMETRI:

* **szTodayDate:** puntatore al buffer contenente la data odierna in formato ANSII (AAAMMGG) NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza parametro non corretta
* **6** DLL non inizializzata

int SPPSRVActivateTransaction()

Release 1 **SCOPO:** abilita l'attività transazionale. Dopo la chiamata a SPPSRVActivateTransaction() le funzioni SPPSRVComuniCommit(), SPPSRVDittaCommit(), SPPSRVComuniRollBack(), SPPSRVDittaRollBack() possono essere utilizzate per controllare l'attività transazionale sui database.

Nella versione Start Suite della DLL la funzione non effettua alcuna azione.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata

int SPPSRVComuniExecuteSQL(LPSTR szSQLstring)[19]

Release 1 **SCOPO:** lancia in esecuzione uno statement SQL sul database dati comuni.

PARAMETRI:

* **szSQLstring:** puntatore al buffer contenente la stringa SQL da eseguire (NULL terminated).

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** errore di esecuzione SQL
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

int SPPSRVDittaExecuteSQL(LPSTR szSQLstring)[20]

Release 1 **SCOPO:** lancia in esecuzione uno statement SQL sul database aziendale.

PARAMETRI:

* **szSQLstring:** puntatore al buffer contenente la stringa SQL da eseguire (NULL terminated).

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** errore di esecuzione SQL
* **6** DLL non inizializzata

[19] Disponibile solo per la versione ODBC.

[20] Disponibile solo per la versione ODBC.

* **7** connessione al database non effettuata

Esempio

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()!=1) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
if(SPPSRVDittaExecuteSQL("UPDATE TASALSOT SET DARE=0")!=0) return(sql_error);
SPPSRVDittaCommit();
SPPSRVExit();
```

int SPPSRVGetComuniDbmsType(LPSTR szBuffer,int iBuflen)

Release 1 **SCOPO:** acquisisce il nome del DBMS utilizzato.

PARAMETRI:

* **szBuffer:** puntatore al buffer che dovrà contenere il nome del DBMS.
* **iBuflen:** numero massimo di bytes che la funzione può ritornare in szBuffer.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

N.B.: per la versione Start Suite viene ritornato il valore costante "Start++".

int SPPSRVGetDittaDbmsType(LPSTR szBuffer,int iBuflen)

Release 1 **SCOPO:** acquisisce il nome del DBMS utilizzato.

PARAMETRI:

* **szBuffer:** puntatore al buffer che dovrà contenere il nome del DBMS.
* **iBuflen:** numero massimo di bytes che la funzione può ritornare in szBuffer.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

N.B.: per la versione Start Suite viene ritornato il valore costante "Start++".

Esempio

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()!=1) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
if(SPPSRVDittaGetDbmsType (buffer,sizeof(buffer)!=0) return(dbms_error);
if(strcmp(buffer,"DB2/400 SQL")==0) {
        ...
}
SPPSRVExit();
```

int SPPSRVComuniCommit()

Release 1 **SCOPO:** esegue una Commit sul database dati comuni. La chiamata alla funzione non ha effetto nella versione Start Suite o se l'attività transazionale non è stata inizializzata con la funzione SPPSRVActivateTransaction().

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-1** errore generato dal DBMS
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

int SPPSRVDittaCommit()

Release 1 **SCOPO:** esegue una Commit sul database aziendale. La chiamata alla funzione non ha effetto nella versione Start Suite o se l'attività transazionale non è stata inizializzata con la funzione SPPSRVActivateTransaction().

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-1** errore generato dal DBMS
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

int SPPSRVComuniRollBack()

Release 1 **SCOPO:** esegue una Rollback sul database dati comuni. La chiamata alla funzione non ha effetto nella versione Start Suite o se l'attività transazionale non è stata inizializzata con la funzione SPPSRVActivateTransaction().

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo

* **-1** errore generato dal DBMS
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

int SPPSRVDittaRollBack()

Release 1 **SCOPO:** esegue una Rollback sul database aziendale. La chiamata alla funzione non ha effetto nella versione Start Suite o se l'attività transazionale non è stata inizializzata con la funzione SPPSRVActivateTransaction().

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-1** errore generato dal DBMS
* **6** DLL non inizializzata
* **7** connessione al database non effettuata

int SPPSRVDittaRollBackEx()[21]

Release 34 **SCOPO:** è una versione estesa della precedente funzione, nel senso che oltre ad effettuare esattamente le stesse operazioni della SPPSRVDittaRollBack esegue anche le impostazioni necessarie per consentire di eseguire nuovamente l'inizializzazione della transazione per la registrazione dei movimenti. Questa funzione può essere eseguita per annullare le modifiche apportare dalle chiamate delle funzioni SPPSRVAppend… eseguite dopo la corrispondente SPPSRVInit…. Si precisa che viene eseguito il rollback di tutti gli statement SQL eseguiti dopo l'ultima commit emessa implicitamente attraverso le SPPSRVEnd…Transaction o esplicitamente attraverso al SPPSRVDittaCommit.

Valgono le stesse considerazioni illustrate per la precedente funzione SPPSRVDittaRollBack, sia per quanto riguarda le note di funzionamento che i parametri e i valori di ritorno.

long int SPPSRVGetNum(LPSTR szNumCode)

Release 1 **SCOPO:** acquisisce un protocollo univoco datta tabella numeratori di SIGLA.

PARAMETRI:

* **szNumCode:** puntatore al buffer che contiene il codice del numeratore da usare (due caratteri NULL terminated)

VALORI DI RITORNO: la funzione torna valori negativi in caso d'errore. Se la funzione va a buon fine il valore di ritorno è il protocollo cercato.

* **-1** numeratore non trovato (codice non esistente)
* **-2** esercizio non impostato
* **-3** connessione al database aziendale non effettutata
* **-4** lunghezza codice numeratore errata

Poiche' SIGLA gestisce i numeratori in funzione dell'esercizio prescelto è fondamentale aver impostato con la funzione SPPSRVSetEsercizio il codice dell'esercizio attuale.

Il numero di protocollo recuperato è il primo disponibile. Il valore del contatore usato da SIGLA per la gestione del numeratore viene automaticamente incrementato di uno. **Per garantire l'univocita' del dato la funzione esegue automaticamente una Commit sul database aziendale.**

Esempio

[21] Disponibile solo per la versione ODBC

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()!=1) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
SPPSRVSetEsercizio("1997");
// acquisice un nuovo protocollo dal numeratore FV
if((protoc=SPPSRVGetNum ("FV"))<0L) return(num_error);
...
SPPSRVExit();
```

int SPPSRVSendMail(LPSTR szSoggetto,LPSTR szAddress,LPSTR szFilepath,LPSTR szFiledescr,LPSTR szNota,BOOL bDodialog)[22]

Release 4 **SCOPO:** invia una E mail.

PARAMETRI:

* **szSoggetto:** puntatore al buffer che contiene il soggetto della mail (NULL terminated)
* **szAddress:** puntatore al buffer che contiene l'indirizzo di posta a cui inviare la mail (NULL terminated)
* **szFilepath:** puntatore al buffer contenente la path name del file da allegare alla mail come attachment. Se non si desidera allegare alcun file il parametro deve puntare ad un valore "\0".
* **szFiledescr:** puntatore al buffer contenente il nome del file allegato così come deve essere visto da chi riceve la mail. Se non si desidera allegare alcun file il parametro deve puntare ad un valore "\0".
* **szNota:** puntatore al buffer contenente il corpo del messaggio da inviare. La dimensione massima del buffer non deve oltrepassare i 4000 bytes. I ritorni a capo possono essere gestiti inserendo opportunamente nel corpo del testo dei caratteri "salto linea" (0A esadecimale)
* **bDodialog:** deve valere TRUE se la mail deve essere proposta all'utente che potra' decidere eventuali modifiche ed eventualmente abortire l'invio, FALSE se la mail deve essere inviata senza alcun prompt per l'utente.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** errore durante l'invio
* **6** DLL non inizializzata

Esempio

[22] Disponibile solo per le versioni a 32 bit

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()<4) {
        return(invalid_version)
}
// il seguente statement invia una mail contenente un attachment. L'utente assume il controllo dopo la preparazione della mail
SPPSRVSenMail("Lettera di prova","s++supp.tecn@leonet.it","c:\\prova\\file1.txt","file1.txt","\nQuesto è il corpo del messaggio di prova.\nIl file contiene un attachment.\n",TRUE);
// il seguente statement invia una mail senza attachment. La mail viene inviata automaticamente senza alcun controllo da parte
// dell'utente
SPPSRVSenMail("Lettera di prova","s++supp.tecn@leonet.it","","","\nQuesto è il corpo del messaggio di prova.\nIl file non contiene attachment.\n",FALSE);
SPPSRVExit();
```

int SPPSRVDoTelephonCall(LPSTR szNumero,LPSTR szChiamato,LPSTR szCommento)[23]

Release 8 **SCOPO:** esegue una chiamata telefonica

PARAMETRI:

* **szNumero:** puntatore al buffer che contiene il numero da comporre (NULL terminated)
* **szChiamato:** puntatore al buffer che contiene il nominativo dell'utente chiamato (NULL terminated)
* **szCommento:** puntatore al buffer contenente un eventuale commento

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **valore diverso da zero:** condizione d'errore

Esempio

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()<4) {
        return(invalid_version)
}
// il seguente statement esegue una chiamata telefonica.
SPPSRVDoTelephonCall("05719988","DeltaPhi","");
SPPSRVExit();
```

int SPPSRVSetEuro(BOOL IsEuro)

Release 9 **SCOPO:** imposta la valuta di riferimento (valuta di conto) per l'immissione

PARAMETRI:

* **IsEuro:** TRUE se la valuta di riferimento per l'immissione è l'Euro, FALSE se la valuta di riferimento è la Lira italiana (default)

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo

[23] Disponibile solo per le versioni a 32 bit

* **valore diverso da zero:** condizione d'errore

Le funzioni per la gestione dei movimenti contabili e di magazzino sono in grado, a partire dalla versione 9 della libreria (compatibile con l'Eurokit) di gestire **automaticamente** la conversione negli appositi campi degli importi dalla valuta di conto alla valuta secondaria. Ad esempio se la valuta di conto è la Lira italiana e si sta aggiungendo un movimento contabile, impostando un valore nel campo IMPORTO l'equivalente in Euro di tale dato sara' automaticamente inserito nel campo EIMPORTO. Viceversa, usando l'Euro come valuta di conto, dopo aver impostato il dato voluto nel campo EIMPORTO si otterra' la sua conversione in Lire nel campo IMPORTO.

La specifica della valuta di conto e i meccanismi di conversione automatica diventano operativi solo se l'eurokit è stato attivato.

int SPPSRVGetSerial(LPSTR szSerial)[24]

Release10 **SCOPO:** legge il numero di serie della chiave di attivazione[25]

PARAMETRI:

* **szSerial:** puntatore al buffer dove sara' copiato il numero di serie

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **valore diverso da zero:** condizione d'errore (chiave non correttamente funzionante o non installata)

In caso di chiamata con successo i primi 4 bytes contengono il serial number della chiave hardware di attivazione.

Funzioni contenute in SPPSERV.DLL: operazioni contabili

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di registrazione/cancellazione di movimenti contabili e iva.

int SPPSRVInitMovcoTransaction()

Release 1 **SCOPO:** inizia una nuova transazione di registrazione di movimenti contabili e iva.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **8** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

1. viene preparato **in memoria** un nuovo record per la tabella MOVCO. I campi numerici sono impostati a 0, i campi carattere sono impostati al valore NULL
2. viene preparato **in memoria** un nuovo record per la tabella MOVIVA. I campi numerici sono impostati a 0, i campi carattere sono impostati al valore NULL
3. viene acquisito un numero univoco per il raggruppamento di registrazioni

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndMovcoTransaction() non appena possibile.**

[24] Disponibile solo per le versioni a 32 bit

[25] A partire dalla versione 42 legge il numero seriale della chiave di protezione anche in ambiente Windows Terminal Services.

int SPPSRVEndMovcoTransaction()

Release 1 **SCOPO:** termina una transazione di registrazione di movimenti contabili e iva.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata
* **35** saldo dare/avere della transazione non bilanciato

La funzione esegue una Commit sul database aziendale. Per iniziare una nuova registrazione contabile è necessario chiamare nuovamente SPPSRVInitMovcoTransaction().

A partire dalla versione 9 della libreria, se l'Eurokit è stato attivato e se è stata configurata la generazione automatica dei movimenti a compensazione sulla valuta secondaria, la chiamata alla funzione SPPSRVEndMovcoTransaction() genera se necessario il movimento contabile a compensazione della quadratura dare/avere della registrazione.

int SPPSRVSetMovcoField(LPSTR szFldName, LPSTR szValue)

Release 1 **SCOPO:** imposta il contenuto del campo specificato nel record della tabella MOVCO prima di procedere all'aggiunta del record. La funzione deve essre chiamata per impostare tutti i valori necessari.

PARAMETRI:

* **szFldName:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValue:** puntatore al buffer contenente il valore da assegnare al campo NULL terminated.**N.B.: per i campi numerici il valore deve essere impostato come stringa di caratteri in formato ASCII..**

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza dato non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata (warning)
* **10** campo non trovato

int SPPSRVCheckMovcoRecord()

Release 1 **SCOPO:** esegue un controllo di congruenza sui dati impostati nel movimento contabile. La funzione **può** essere chiamata **prima** di procedere all'inserimento del record nella tabella MOVCO tramite la funzione SPPSRVAppendMovcoRecord().

La funzione imposta automaticamente, prelevandoli dalla causale contabile, i seguenti flag sul movimento contabile:

* SOLOANA_SN
* SIMULAZ_SN
* SOSPESO_SN

Vengono inoltre impostati automaticamente i seguenti campi:

* TIPOCAUSAL (dalla causale contabile)
* C_F (dal sottoconto movimentato)

A partire dalla release 71, se la causale contabile prevede anche la registrazione della parte IVA (imponibili e imposte) viene automaticamente impostato il seguente campo:

* TIPONUMERA nella tabella MOVIVA(dalla causale contabile)

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata
* **36** sottoconto non trovato
* **37** causale contabile non trovata

int SPPSRVAppendMovcoRecord()

Release 1 **SCOPO:** appende il record corrente alla tabella MOVCO. Prima di procedere all'append fisico del record vengono eseguiti una serie di controlli di congruenza coomplementari a quelli eseguiti dalla funzione SPPSRVCheckMovcoRecord().

La funzione aggiorna automaticamente la tabella TASALSOT con i saldi dare/avere del sottoconto movimentato. Se il record necessario non esiste viene creato.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata
* **11** segno dare/avere non valido (il campo SEGNO del record non contiene il valore "D" o "A")
* **12** sottoconto non impostato (il campo SOTTOCONTO è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sull'esistenza del sottoconto impostato. Il controllo in questione viene eseguito da SPPSRVCheckMovcoRecord()**
* **13** flag SIMULAZ_SN non impostato in modo corretto (il campo deve contenere il valore "S" o "N")
* **14** flas SOSPESO_SN non impostato in modo corretto (il campo deve contenere il valore "S" o "N")
* **15** flag TEMPORA_SN non impostato in modo corretto (il campo deve contenere il valore "S" o "N")
* **16** flag SOLOANA_SN non impostato in modo corretto (il campo deve contenere il valore "S", "N" o "C")
* **17** causale contabile non impostata (il campo CAUSALE è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sull'esistenza della causale impostata. Il controllo in questione viene eseguito da SPPSRVCheckMovcoRecord()**
* **18** data di registrazione non impostata (il campo DATAREG è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sulla validita' della data impostata.**
* **19** esercizio di competenza non impostato (il campo ESECOMPET è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sulla validita' del codice esercizio di competenza impostato.**
* **20** esercizio di registrazione non impostato (il campo ESEREGISTR è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sulla validita' del codice esercizio di registrazione impostato.**

* **21** data di competenza non impostata (il campo DATACOMPET è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sulla validita' della data impostata.**
* **34** caso non correttamente impostato: il campo CASO non è stato impostato ad un valore adeguato. I possibili valori ammessi sono:
    1. "0" per le registrazioni relative a fatture di acquisto/vendita, note di credito e note di debito
    2. "1" per le registrazioni relative a fatture di acquisto intracomunitarie
    3. "2" per la registrazione di operazioni di pagamento di fatture in sospensione d'imposta
    4. "4" per registrazioni di prima nota senza movimentazione iva
    5. "5" per registrazione di fatture extracee

La funzione non effettua alcun controllo sul riempimento del campo CCOSTO che deve eventualemnte essere riempito e controllato a cura del programmatore se il movimento inserito è relativo alla contabilita' analitica.

Int SPPSRVDeleteMovcoRecord(LPSTR szGroup,LPSTR szRow)

Release 1 **SCOPO:** la funzione consente la cancellazione di una singola scrittura contabile. Se la registrazione è associata ad una movimentazione iva anche la parte presente nella tabella MOVIVA viene cancellata. **Non è possibile usare questa funzione per cancellare movimenti inseriti con le funzioni precedenti prima che sia chiamata la funzione SPPSRVEndMovcoTransaction().**

PARAMETRI:

* **szGroup:** puntatore al buffer contenente il numero del raggruppamento di registrazioni che contiene il record da cancellare NULL terminated.
* **szRow:** puntatore al buffer contenente il numero di riga da cancellare all'interno del raggruppamento di registrazioni NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-2** record non trovato: la coppia szGroup,szRow non individua un record presente nella tabella MOVCO
* **5** lunghezza dato non valida: il numero di gruppo deve essere una stringa di 7 bytes zerofilled, il numero di riga deve essere una stringa di 5 bytes zerofilled
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

N.B.: la funzione non esegue una Commit. L'utilizzatore è responsabile della chiusura della transazione che deve essere effettuata chiamando SPPSRVDittaCommit().

int SPPSRVDeleteMovcoGroup(LPSTR szGroup)

Release 1 **SCOPO:** cancella un intero raggruppamento di registrazioni. Se il raggruppamento contiene una registrazione associata ad una movimentazione iva (quindi con con MOV_IVA_SN='S') anche la parte presente nella tabella MOVIVA viene cancellata. **Non è possibile usare questa funzione per cancellare movimenti inseriti con le funzioni precedenti prima che sia chiamata la funzione SPPSRVEndMovcoTransaction().**

PARAMETRI:

* **szGroup:** puntatore al buffer contenente il numero del raggruppamento di registrazioni che contiene il record da cancellare NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo (il valore viene tornato anche se il raggruppamento da cancellare non esiste)
* **7** non connesso al database aziendale o al database dati comuni

N.B.: la funzione non esegue una Commit. L'utilizzatore è responsabile della chiusura della transazione che deve essere effettuata chiamando SPPSRVDittaCommit().

int SPPSRVSetMovIvaField(LPSTR szFldName, LPSTR szValue)

Release 1 **SCOPO:** imposta il contenuto del campo specificato nel record della tabella MOVIVA prima di procedere all'aggiunta del record. La funzione deve essre chiamata per impostare tutti i valori necessari.

PARAMETRI:

* **szFldName:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValue:** puntatore al buffer contenente il valore da assegnare al campo NULL terminated.**N.B.: per i campi numerici il valore deve essere impostato come stringa di caratteri in formato ASCII.**

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza dato non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **10** campo non trovato

int SPPSRVCheckMovIvaRecord()

Release 1 **SCOPO:** esegue un controllo di congruenza sui dati impostati nel movimento iva. La funzione **può** essere chiamata **prima** di procedere all'inserimento del record nella tabella MOVIVA tramite la funzione SPPSRVAppendMovIvaRecord().

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata
* **38** sottoconto non trovato
* **39** tipo documento iva non trovato
* **40** registro iva non trovato
* **41** aliquota iva non trovata
* **104** tipo numeratore iva impostato non correttamente: il campo TIPONUMERA deve contenere i valori "F", "C" o "D"

La funzione imposta automaticamente, prelevandoli dal sottoconto, i seguenti dati sul movimento iva:

* C_F (dal sottoconto movimentato)

int SPPSRVAppendMovIvaRecord()

Release 1 **SCOPO:** appende il record corrente alla tabella MOVIVA. Prima di procedere all'append fisico del record vengono eseguiti una serie di controlli di congruenza coomplementari a quelli eseguiti dalla funxione SPPSRVCheckMovIvaRecord(). La data di registrazione del movimento iva viene impostata automaticamente uguale a quella del movimento contabile impostato in memoria al momento dell'esecuzione. La stessa data determina l'anno solare iva di competenza della registrazione ai fini della gestione della liquidazione.

Il movimento contabile associato deve avere i seguenti campi impostati in modo congruente alla registrazione sulla tabella MOVIVA:

1. REGIVA codice del registro iva
2. TIPODOCIVA codice del tipo dodumento iva

3. MOV_IVA_SN deve essere impostato a “S”

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata
* **12** sottoconto non impostato (il campo SOTTOCONTO è stato lasciato in bianco). **N.B.: la funzione non effettua alcun controllo sull’esistenza del sottoconto impostato. Il controllo in questione viene eseguito da SPPSRVCheckMovIvaRecord()**
* **18** data di registrazione non impostata. Poiche’ la data di registrazione viene riportata automaticamente dal movimento contabile associato è necessario verificare di aver impostato tale dato. **N.B.: la funzione non effettua alcun controllo sulla validita’ della data impostata.**
* **22** anno di competenza iva non valido. Poiche’ tale dato viene desunto dalla data di registrazione il problema va risolto a livello di impostazione della data di registrazione sul movimento contabile corrispondente
* **23** numero protocollo non impostato: il campo NUMPROTOC non è stato impostato.
* **24** data documento non impostata: il campo DATADOC non è stato impostato
* **25** numero documento non impostato: il campo NUMERODOC non è stato impostato
* **26** tipo cliente/fornitore impostato non correttamente: il campo C_F deve contenere i valori “C”, “F” o “A”
* **27** tipo documento iva non impostato: il campo TIPODOCIVA non è stato impostato. **N.B.: la funzione non effettua alcun controllo sull’esistenza del dato impostato. Il controllo in questione viene eseguito da SPPSRVCheckMovIvaRecord()**
* **28** codice registro iva non impostato: il campo REGISTRO non è stato impostato. **N.B.: la funzione non effettua alcun controllo sull’esistenza del dato impostato. Il controllo in questione viene eseguito da SPPSRVCheckMovIvaRecord()**
* **29** aliquota iva non impostata: il campo CODIVA non è stato impostato. **N.B.: la funzione non effettua alcun controllo sull’esistenza del dato impostato. Il controllo in questione viene eseguito da SPPSRVCheckMovIvaRecord()**
* **30** il flag di registrazione in ritardo non è stato correttamente impostato. Il campo RITARDO_SN deve contenere i valori “S” o “N”
* **31** il flag “movimento liquidato” non è stato correttamente impostato. Il campo STLIQ_SN deve contenere i valori “S” o “N”
* **32** il flag “movimento stampato su bollato” non è stato impostato correttamente. Il campo STAMPAT_SN deve contenere i valori “S” o “N”
* **33** il flag “movimento temporaneo” non è stato correttamente impostato. Il campo TEMPORA_SN deve contenere i valori “S” o “N”
* **104** tipo numeratore iva impostato non correttamente: il campo TIPONUMERA deve contenere i valori “F”, “C” o “D”

int SPPSRVDeleteMovIvaGroup(LPSTR szGroup)

Release 1 **SCOPO:** cancella un intero raggruppamento di registrazioni iva. La parte contabile della registrazione non viene cancellata; per ottenere questo risultato è opportuno utilizzare le funzioni SPPSRVDeleteMovcoGroup() o SPPSRVDeleteMovcoRecord(). **Non è possibile usare questa funzione per cancellare movimenti inseriti con le funzioni precedenti prima che sia chiamata la funzione SPPSRVEndMovcoTransaction().**

PARAMETRI:

* **szGroup:** puntatore al buffer contenente il numero del raggruppamento di registrazioni che contiene il record da cancellare NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-1** errore generico durante la cancellazione.
* **5** lunghezza dato non valida: il numero di raggruppamennto deve essere una stringa zeofilled di 7 bytes
* **7** non connesso al database aziendale o al database dati comuni

N.B.: la funzione non esegue una Commit. L'utilizzatore è responsabile della chiusura della transazione che deve essere effettuata chiamando SPPSRVDittaCommit().

int SPPSRVMovcoSelect(LPSTR szInitialvalue,LPSTR szFinalvalue,int iCriteriaNo)

Release 4 **SCOPO:** estrae dalla tabella dei movimenti contabili un cursore di dati. Le informazioni recuperate possono essere gestite tramite le funzioni **SPPSRVGetMovcoField(),SPPSRVIsMovcoEOF() e SPPSRVMovcoNext().**

PARAMETRI:

* **szInitialvalue:** puntatore al buffer contenente il valore iniziale per la selezione dei dati NULL terminated. La formattazione del valore contenuto del buffer varia in funzione del valore di **iCriteriaNo**
* **szFinalvalue:** puntatore al buffer contenente il valore finale per la selezione dei dati NULL terminated. La formattazione del valore contenuto del buffer varia in funzione del valore di **iCriteriaNo**
* **iCriteriaNo:** intero che individua il tipo di estrazione richiesta. I valori possibili sono elencati di seguito

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** il cursore selezionato non contiene dati (nessun record soddisfa le condizioni richieste)
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

SELEZIONI POSSIBILI: di seguito sono specificati i valori possibili per il parametro **iCriteriaNo.**

iCriteriaNo	Selezione	szInitialvalue	szFinalvalue
1	**estrae le righe per la stampa del giornale contabile per l'esercizio impostato tramite SPPSRVSetEsercizio(). Vengono estratte le sole righe ancora non stampate in forma definitiva. I dati estratti sono ordinati per data di registrazione. Prima di chiamare la funzione deve essere impostato tramite una chiamata a SPPSRVSetMovcoField() un valore per il campo "UTENTE". Se il campo utente contiene una stringa vuota vengono estratti i movimenti inseriti da tutti gli utenti. Se il campo utente è impostato sul codice di un particolare utente vengono estratti solo i movimenti registrati dall'utente indicato.**	**data inizio estrazione in formato ANSI (AAAAMMGG)**	**data fine estrazione in formato ANSI (AAAAMMGG)**
2	**estrae le righe per le schede contabili. Vengono selezionate le righe relative all'esercizio impostato tramite SPPSRVSetEsercizio() e comprese fra un range di sottoconti e un range di date di registrazione. I dati estratti sono ordinati per codice sottoconto e data registrazione.**	**Codice sottoconto iniziale (primi 10 caratteri) più data iniziale in formato ANSI (il dato è lungo in totale 18 caratteri).**	**Codice sottoconto finale (primi 10 caratteri) più data finale in formato ANSI (il dato è lungo in totale 18 caratteri).**

3	**estrae le righe per la stampa del giornale contabile per l'esercizio impostato tramite SPPSRVSetEsercizio(). Vengono estratte le sole righe ancora non stampate in forma definitiva. I dati estratti sono ordinati per data di ultimo aggiornamento. Prima di chiamare la funzione deve essere impostato tramite una chiamata a SPPSRVSetMovcoField() un valore per il campo "UTENTE". Se il campo utente contiene una stringa vuota vengono estratti i movimenti inseriti da tutti gli utenti. Se il campo utente è impostato sul codice di un particolare utente vengono estratti solo i movimenti registrati dall'utente indicato.**	**data inizio estrazione in formato ANSI (AAAAMMGG)**	**data fine estrazione in formato ANSI (AAAAMMGG)**
5	**estrae le righe di scadenzario in ordine di data scadenza. Prima di chiamare la funzione devono essere impostati tramite chiamate a SPPSRVSetMovcoField() adeguati valori per i campi:** **"UTENTE". Se il campo contiene una stringa vuota vengono estratti i movimenti inseriti da tutti gli utenti. Se il campo utente è impostato sul codice di un particolare utente vengono estratti solo i movimenti registrati dall'utente indicato.** **"ATT_PASSIV". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se contiene una "P" vengono estratti i movimenti dello scadenzario passivo, se contiene una "A" vengono estratti i movimenti dello scadenzario attivo.** **"TIPOEFFETT". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se è impostato su un valore ammissibile per il tipo effettivo di pagamento (vedi documentazione del database note_tec.hlp) vengono estratti i soli records relativi al tipo effettivo di pagamento individuato.** **"SOSPESO_SN". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se contiene una "S" vengono estratti i soli movimenti sospesi, se contiene una "N" vengono estratti i soli movimenti non sospesi.** **"SOTTOCONTO". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se viene impostato il codice di un sottoconto vengono estratti i soli records relativi al sottoconto specificato.** **"CODVALUTA". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se viene impostato il codice di una valuta estera vengono estratti i soli**	**data scadenza per inizio estrazione in formato ANSI (AAAAMMGG)**	**data scadenza per fine estrazione in formato ANSI (AAAAMMGG)**

	records relativi alla valuta specificata.		
6	estrae le righe di scadenzario in ordine di codice sottoconto e data scadenza. Prima di chiamare la funzione devono essere impostati tramite chiamate a SPPSRVSetMovcoField() adeguati valori per i campi: "UTENTE". Se il campo contiene una stringa vuota vengono estratti i movimenti inseriti da tutti gli utenti. Se il campo utente è impostato sul codice di un particolare utente vengono estratti solo i movimenti registrati dall'utente indicato. "ATT_PASSIV". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se contiene una "P" vengono estratti i movimenti dello scadenzario passivo, se contiene una "A" vengono estratti i movimenti dello scadenzario attivo. "TIPOEFFETT". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se è impostato su un valore ammissibile per il tipo effettivo di pagamento (vedi documentazione del database note_tec.hlp) vengono estratti i soli records relativi al tipo effettivo di pagamento individuato. "SOSPESO_SN". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se contiene una "S" vengono estratti i soli movimenti sospesi, se contiene una "N" vengono estratti i soli movimenti non sospesi. "SOTTOCONTO". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se viene impostato il codice di un sottoconto vengono estratti i soli records relativi al sottoconto specificato. "CODVALUTA". Se il campo contiene una stringa vuota vengono estratti tutti i movimenti, se viene impostato il codice di una valuta estera vengono estratti i soli records relativi alla valuta specificata.	data scadenza per inizio estrazione in formato ANSI (AAAAMMGG)	data scadenza per fine estrazione in formato ANSI (AAAAMMGG)
102	estrae le righe per le schede contabili. Vengono selezionate le righe relative all'esercizio impostato tramite SPPSRVSetEsercizio() e comprese fra un range di sottoconti e un range di date di registrazione. I dati estratti sono ordinati per codice sottoconto e numero documento.	Codice sottoconto iniziale (primi 10 caratteri) più data iniziale in formato ANSI (il dato è lungo in totale 18 caratteri).	Codice sottoconto finale (primi 10 caratteri) più data finale in formato ANSI (il dato è lungo in totale 18 caratteri).
202	estrae le righe per le schede contabili. Vengono selezionate le righe relative all'esercizio impostato tramite SPPSRVSetEsercizio() e comprese fra un range di sottoconti e un range di date di	Codice sottoconto iniziale (primi 10 caratteri) più data iniziale in formato ANSI (il dato è	Codice sottoconto finale (primi 10 caratteri) più data finale in formato ANSI (il dato è

	registrazione. I dati estratti sono ordinati per codice sottoconto e numero protocollo.	lungo in totale 18 caratteri).	lungo in totale 18 caratteri).
302	estrae le righe per le schede contabili. Vengono selezionate le righe relative all'esercizio impostato tramite SPPSRVSetEsercizio() e comprese fra un range di sottoconti e un range di date di registrazione. I dati estratti sono ordinati per codice sottoconto, codice contropartita e data registrazione.	Codice sottoconto iniziale (primi 10 caratteri) più data iniziale in formato ANSI (il dato è lungo in totale 18 caratteri).	Codice sottoconto finale (primi 10 caratteri) più data finale in formato ANSI (il dato è lungo in totale 18 caratteri).
402	estrae le righe per le schede contabili. Vengono selezionate le righe relative all'esercizio impostato tramite SPPSRVSetEsercizio() e comprese fra un range di sottoconti e un range di date di registrazione. I dati estratti sono ordinati per codice sottoconto, importo e data registrazione.	Codice sottoconto iniziale (primi 10 caratteri) più data iniziale in formato ANSI (il dato è lungo in totale 18 caratteri).	Codice sottoconto finale (primi 10 caratteri) più data finale in formato ANSI (il dato è lungo in totale 18 caratteri).

Esempio

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()<4) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
SPPSRVSetEsercizio("1997");
// estrazione movimenti sospesi dallo scadenzario attivo in ordine di data scadenza
SPPSRVSetMovcoField("UTENTE","");
SPPSRVSetMovcoField("ATT_PASSIV","A");
SPPSRVSetMovcoField("TIPOEFFETT","");
SPPSRVSetMovcoField("SOSPESO_SN","S");
SPPSRVSetMovcoField("SOTTOCONTO","");
SPPSRVSetMovcoField("CODVALUTA","");
if(SPPSRVMovcoSelect("19970101","19971231",5)==0) {
        while(SPPSRVIsMovcoEOF()==0) {
                SPPSRVGetMovcoField("IMPORTO",buffer,sizeof(buffer));
                SPPSRVMovcoNext();
        }
}
SPPSRVExit();
```

int SPPSRVMovcoNext()

Release 4 **SCOPO:** si posiziona sul record successivo del cursore dati estratto con una chiamata a **SPPSRVMovcoSelect()**.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** il cursore selezionato non contiene ulteriori dati (si è raggiunta la fine del cursore)
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

int SPPSRVIsMovcoEOF()

Release 4 **SCOPO:** testa se il cursore dati estratto con una chiamata a **SPPSRVMovcoSelect()** contiene records.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** il cursore contiene ulteriori dati
* **1** il cursore selezionato non contiene ulteriori dati (si è raggiunta la fine del cursore)
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

int SPPSRVGetMovcoField(LPSTR szFiledname, LPSTR szValuebuffer, int iBuflen)

Release 4 **SCOPO:** recupera il valore contenuto in un campo del record attuale della tabella movimenti contabili.

PARAMETRI:

* **szFieldname:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValuebuffer:** puntatore al buffer dove il valore recuperato dovrà essere copiato
* **iBuflen:** ampiezza nmassima in caratteri del buffer dove il valore recuperato dovrà essere copiato

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza buffer non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **10** nome campo non valido

N.B.: i dati numerici vengono tornati come stringe ASCII formattate su 15 caratteri.

int SPPSRVMovIvaSelect(LPSTR szInitialvalue,LPSTR szFinalvalue,int iCriteriaNo)

Release 4 **SCOPO:** estrae dalla tabella dei movimenti iva un cursore di dati. Le informazioni recuperate possono essere gestite tramite le funzioni **SPPSRVGetMovIvaField(),SPPSRVIsMovIvaEOF() e SPPSRVMovIvaNext().**

PARAMETRI:

* **szInitialvalue:** puntatore al buffer contenente il valore iniziale per la selezione dei dati NULL terminated. La formattazione del valore contenuto del buffer varia in funzione del valore di **iCriteriaNo**

* **szFinalvalue:** puntatore al buffer contenente il valore finale per la selezione dei dati NULL terminated. La formattazione del valore contenuto del buffer varia in funzione del valore di **iCriteriaNo**
* **iCriteriaNo:** intero che individua il tipo di estrazione richiesta. I valori possibili sono elencati di seguito

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** il cursore selezionato non contiene dati (nessun record soddisfa le condizioni richieste)
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

SELEZIONI POSSIBILI: di seguito sono specificati i valori possibili per il parametro **iCriteriaNo.**

iCriteriaNo	Selezione	szInitialvalue	szFinalvalue
0	**estrae le righe per la stampa di un registro iva in ordine di data registrazione e numero protocollo.** **Vengono estratte le sole righe ancora non stampate in forma definitiva. Prima di chiamare la funzione deve essere impostato tramite una chiamata a SPPSRVSetMovIvaField() un valore per il campo "REGISTRO", corrispondente al codice del registro iva di cui interessano i dati**	**data inizio estrazione in formato ANSI (AAAAMMGG)**	**data fine estrazione in formato ANSI (AAAAMMGG)**

int SPPSRVMovIvaNext()

Release 4 **SCOPO:** si posiziona sul record successivo del cursore dati estratto con una chiamata a **SPPSRVMovIvaSelect()**.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **1** il cursore selezionato non contiene ulteriori dati (si è raggiunta la fine del cursore)
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

int SPPSRVIsMovIvaEOF()

Release 4 **SCOPO:** testa se il cursore dati estratto con una chiamata a **SPPSRVMovIvaSelect()** contiene records.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** il cursore contiene ulteriori dati
* **1** il cursore selezionato non contiene ulteriori dati (si è raggiunta la fine del cursore)
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

int SPPSRVGetMovIvaField(LPSTR szFiledname, LPSTR szValuebuffer, int iBuflen)

Release 4 **SCOPO:** recupera il valore contenuto in un campo del record attuale della tabella movimenti contabili.

PARAMETRI:

* **szFieldname:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValuebuffer:** puntatore al buffer dove il valore recuperato dovrà essere copiato
* **iBuflen:** ampiezza nmassima in caratteri del buffer dove il valore recuperato dovrà essere copiato

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza buffer non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **10** nome campo non valido

N.B.: i dati numerici vengono tornati come stringe ASCII formattate su 15 caratteri.

int SPPSRVCalcolaScadenze(LPSTR szTipoPagamento,BOOL bValuta,double dImporto,LPSTR szDataFattura,LPSTR szDataDecorrenza,LPSTR szGiornoParticolare,LPSTR szMeseSalto1,LPSTR szMeseSalto2,LPSTR szGiornoSpostamento)

Release 4

SCOPO: sviluppa le scadenze relative ad un tipo pagamento[26]. Le scadenze sviluppate possono essere recuperate con la funzione SPPSRVReadScadenza()

PARAMETRI:

* **szTipoPagamento:** puntatore al buffer contenente il codice del tipo pagamento da usare NULL terminated
* **bValuta:** TRUE se le scadenze da sviluppare sono in valuta (in questo caso vengono gestiti gli arrotondamenti sui centesimi), FALSE altrimenti
* **dImporto:** importo totale da suddividere
* **szDataFattura:** puntatore al buffer contenente la data del documento che origina le scadenze in formato ANSI (AAAAMMGG) NULL terminated
* **szDataDecorrenza:** puntatore al buffer contenente la data decorrenza pagamenti in formato ANSI (AAAAMMGG) NULL terminated
* **szGiornoParticolare:** puntatore al buffer contenente il giorno particolare espresso su due caratteri (GG) NULL terminated. Il dato può essere impostato al valore "\0" se non necessario.
* **szMeseSalto1:** puntatore al buffer contenente il primo mese di salto espresso su due caratteri (MM) NULL terminated. Il dato può essere impostato al valore "\0" se non necessario.
* **szMeseSalto2:** puntatore al buffer contenente il secondo mese di salto espresso su due caratteri (MM) NULL terminated. Il dato può essere impostato al valore "\0" se non necessario.
* **szGiornoSpostamento:** puntatore al buffer contenente il giorno di spostamento per le scadenze che cadono nel mese di salto espresso su due caratteri (GG) NULL terminated. Il dato può essere impostato al valore "\0" se non necessario.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **-2** errore nello sviluppo: verificare che il codice del tipo pagamento sia stato definito negli archivi
* **<N>** numero delle rate sviluppate

[26] Per la "Tipologia prima rata" definita dal tipo pagamento viene gestita solo l'opzione relativa al totale documento ripartito su tutte le rate, pertanto differenti tipologie, come ad esempio quella relativa all'imposta sulla prima rata, devono essere gestite prima di chiamare la funzione di calcolo (indicando cioè l'effettivo totale da ripartire).

int SPPSRVReadScadenza(int iNumeroScadenza,double *dImporto,LPSTR szData,LPSTR szTipo)

Release 4 **SCOPO:** recupera il valore contenuto in un campo del record attuale della tabella movimenti contabili.

PARAMETRI:

* **iNumeroScadenza:** numero della scadenza da leggere. La prima scadenza è individuata dal numero 1.
* **dImporto:** puntatore al buffer dove la funzione deve memorizzare l'importo della scadenza. Il dato viene gestito come double float
* **szData:** puntatore al buffer dove la funzione deve memorizzare la data di scadenza. Il dato viene gestito come stringa NULL terminated di 8 caratteri in formato ANSI (AAAAMMGG)
* **szTipo:** puntatore al buffer dove la funzione deve memorizzare il tipo di scadenza generata. Il dato viene gestito come stringa NULL terminated di 1 carattere.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo

Il tipo di scadenza generata puo assumere i seguenti valori:

* **0** rimessa diretta o contanti
* **1** tratta
* **2** ricevuta bancaria
* **3** cessione
* **4** paghero'
* **5** lettera di credito
* **6** tratta accettata
* **7** RiBa
* **8** ritardato pagamento
* **9** cambiale
* **A** altro pagamento
* **B** bonifico bancario

Esempio

```
SPPSRVInit(hMainWnd);
if(SPPSRVVersion()<4) {
        return(invalid_version)
}
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
SPPSRVSetEsercizio("1997");
i=SPPSRVCalcolaScadenze("TR30",FALSE,1000000.,"19970101","19970101","","08","12","09");
for(j=1;j<=i;j++) {
        k=SPPSRVReadScadenza(j,&importo,data,tipo);
}
SPPSRVExit();
```

Funzioni contenute in SPPSERV.DLL: movimenti di magazzino

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di registrazione/cancellazione di movimenti di magazzino e righe documenti (si ricorda che per SIGLA le righe documenti sono movimenti di magazzino).

int SPPSRVInitMovmagTransaction()

Release 1 **SCOPO:** inizia una nuova transazione di registrazione di movimenti di magazzino.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **7** non connesso al database aziendale o al database dati comuni
* **42** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

1. viene preparato **in memoria** un nuovo record per la tabella MOVIMAG. I campi numerici sono impostati a 0, i campi carattere sono impostati al valore NULL
2. viene preparato **in memoria** un nuovo record per la tabella MOVITAG. I campi numerici sono impostati a 0, i campi carattere sono impostati al valore NULL
3. viene acquisito un numero univoco per il raggruppamento di registrazioni
4. a partire dalla **release 8** della libreria il numero acquisito viene automaticamente impostato sul record in memoria per la gestione delle testate documenti

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndMovmagTransaction() non appena possibile.**

int SPPSRVEndMovmagTransaction()

Release 1 **SCOPO:** termina una transazione di registrazione di movimenti si magazzino.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **43** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale. Per iniziare una nuova registrazione contabile è necessario chiamare nuovamente SPPSRVInitMovmagTransaction().

int SPPSRVSetMovmagField(LPSTR szFldName, LPSTR szValue)

Release 1 **SCOPO:** imposta il contenuto del campo specificato nel record della tabella MOVIMAG prima di procedere all'aggiunta del record. La funzione deve essre chiamata per impostare tutti i valori necessari.

PARAMETRI:

* **szFldName:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValue:** puntatore al buffer contenente il valore da assegnare al campo NULL terminated.**N.B.: per i campi numerici il valore deve essere impostato come stringa di caratteri in formato ASCII..**

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza dato non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **10** campo non trovato

int SPPSRVSetMovmagQnt(LPSTR szQnt)

Release 1 **SCOPO:** imposta la quantità da movimentare su un movimento non gestito per taglia. La DLL si predispone a registrare una riga senza gestione taglie (verra' aggiunto un record solo nella tabella MOVIMAG)

PARAMETRI:

* **szQnt:** puntatore al buffer contenente la quantità da movimentare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

int SPPSRVSetMovmagEvaso(LPSTR szQnt)

Release 1 **SCOPO:** imposta la quantità evasa su un movimento non gestito per taglia. La DLL si predispone a registrare una riga senza gestione taglie (verra' aggiunto un record solo nella tabella MOVIMAG)

PARAMETRI:

* **szQnt:** puntatore al buffer contenente la quantità da impostare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata

* **7** non connesso al database aziendale o al database dati comuni

int SPPSRVSetMovTagQnt(int iTagNo,LPSTR szQnt)

Release 1 **SCOPO:** imposta la quantità da movimentare su un movimento gestito per taglia. La DLL si predispone a registrare una riga con gestione taglie (verra' aggiunto un record nella tabella MOVIMAG e uno nella tabella MOVITAG). La funzxione deve essere chiamata una volta per ognuna delle componenti per taglia da impostare.

PARAMETRI:

* **iTagNo:** numero della componente taglia da impostare (le componenti vanno da 1 a 20)
* **szQnt:** puntatore al buffer contenente la quantità da movimentare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **46** numero della componente non valido (il range possibile è 1-20)

int SPPSRVSetMovTagEvaso(int iTagNo,LPSTR szQnt)

Release 1 **SCOPO:** imposta la quantità evasa su un movimento gestito per taglia. La DLL si predispone a registrare una riga con gestione taglie (verra' aggiunto un record nella tabella MOVIMAG e uno nella tabella MOVITAG). La funzxione deve essere chiamata una volta per ognuna delle componenti per taglia da impostare.

PARAMETRI:

* **iTagNo:** numero della componente taglia da impostare (le componenti vanno da 1 a 20)
* **szQnt:** puntatore al buffer contenente la quantità da impostare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **46** numero della componente non valido (il range possibile è 1-20)

int SPPSRVAppendMovmagRecord(BOOL bCausConcat)

Release 1 **SCOPO:** appende il record corrente alle tabelle MOVIMAG/MOVITAG. Prima di procedere all'append fisico del record vengono eseguiti una serie di controlli di congruenza.

La funzione riporta automaticamente sul movimento di magazzino i flags necessari alla corretta gestione della giacenza/costistenza prelevandoli dalla causale impostata.

La funzione aggiorna automaticamente le tabelle GIAC, GIACESE,GIACTAG con i saldi di giacenza/consistenza. Se i record necessari non esistono vengono creati.

Se è riempito il campo UBICAZIONE la funzione aggiorna automaticamente la tabella GIAUBICA, con i saldi di giacenza. Se il record necessario non esiste sara' creato.

N.B. per le ubicazioni non viene eseguito senza nessun controllo di compatibilita' con l'anagarfica articoli e di magazzino.

PARAMETRI:

* **bCausConcat:** deve essere a TRUE se si desidera che l'eventuale concatenazione di causali di magazzino sia gestita, FALSE altrimenti, Se il dato è TRUE e la causale impostata è concatenata ad altre vengono registrati tanti movimenti quante sono le causali nella catena.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo

* **-1**errore generico durante l'esecuzione della movimentazione di magazzino
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **43** inizializzazione della transazione non effettuata
* **44** articolo non valido: il campo ARTICOLO non contiene il codice valido di un articolo di magazzino
* **45** causale di magazzino non valida: il campo CAUSALE non contiene il codice valido di una causale di magazzino
* **47** il campo TIPO contiene un valore non valido. I valori possibili sono "MM" per i movimenti di magazzino, "DC" per le righe dei documenti
* **48** data non valida: il campo DATA non è stato impostato. **N.B.: la funzione non effettua alcun controllo sulla validita' della data impostata.**
* **49** codice magazzino non impostato: il campo MAGAZZINO non è stato impostato. **N.B.: la funzione non effettua alcun controllo sull'esistenza del magazzino impostato. Il valore di ritorno è un "warning": l'operazione di append viene comunque eseguita.**

int SPPSRVAppendMovmagRecordEx(BOOL bCausConcat)

Release 26 **SCOPO:** è una versione estesa della precedente funzione, nel senso che oltre ad effettuare esattamente le stesse operazioni della SPPSRVAppendMovmagRecord viene anche gestito il flag di configurazione "Agg. Ordin. solo p. S.O." (aggiorna ordinato solo per scarico ordine). Pertanto se tale flag di configurazione è impostato l'ordinato cliente/fornitore sarà modificato soltanto se per la riga immessa è stato riempito il campo NUMERORIF (normalmente destinato a contenere il riferimento all'ordine). Questa opzione viene gestita esclusivamente per le righe documento di tipo bolla di carico, bolla di scarico, fattura, fattura proforma, fattura accompagnatoria e nota di credito.

Valgono le stesse considerazioni illustrate per la precedente funzione SPPSRVAppendMovmagRecord, sia per quanto riguarda le note di funzionamento che i parametri e i valori di ritorno.

int SPPSRVDeleteMovmagRecord(LPSTR szGroup,LPSTR szRow)

Release 1 **SCOPO:** la funzione consente la cancellazione di un singolo movimento di magazzino. Se la registrazione è associata ad una movimentazione per taglia anche la parte presente nella tabella MOVITAG viene cancellata. Si precisa che la funzione provvede anche all'aggiornamento delle informazioni sulla giacenza e consistenza di magazzino[27]. **Non è possibile usare questa funzione per cancellare movimenti inseriti con le funzioni precedenti prima che sia chiamata la funzione SPPSRVEndMovmagTransaction().**

PARAMETRI:

* **szGroup:** puntatore al buffer contenente il numero del raggruppamento di registrazioni che contiene il record da cancellare NULL terminated.
* **szRow:** puntatore al buffer contenente il numero di riga da cancellare all'interno del raggruppamento di registrazioni NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-2** record non trovato: la coppia szGroup,szRow non individua un record presente nella tabella MOVIMAG
* **5** lunghezza dato non valida: il numero di gruppo deve essere una stringa di 7 bytes zerofilled, il numero di riga deve essere una stringa di 5 bytes zerofilled
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni

[27] Quando viene cancellata una riga di un documento è necessario gestire direttamente anche le eventuali informazioni collegate, ad esempio se si cancella una riga di ordine è necessario operare in modo opportuno in caso di ordine evaso (parzialmente o integralmente) ecc. .

N.B.: la funzione non esegue una Commit. L'utilizzatore è responsabile della chiusura della transazione che deve essere effettuata chiamando SPPSRVDittaCommit().

long int SPPSRVReadMovmagNumber()

Release 3 **SCOPO:** la funzione ritorna il numero progressivo del raggruppamento di movimenti attualmente in corso di memorizzazione. **Attenzione**: per valori di ritorno maggiori o uguali a 10.000.000 il reale valore del numero del raggruppamento è contenuto nel campo NUMERO[28] della tabella MOVIMAG e deve essere letto con la funzione SPPSRVGetMovmagField.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

* **-1** errore generico
* **>=0** numero del gruppo di registrazioni attualmente in fase di memorizzazione

int SPPSRVDeleteMovmagSet(LPSTR szExtKey)[29]

Release 5 **SCOPO:** la funzione cancella il set di movimenti di magazzino individuati dal valore fornito della chiave esterna (campo CHIAVE_EST della tabella MOVIMAG).

PARAMETRI:

- **szExtKey:** puntatore al buffer contenente il codice della chiave esterna da cancellare NULL terminated

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **-2** chiave non trovata

int SPPSRVSetMovmagNumero(long int NewNumero)

Release 11 **SCOPO:** la funzione imposta il numero da usare per il raggruppamento di movimenti di magazzino che si sta registrando. Lo scopo è quello di consentire l'aggiunta di nuove righe a un raggruppamento di movimenti registrato in precedenza. **ATTENZIONE:** l'uso improprio di questa funzione può comportare l'illecita duplicazione di movimenti di magazzino. Il numero del raggruppamento fa infatti parte, insieme al numero di riga, della chiave univoca.

PARAMETRI:

- **NewNumero:** numero da assegnare al raggruppamento di movimenti

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **5** parametro non valido
* **43** transazione di aggiunta movimenti non iniziailizzata

int SPPSRVSetMovmagRiga(long int NewNumero)

Release 11 **SCOPO:** la funzione imposta il numero riga da usare per il raggruppamento di movimenti di

[28] Questo campo e' di tipo carattere, lungo 7 byte, pertanto valori maggiori o uguali a 10.000.000 vengono rappresentati con stringhe il cui primo elemento non e' una cifra ma una lettera, ovvero 'A000000' corrisponde a 10.000.000, 'A000001' corrisponde a 10.000.001, , 'A999999' corrisponde a 10.999.999, 'B000000' corrisponde a 11.000.000 ecc..

[29] disponibile solo per le versioni client/server della libreria

magazzino che si sta registrando. Lo scopo è quello di consentire l'aggiunta di nuove righe a un raggruppamento di movimenti registrato in precedenza. **ATTENZIONE:** l'uso improprio di questa funzione può comportare l'illecita duplicazione di movimenti di magazzino. Il numero riga del raggruppamento fa infatti parte, insieme al numero di raggruppamento, della chiave univoca.

PARAMETRI:

- **NewNumero:** numero da assegnare alla prossima riga del raggruppamento di movimenti

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **5** parametro non valido
* **43** transazione di aggiunta movimenti non iniziailizzata

int SPPSRVSetTesDocumField(LPSTR szFldName, LPSTR szValue)

Release 7 **SCOPO:** imposta il contenuto del campo specificato nel record della tabella TESDOCUM prima di procedere all'aggiunta del record. La funzione deve essre chiamata per impostare tutti i valori necessari.

PARAMETRI:

* **szFldName:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValue:** puntatore al buffer contenente il valore da assegnare al campo NULL terminated.**N.B.: per i campi numerici il valore deve essere impostato come stringa di caratteri in formato ASCII.**

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza dato non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **10** campo non trovato

int SPPSRVAppendTesDocumRecord()

Release 7 **SCOPO:** appende il record corrente alla tabella TESDOCUM. Il controllo sulla congruenza dei dati contenuti nei campi è a carico dell'utilizzatore. La funzione **non** gestisce la commit sul database.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **9** inizializzazione della transazione non effettuata

N.B.: a partire dalla versione 8 della libreria l'impostazione del campo NUMERO che serve a collegare la testata del documento alle righe presenti nella tabella MOVMAG viene eseguita automaticamente dalla chiamata SPPSRVInitMovMagTransaction.

int SPPSRVGetMovmagField(LPSTR szFiledname, LPSTR szValuebuffer, int iBuflen)

Release 4 **SCOPO:** recupera il valore contenuto in un campo del record attuale della tabella movimenti di magazzino.

PARAMETRI:

* **szFieldname:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
* **szValuebuffer:** puntatore al buffer dove il valore recuperato dovrà essere copiato
* **iBuflen:** ampiezza nmassima in caratteri del buffer dove il valore recuperato dovrà essere copiato

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **5** lunghezza buffer non valida
* **6** DLL non inizializzata
* **7** non connesso al database aziendale o al database dati comuni
* **10** nome campo non valido

N.B.: i dati numerici vengono tornati come stringhe ASCII formattate su 15 caratteri.

Chiamate alla libreria SPPSERV.DLL effettuate dalla SIGPPDLL.DLL

La libreria SPPSERV può essere usata per l'implementazione di personalizzazioni che si appoggino alla libreria SIGPPDLL.In questo caso, per le versioni client/server della procedura, si pone il problema del numero complessivo di connessioni che il client è costretto ad allocare verso il DBMS. Infatti, a run time, alle connessioni stabilite da SIGLA (almeno due) devono essere aggiunte le connessioni stabilite dalla libreria di servizio SPPSERV (ancora almeno due).

Potrebbe essere in questo caso interessante far si che la libreria SPPSERV riesca ad operare utilizzando **le stesse connessioni verso il DBMS** su cui lavora SIGLA. Allo scopo di consentire quanto esposto sono state rilasciate, a partire dalla versione 8 della libreria, delle apposite API che devono essere richiamate al posto delle corrispondenti funzioni create per una operativita' indipendente dall'esecuzione contemporanea di SIGLA:

int SPPSRVInternalInit()[30]

Release 8 **SCOPO:** inizializza la libreria in modo che essa operi sulle stesse connessioni utilizzate da SIGLA. La funzione deve essere chiamata **in alternativa** a SPPSRVInit e può essere usata solo congiuntamente con SIGPPDLL

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **2** DLL già inizializzata

int SPPSRVInternalExit()[31]

Release 8 **SCOPO:** termina l'attività della DLL ed esegue le azioni di cleanup necessarie. Il tentativo di usare altre funzioni dopo aver chiamato SPPSRVInternalExit() può comportare la generazione di errori. La funzione deve essere chiamata **in alternativa** a SPPSRVExit **solo se la libreria è stata inizializzata con la chiamata a SPPSRVInternalInit.**

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non predentemente inizializzata

int SPPSRVInternalComuniConnect(DPDbase *Comuni)[32]

Release 8 **SCOPO:** imposta la connessione stabilita da SIGLA per operare sul database dati comuni. La funzione deve essere chiamata in alternativa a SPPSRVComuniConnect **prima** di usare le

[30] disponibile solo per SIGLA

[31] disponibile solo per SIGLA

[32] disponibile solo per SIGLA

funzioni operative e può essere usata **solo** se la libreria è stata inizializzata con la chiamata SPPSRVInternalInit.La chiamata **deve** avvenire reimplementando la funzione SIGLAPPInit della libreria SIGPPDLL.

PARAMETRI:

* **Comuni:** puntatore ad un oggetto C++ di tipo DPDbase. Alla funzione **deve** essere passato come parametro l'argomento fornito da SIGLA durante l'esecuzione della funzione SIGLAPPInit

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-3** connessione non stabilita perché la versione degli archivi non corrisponde
* **3** connesione al database già stabilita in precedenza
* **4** errore durante la connessione
* **6** DLL non inizializzata

Esempio:

```
HANDLE ll;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvVersion)();
void DLLCALL SIGLAPPInit(LPSTR utente, LPSTR todaydate, LPSTR codditta, LPSTR ragsoc,  LPSTR termname,
        LPSTR dittaconnectstring,LPSTR comuniconnectstring, DPDbase *ditta, DPDbase *comuni)
        {
                if((ll = LoadLibrary("SPPSERV.DLL"))==NULL) {
                        return;
                }
                if((lpSrvInternalInit=GetProcAddress(ll,"SPPSRVInternalInit"))==NULL) {
                        return;
                }
                if((lpSrvInternalExit=GetProcAddress(ll,"SPPSRVInternalExit"))==NULL) {
                        return;
                }
                if((lpSrvInternalComuniConnect=(int(_stdcall*)(DPDbase*))
         GetProcAddress(ll,"SPPSRVInternalComuniConnect"))==NULL) {
                        return;
                }
                if((lpSrvInternalDittaConnect=(int(_stdcall*)(DPDbase*))
         GetProcAddress(ll,"SPPSRVInternalDittaConnect"))==NULL) {
                        return;
                }
                (*lpSrvInternalInit)();

                (*lpSrvInternalComuniConnect)(comuni);
                (*lpSrvInternalDittaConnect)(ditta);
        }
```

int SPPSRVInternalDittaConnect(DPDbase * Ditta)[33]

Release 8 **SCOPO:** imposta la connessione stabilita da SIGLA per operare sul database aziendale. La funzione deve essere chiamata in alternativa a SPPSRVDittaConnect **prima** di usare le funzioni operative e può essere usata **solo** se la libreria è stata inizializzata con la chiamata SPPSRVInternalInit.La chiamata **deve** avvenire reimplementando la funzione SIGLAPPInit della libreria SIGPPDLL. SPPSRVInternalDittaConnect può essere chiamata più volte nel corso della stessa applicazione. **La connessione in vigore è quella relativa all'ultima chiamata effettuata.**

La funzione procede alla lettura della tabella di configurazione della ditta (DPCONFIG) e adotta i parametri in essa impostati per l'attività sul database aziendale (lunghezza della chiave di magazzino, lunghezza codice piano dei conti etc.)

La connessione al database dati comuni **deve** essere effettuata **prima** di chiamare questa funzione.

PARAMETRI:

- **Ditta:** puntatore ad un oggetto C++ di tipo DPDbase. Alla funzione **deve** essere passato come parametro l'argomento fornito da SIGLA durante l'esecuzione della funzione SIGLAPPInit

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **-3** connessione non stabilita perché la versione degli archivi non corrisponde
* **4** errore durante la connessione
* **6** DLL non inizializzata
* **50** connessione al database dati comuni non eseguita
* **51** dati necessari all'uso dell'Eurokit non impostati

Funzioni contenute in SPPSERV.DLL: movimenti per lotto

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di registrazione/cancellazione di movimenti per lotto. Ricordiamo qui che i movimenti per lotto devono necessariamente essere associati ad un documento che può essere una Bolla di carico, una bolla di scarico, una fattura, una fattura accompagnatoria o un ordine cliente. Ricordiamo inoltre, che ad ogni rigo di bolla di carico e di ordine cliente può essere associato uno ed un solo movimento per lotto. Mentre per i restanti tipi documenti sopra citati, si possono associare fino ad un massimo di 40 righi lotto per ogni rigo documento. E' cura del programmatore rispettare tali vincoli, in quanto la dll non opera tali controlli. C'e' inoltre da sottolineare che per movimentare un articolo per lotto e/o per ubicazione l'articolo deve essere definito per lotto e/o per ubicazione. Inoltre se sono configurati i magazzini multipli, il magazzino di movimentazione deve essere definito a lotti e/o ad ubicazioni. La funzione SPPSRVAppendMovLotRecord opera il controllo sul codice articolo, rimane a cura del programmatore il controllo sul magazzino (se configurati magazzini multipli), questo per esigenze di performance.

int SPPSRVInitMovLotTransaction()

Release 13 **SCOPO:** inizia una nuova transazione di registrazione di movimenti per lotto

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **53** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

[33] disponibile solo per SIGLA

1. viene preparato **in memoria** un nuovo record per la tabella MOVLOT. I campi numerici sono impostati a 0, i campi carattere sono impostati al valore NULL
2. viene acquisito un numero univoco per la registrazione e memorizzato nel campo NUMERO

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndMovLotTransaction() non appena possibile.**

int SPPSRVInitMovLotTransactionPlus(int nnumero)

Release 13 **SCOPO:** inizia una nuova transazione di registrazione di movimenti per lotto, analogamente alla funzione SPPSRVInitMovLotTransaction, impegnando pero' nnumero progressivi numero. Questa funzione deve essere invocata nel caso di inserimento di una Bolla di Vendita, se il numero di movimenti per lotto da associare ad una certa riga, è maggiore di 1. La funzione memorizza nel campo NUMERO il primo numero e porta avanti il progressivo (contenuto nella tabella DPCONFIG) del valore passato nel parametro d'ingresso nnumero, impegnando così nnumero progressivi.

PARAMETRI:

- **nnumero** numero di progressivi che devono essere impegnati

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **53** inizializzazione già effettuata e non conclusa
- **64** nnumero non valido (maggiore di 40 oppure minore di 0)

La funzione esegue le seguenti operazioni:

1. viene preparato **in memoria** un nuovo record per la tabella MOVLOT. I campi numerici sono impostati a 0, i campi carattere sono impostati al valore NULL
2. vengono impegnati nnumero progressivi per le registrazioni e viene memorizzato nel campo NUMERO il primo progressivo acquisito.

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndMovLotTransaction() non appena possibile.**

int SPPSRVSetMovLotArticolo(LPSTR articolo)

Release 13 **SCOPO:** imposta in memoria l'articolo su cui movimentare il lotto

PARAMETRI:

- **articolo** codice articolo di magazzino

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata
- **59** articolo non valido

int SPPSRVSetMovLotTipoEffettivo(LPSTR tipoeffettivo)

Release 13 **SCOPO:** imposta in memoria il tipo efettivo del documento a cui è associato il movimento per lotto

PARAMETRI: puntatore al buffer contenente un byte che identifica il tipo documento. I valori possibili sono:

- B per Bolla di carico

- G per Bolla di scarico
- F per Fattura
- O per Ordine cliente
- A per Fattura accompagnatoria

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata

int SPPSRVSetMovLotField(LPSTR szFldName, LPSTR szValue)

Release 13 **SCOPO:** imposta il contenuto del campo specificato nel record della tabella MOVLOTTI prima di procedere all'aggiunta del record. La funzione deve essere chiamata per impostare tutti i valori necessari.

PARAMETRI:

- **szFldName:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
- **szValue:** puntatore al buffer contenente il valore da assegnare al campo NULL terminated.**N.B.: per i campi numerici il valore deve essere impostato come stringa di caratteri in formato ASCII..**

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata
- **5** lunghezza dato non valida
- **10** campo non trovato

int SPPSRVSetMovLotQnt(LPSTR szQnt)

Release 13 **SCOPO:** imposta la quantità da movimentare su un movimento non gestito per taglia. La DLL si predispone a registrare una riga senza gestione taglie.

PARAMETRI:

- **szQnt:** puntatore al buffer contenente la quantità da movimentare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata

int SPPSRVSetMovLotQntTag (int iTagNo,LPSTR szQnt)

Release 13 **SCOPO:** imposta la quantità da movimentare su un movimento gestito per taglia. La DLL si predispone a registrare una riga con gestione taglie. La funzione deve essere chiamata una volta per ognuna delle componenti per taglia da impostare.

PARAMETRI:

- **iTagNo:** numero della componente taglia da impostare (le componenti vanno da 1 a 20)

- **szQnt:** puntatore al buffer contenente la quantità da movimentare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **61** numero della componente non valido (il range possibile è 1-20)
- **52** inizializzazione della transazione non effettuata

int SPPSRVSetMovLotEvaso(LPSTR szQnt)

Release 13 **SCOPO:** imposta la quantità evasa su un movimento non gestito per taglia. La DLL si predispone a registrare una riga senza gestione taglie

PARAMETRI:

- **szQnt:** puntatore al buffer contenente la quantità da impostare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata

int SPPSRVSetMovLotEvasoTag (int iTagNo,LPSTR szQnt)

Release 13 **SCOPO:** imposta la quantità evasa su un movimento gestito per taglia. La DLL si predispone a registrare una riga con gestione taglie. La funzxione deve essere chiamata una volta per ognuna delle componenti per taglia da impostare.

PARAMETRI:

- **iTagNo:** numero della componente taglia da impostare (le componenti vanno da 1 a 20)
- **szQnt:** puntatore al buffer contenente la quantità da impostare in formato ASCII NULL terminated. La parte decimale della quantità deve essere separata da un punto.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **46** numero della componente non valido (il range possibile è 1-20)
- **52** inizializzazione della transazione non effettuata

int SPPSRVAppendMovLotRecord()

Release 13 **SCOPO:** appende il record corrente alle tabelle MOVLOT. Prima di procedere all'append fisico del record vengono eseguiti una serie di controlli di congruenza.

La funzione aggiorna automaticamente le tabelle GIALOTTI, con i saldi di giacenza/consistenza. Se i record necessari non esistono vengono creati.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **-1** errore generico durante l'esecuzione della movimentazione
- **6** DLL non inizializzata

- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata
- **54** codice lotto non valido (valore contenuto nel campo CODICE della tabella MOVLOTTI)
- **55** tipo operazione non valido (valore contenuto nel campo TIPOOP della tabella MOVLOTTI)
- **56** operazione non valida (valore contenuto nei campi OPGIACENZA e OPIMPEGNAT della tabella MOVLOTTI)
- **57** campo NUMMOVIMAG della tabella MOVLOTTI non valorizzato (deve contenere il valore del campo NUMERO della tabella MOVIMAG, del record associato al movimento per lotto)
- **58** campo RIGMOVIMAG della tabella MOVLOTTI non valorizzato (deve contenere il valore del campo RIGA della tabella MOVIMAG, del record associato al movimento per lotto)
- **59** articolo non valido
- **60** tipo effettivo del documento non valido
- **63** codice magazzino non impostato: il campo MAGAZZINO non è stato impostato. **N.B.: la funzione non effettua alcun controllo sull'esistenza del magazzino impostato. Il valore di ritorno è un "warning": l'operazione di append viene comunque eseguita.**

int SPPSRVEndMovLotTransaction()

Release 13 **SCOPO:** termina una transazione di registrazione di movimenti per lotto.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale. Per iniziare una nuova registrazione contabile è necessario chiamare nuovamente SPPSRVInitMovLotTransaction().

int SPPSRVDeleteMovLotRecord(LPSTR szNumero)

Release 13 **SCOPO:** la funzione consente la cancellazione di un singolo movimento per lotto. **Non è possibile usare questa funzione per cancellare movimenti inseriti con le funzioni precedenti prima che sia chiamata la funzione SPPSRVEndMovLotTransaction().**

PARAMETRI:

- **szNumero:** puntatore al buffer contenente il numero del record da cancellare NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **64** lunghezza dato non valida: il numero deve essere una stringa di 10 bytes zerofilled
- **65** record non trovato: il parametro szNumero non individua un record presente nella tabella MOVLOTTI

N.B.: la funzione non esegue una Commit. L'utilizzatore è responsabile della chiusura della transazione che deve essere effettuata chiamando SPPSRVDittaCommit().

int SPPSRVDelAllMovLotperMovMag(LPSTR szNumMovMag, LPSTR szRigaMovMag)

Release 13 **SCOPO:** la funzione consente la cancellazione di tutti i movimenti per lotto associati ad una riga di un documento.

Non è possibile usare questa funzione per cancellare movimenti inseriti con le funzioni precedenti prima che sia chiamata la funzione SPPSRVEndMovLotTransaction().

PARAMETRI:

- **szNumMovmag:** puntatore al buffer contenente il valore del campo numero del record di MOVIMAG collegato ai lotti da cancellare NULL terminated.
- **SzRigaMovMag:** puntatore al buffer contenente il valore del campo riga del record di MOVIMAG collegato ai lotti da cancellare NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **66** lunghezza di szNumMovMag e/o szRigaMovMag non valida: szNumMovMag deve essere una stringa di 7 bytes zerofilled, mentre szRigaNumMag deve essere una stringa di 5 bytes zerofilled
- **65** record non trovato: la coppia (szNumMovMag, szRigaMovMag) parametro szNumero non individua un record presente nella tabella MOVLOTTI

N.B.: la funzione non esegue una Commit. L'utilizzatore è responsabile della chiusura della transazione che deve essere effettuata chiamando SPPSRVDittaCommit().

long int SPPSRVReadMovLotNumber()

Release 13 **SCOPO:** la funzione ritorna il numero progressivo del movimento attualmente in corso di memorizzazione.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **-1** errore generico
- **>=0** numero di registrazione attualmente in fase di memorizzazione

int SPPSRVSetMovLotNumero(long int liNewNumero)

Release 13 **SCOPO:** la funzione imposta il numero da usare per il movimento per lotto che si sta registrando.

ATTENZIONE: l'uso improprio di questa funzione può comportare l'illecita duplicazione di movimenti per lotto. Il numero, definisce infatti, la chiave univoca.

PARAMETRI:

- **NewNumero:** numero da assegnare al raggruppamento di movimenti

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **52** transazione di aggiunta movimenti non iniziailizzata
- **5** parametro non valido

long int SPPSRVGetNewCodiceAnLot ()

Release 13 **SCOPO:** la funzione ritorna il codice progressivo da assegnare ad un nuovo lotto. Questa funzione deve essere invocata quando si deve definire un nuovo lotto, da aggiungere in Anagrafica Lotti. Il valore di ritorno, deve essere assegnato al campo CODICE della tabella ANALOTTI, nel formato: stringa di caratteri lunga 10 byte zerofilled.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **-1** errore generico
- **-2** connessione al database aziendale non effettutata
- **>=0** nuovo codice

Funzioni contenute in SPPSERV.DLL: movimenti per matricole

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di registrazione/cancellazione di movimenti per matricole. Ricordiamo qui che i movimenti per matricola devono necessariamente essere associati ad un documento che può essere una Bolla di carico, una bolla di scarico, una fattura, una fattura accompagnatoria o un ordine cliente. Ad ogni rigo dei documenti elencati precedentemente possono essere associati un numero indefinito di movimenti per matricola, saranno pari alla quantità della riga di movimento, la quantità associata ad un rigo di movimenti per matricola sarà sempre uguale ad uno, è infine da ricordare che una matricola non ha giacenza essa è invece contraddistinta da uno stato presente o assente dal magazzino. C'e' inoltre da sottolineare che per movimentare un articolo per matricola l'articolo deve essere definito per matricola. Nessuna indicazione relativa alla gestione delle matricole è presente sul magazzino, in caso di attivazione della gestione matricole tutti i magazzini saranno interpretati come gestiti a matricole. La funzione SPPSRVAppendMovMatRecord opera il controllo sul codice articolo, inoltre viene effettuato il controllo se la matricola indicata è coerente con il movimento che si stà inserendo, non sarà cioè possibile inserire su un documento di vendita un codice matricola non presente in magazzino per i documenti di carico vale il viceversa.

int SPPSRVInitMovMatTransaction()

Release 32 **SCOPO:** inizia una nuova transazione di registrazione di movimenti per lotto

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **67** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

3. viene preparato **in memoria** un nuovo record per la tabella MOVMAT. I campi carattere sono impostati al valore NULL
4. viene acquisito un numero univoco per la registrazione e memorizzato nel campo NUMERO

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndMovMatTransaction() non appena possibile.**

int SPPSRVInitMovMatTransactionPlus(int nnumero)

Release 32 **SCOPO:** inizia una nuova transazione di registrazione di movimenti per matricola, analogamente alla funzione SPPSRVInitMovMatTransaction, impegnando pero' nnumero progressivi numero. Questa funzione deve essere invocata nel caso di inserimento di un numero maggiore di 1 di movimenti per matricola. La funzione memorizza nel campo NUMERO il primo numero e porta avanti il progressivo (contenuto nella tabella DPCONFIG) del valore passato nel parametro d'ingresso nnumero, impegnando così nnumero progressivi.

PARAMETRI:

- **nnumero** numero di progressivi che devono essere impegnati

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **67** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

3. viene preparato **in memoria** un nuovo record per la tabella MOVMAT. I campi carattere sono impostati al valore NULL
4. vengono impegnati nnumero progressivi per le registrazioni e viene memorizzato nel campo NUMERO il primo progressivo acquisito.

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndMovMatTransaction() non appena possibile.**

int SPPSRVSetMovMatArticolo(LPSTR articolo)

Release 32 **SCOPO:** imposta in memoria l'articolo su cui movimentare il lotto

PARAMETRI:

- **articolo** codice articolo di magazzino

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **76** inizializzazione della transazione non effettuata
- **71** articolo non valido

int SPPSRVSetMovMatField(LPSTR szFldName, LPSTR szValue)

Release 32 **SCOPO:** imposta il contenuto del campo specificato nel record della tabella MOVMATR prima di procedere all'aggiunta del record. La funzione deve essere chiamata per impostare tutti i valori necessari.

PARAMETRI:

- **szFldName:** puntatore al buffer contenente il nome del campo da aggiornare NULL terminated
- **szValue:** puntatore al buffer contenente il valore da assegnare al campo NULL terminated.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **76** inizializzazione della transazione non effettuata

int SPPSRVAppendMovMatRecord()

Release 32 **SCOPO:** appende il record corrente alle tabelle MOVMATR. Prima di procedere all'append fisico del record vengono eseguiti una serie di controlli di congruenza.

La funzione aggiorna automaticamente lo stato della matricola nella tabella ANMATR cambiandolo da presente ad assente in magazzino nel caso di documento di vendita, da assente a presente nel caso di documento di carico. In caso di documento di carico le matricole che non esistono vengono create.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **76** inizializzazione della transazione non effettuata
- **77** tipo documento inesistente
- **72** tipo effettivo del documento non valido
- **73** per i documenti di carico, matricola già presente in magazzino
- **74** per i documenti di scarico, matricola non presente in magazzino
- **68** codice matricola non valido (valore contenuto nel campo CODICE della tabella MOVMATR)
- **69** tipo operazione non valido (valore contenuto nel campo TIPOOP della tabella MOVMATR)
- **70** campo NUMMOVIMAG della tabella MOVMATR non valorizzato (deve contenere il valore del campo NUMERO della tabella MOVIMAG, del record associato al movimento per matricola)
- **71** articolo non valido
- **75** codice magazzino non impostato: il campo MAGAZZINO non è stato impostato. **N.B.: la funzione non effettua alcun controllo sull'esistenza del magazzino impostato. Il valore di ritorno è un "warning": l'operazione di append viene comunque eseguita.**

int SPPSRVEndMovMatTransaction()

Release 32 **SCOPO:** termina una transazione di registrazione di movimenti per lotto.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **76** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale. Per iniziare una nuova registrazione contabile è necessario chiamare nuovamente SPPSRVInitMovLotTransaction().

long int SPPSRVReadMovMatNumber()

Release 32 **SCOPO:** la funzione ritorna il numero progressivo del movimento attualmente in corso di memorizzazione.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **0** successo
- **7** non connesso al database aziendale o al database dati comuni
- **76** inizializzazione della transazione non effettuata

int SPPSRVSetMovMatNumero(long int liNewNumero)

Release 32 **SCOPO:** la funzione imposta il numero da usare per il movimento per matricola che si sta registrando.

ATTENZIONE: l'uso improprio di questa funzione può comportare l'illecita duplicazione di movimenti per matricola. Il numero, infatti, è parte della chiave univoca.

PARAMETRI:

- **liNewNumero:** numero da assegnare al raggruppamento di movimenti

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **0** successo
- **7** non connesso al database aziendale o al database dati comuni
- **76** transazione di aggiunta movimenti non iniziailizzata
- **5** parametro non valido

int SPPSRVSetMovMatRiga(long int liNewRiga)

Release 32

SCOPO: la funzione imposta il numero di riga da usare per il movimento per matricola che si sta registrando.

ATTENZIONE: l'uso improprio di questa funzione può comportare l'illecita duplicazione di movimenti per matricola. Il numero di riga, infatti, è parte della chiave univoca insieme al campo numero.

PARAMETRI:

- **liNewRiga:** numero di riga da assegnare al movimento

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **0** successo
- **7** non connesso al database aziendale o al database dati comuni
- **76** transazione di aggiunta movimenti non iniziailizzata
- **5** parametro non valido

long int SPPSRVGetNewCodiceAnMat()

Release 32

SCOPO: la funzione ritorna il codice progressivo da assegnare ad una nuova matricola. Questa funzione deve essere invocata quando si deve definire una nuova matricola, da aggiungere in Anagrafica Matricole. Il valore di ritorno, deve essere assegnato al campo CODICE della tabella ANAMATR, nel formato: stringa di caratteri lunga 10 byte zerofilled.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero lungo che può assumere i seguenti valori:

- **-1** errore generico
- **-3** connessione al database aziendale non effettutata
- **>=0** nuovo codice

Funzioni contenute in SPPSERV.DLL: ricalcolo saldi di magazzino

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di ricalcalo dei saldi di magazzino.

int SPPSRVInitRicMagazzinoTransaction()

Release 41

SCOPO: inizia una nuova transazione di ricalcolo dei saldi di magazzino

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **79** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

5. Vengono inizializzate le procedure di ricalcolo per i saldi di magazzino normali e per taglia

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndRicMagazzinoTransaction() non appena possibile.**

int SPPSRVRicalcolaMagazzino(LPSTR szArticoloIniziale, LPSTR szArticoloFinale)

Release 41 **SCOPO:** Effettua il ricalcolo dei saldi di magazzino normale e per taglia. I parametri articolo iniziale e articolo finale permettono di limitare il ricalcolo ad un determinato gruppo di articoli; se i parametri sono nulli il ricalcolo verrà effettuato per tutti gli articoli.

ATTENZIONE: questa funzione può elaborare un'elevata quantità di dati ed è stata pensata per essere utilizzata quando non è eseguita altra attività da parte degli utenti, l'uso improprio di questa funzione può comportare notevoli problemi di performance.

PARAMETRI:

* **szArticoloIniziale:** puntatore al buffer contenente il codice iniziale dell'articolo di magazzino NULL terminated
* **szArticolofinale:** puntatore al buffer contenente il codice finale dell'articolo di magazzino NULL terminated

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **79** inizializzazione già effettuata e non conclusa
- **83** errore nel ricalcolo dei saldi di magazzino
- **84** errore nel ricalcolo dei saldi di magazzino per taglia

La funzione esegue le seguenti operazioni:

5. La funzione effettua la procedura di ricalcolo dei saldi di magazzino normale e per taglia

int SPPSRVEndRicMagazzinoTransaction()

Release 41 **SCOPO:** termina una transazione di ricalcolo dei saldi di magazzino.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **79** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

Funzioni contenute in SPPSERV.DLL: ricalcolo saldi contabili

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di ricalcolo dei saldi contabili.

int SPPSRVInitRicContabilitaTransaction()

Release 41 **SCOPO:** inizia una nuova transazione di ricalcolo dei saldi contabili

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo

- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **81** inizializzazione già effettuata e non conclusa

La funzione esegue le seguenti operazioni:

6. Vengono inizializzate le procedure di ricalcolo per i saldi contabili.

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndRicContabilitaTransaction() non appena possibile.**

int SPPSRVRicalcolaContabilita()

Release 41 **SCOPO:** Effettua il ricalcolo dei saldi contabili.

ATTENZIONE: questa funzione può elaborare un'elevata quantità di dati ed è stata pensata per essere utilizzata quando non è eseguita altra attività da parte degli utenti, l'uso improprio di questa funzione può comportare notevoli problemi di performance.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **81** inizializzazione già effettuata e non conclusa
- **82** errore nel ricalcolo dei saldi contabili

La funzione esegue le seguenti operazioni:

6. La funzione effettua la procedura di ricalcolo dei saldi contabili.

int SPPSRVEndRicContabilitaTransaction()

Release 41 **SCOPO:** termina una transazione di ricalcolo dei saldi contabili.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **80** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

Funzioni contenute in SPPSERV.DLL: calcolo prezzo, sconti/maggioraz.

Le funzioni di seguito determinano i prezzi netto e lordo, gli sconti/maggiorazioni di riga e gli sconti di testata (sconto incondizionato e sconto pagamento). Sono disponibili anche altre due funzioni per la lettura dei vari valori precedentemente calcolati. L'invocazione delle funzioni di lettura prima dell'esecuzione del calcolo non provocano alcun errore ma i valori delle varie grandezze sono tutti pari a zero.

int SPPSRVCalcolaPrezzi(LPSTR szTipo, LPSTR szCliFor, LPSTR szArticolo, double Qty)

Release 43 **SCOPO:** determina i prezzi netto/lordo e gli sconti/maggiorazioni gestendo in modo automatico la logica di calcolo decisa in base alle impostazioni definite dai cìvalori delle opzioni di configurazione di SIGLA. Tutti i parametri necessari sono stabiliti in base ai valori presenti in anagrafica cliente/fornitore o articoli di magazzino (è il caso ad esempio della categoria di sconto, del codice listino, del codice del tipo pagamento e della valuta). I valori dei prezzi possono essere recuperati con la funzione SPPSRVReadPrezzo mentre quelli i valori degli sconti e maggiorazioni con la funzione SPPSRVReadSconto.

PARAMETRI:

* **szTipo:** puntatore al buffer contenente il tipo codice cliente/fornitore NULL terminated (i valori ammessi sono ‘C’ per i lcienti o ‘F’ per i fornitori)
* **szCliFor:** puntatore al buffer contenente il codice del cliente/fornitore NULL terminated
* **szArticolo:** puntatore al buffer contenente il codice dell’articolo di magazzino NULL terminated
* **Qty:** quantità (necessaria nel caso dei listini a fasce di quantità).

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **85** codice cliente non inserito
- **86** codice articolo non inserito

int SPPSRVCalcolaPrezziEx(LPSTR szTipo, LPSTR szCliFor, LPSTR szArticolo, double Qty, LPSTR szTipoPag, LPSTR szValuta, LPSTR szListino)

Release 43 **SCOPO:** determina i prezzi netto/lordo e gli sconti/maggiorazioni gestendo in modo automatico la logica di calcolo decisa in base alle impostazioni definite dai cìvalori delle opzioni di configurazione di SIGLA. Tutti i parametri necessari non indicati sono stabiliti in base ai valori presenti in anagrafica cliente/fornitore o articoli di magazzino (è il caso ad esempio della categoria di sconto, del codice listino, del codice del tipo pagamento e della valuta). I valori dei prezzi possono essere recuperati con la funzione SPPSRVReadPrezzo mentre quelli i valori degli sconti e maggiorazioni con la funzione SPPSRVReadSconto.

PARAMETRI:

* **szTipo:** puntatore al buffer contenente il tipo codice cliente/fornitore NULL terminated (i valori ammessi sono ‘C’ per i lcienti o ‘F’ per i fornitori)
* **szCliFor:** puntatore al buffer contenente il codice del cliente/fornitore NULL terminated
* **szArticolo:** puntatore al buffer contenente il codice dell’articolo di magazzino NULL terminated
* **Qty:** quantità (necessaria nel caso dei listini a fasce di quantità).
* **szTipoPag:** puntatore al buffer contenente il codice del tipo pagamento NULL terminated
* **szValuta:** puntatore al buffer contenente il codice della valuta NULL terminated
* **szListino:** puntatore al buffer contenente il codice del listino NULL terminated

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **85** codice cliente non inserito
- **86** codice articolo non inserito

int SPPSRVReadPrezzo(int iTipo, double *dPrezzo)

Release 43 **SCOPO:** recupera il valore del prezzo precedentemente calcolato dalla funzione SPPSRVCalcolaPrezzi.

PARAMETRI:

* **iTipo:** tipo di prezzo da leggere
    * **1** prezzo netto in valuta di conto (euro)
    * **2** prezzo netto in valuta estera
    * **3** prezzo lordo in valuta di conto (euro)

    * **4** prezzo lordo in valuta estera
* **dPrezzo:** puntatore al buffer dove la funzione deve memorizzare l'importo del prezzo. Il dato viene gestito come double float

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **87** tipo prezzo non valido

int SPPSRVReadSconto(int iTipo, double *dValore)

Release 43 **SCOPO:** recupera il valore degli sconti/maggiorazioni riga, dello sconto cliente/pagamento precedentemente calcolati dalla funzione SPPSRVCalcolaPrezzi.

PARAMETRI:

* **iTipo:** tipo di sconto/maggiorazione da leggere
    * **1-5** sconto riga n. 1-5
    * **6-7** maggiorazione riga n.1-2
    * **8** sconto incondizionato cliente/fornitore
    * **9** sconto pagamento
* **dValore:** puntatore al buffer dove la funzione deve memorizzare il valore dello sconto/maggiorazione. Il dato viene gestito come double float

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **87** tipo sconto/maggiorazione non valido

Funzioni contenute in SPPSERV.DLL: importazione documenti

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di importazione dei documenti. Vengono utilizzati gli stessi file (posizionati nella stessa cartella) descritti nel paragrafo "Importazione di massa di dati esterni".

int SPPSRVInitImpDocumentiTransaction()

Release 43 **SCOPO:** inizia una nuova transazione di importazione dei documenti

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **91** inizializzazione già effettuata e non conclusa

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndImpDocumentiTransaction() non appena possibile.**

int SPPSRVImportazioneDocumenti(BOOL bDiProva, BOOL bDaMovimento)

Release 43 **SCOPO:** Effettua l'importazione dei documenti. I parametri `bDiProva` e `bDaMovimento` permettono rispettivamente di effettuare l'importazione dei documenti in prova, producendo così solo un file di log che riporta eventuali errori, e di utilizzare il codice del magazzino e i valori dei flag che definiscono la movimentazione dei totalizzatori di magazzino dal movimento anziché dalla causale. Il file di log, che viene comunque creato indipendentemente dal valore dei parametri, verrà memorizzato nella directory nella quale si trova la dll.

PARAMETRI:

* **bDiProva** importazione "di prova"

* **bDaMovimenti** codice magazzino e flag dal movimento

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **90** inizializzazione già effettuata e non conclusa
- **93** errore nell'importazione dei documernti

int SPPSRVEndImpDocumentiTransaction()

Release 43 **SCOPO:** termina una transazione di importazione dei documenti.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **90** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

Funzioni contenute in SPPSERV.DLL: importazione movimenti di magazzino

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di importazione dei movimenti di magazzino. Viene utilizzato lo stesso file (posizionato nella stessa cartella) descritto nel paragrafo "Importazione di massa di dati esterni".

int SPPSRVInitImpMovMagazzinoTransaction()

Release 43 **SCOPO:** inizia una nuova transazione di importazione dei movimenti di magazzino

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **89** inizializzazione già effettuata e non conclusa

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndImpMovMagazzinoTransaction() non appena possibile.**

int SPPSRVImportazioneMovMagazzino(BOOL bDiProva, BOOL bDaMovimento)

Release 43 **SCOPO:** Effettua l'importazione dei movimenti di magazzino. I parametri `bDiProva` e `bDaMovimento` permettono di effettuare l'importazione dei movimenti di magazzino in prova, producendo così solo un file di log che riporta eventuali errori, e di utilizzare il codice del magazzino e i valori dei flag che definiscono la movimentazione dei totalizzatori di magazzino dal movimento anziché dalla causale. Il file di log, che viene comunque creato indipendentemente dal valore dei parametri, verrà memorizzato nella directory nella quale si trova la dll.

PARAMETRI:

* **bDiProva** importazione "di prova"

* **bDaMovimenti** codice magazzino e flag dal movimento

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **88** inizializzazione già effettuata e non conclusa
- **92** errore nell'importazione dei movimenti di magazzino

int SPPSRVEndImpMovMagazzinoTransaction()

Release 43 **SCOPO:** termina una transazione di importazione dei movimenti di magazzino.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **88** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

Funzioni contenute in SPPSERV.DLL: importazione movimenti contabili

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di importazione dei movimenti contabili. Vengono utilizzati gli stessi file (posizionati nella stessa cartella) descritti nel paragrafo "Importazione di massa di dati esterni".

int SPPSRVInitImpMovContabiliTransaction()

Release 43 **SCOPO:** inizia una nuova transazione di importazione dei movimenti contabili

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **95** inizializzazione già effettuata e non conclusa

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndImpMovContabiliTransaction() non appena possibile.**

int SPPSRVImportazioneMovContabili(BOOL bDiProva, BOOL bStampaErrori, BOOL bControlloSequenza, LPSTR szControlloQuadratura)

Release 43 **SCOPO:** Effettua l'importazione dei movimenti contabili. Il parametro `bDiProva` indica che l'importazione viene fatta in modalità simulata, il parametro `bStampaErrori` indica che si vuole che nel log prodotto siano riportati eventuali errori rilevati dalla procedura, il parametro `bControlloSequenza` indica che si vuole che la procedura controlli che i movimenti importati non generino fuori sequenza e il parametro `controlloquad` indica il tipo di controolo di quadratura dare/avere da eseguire (può assumere i seguenti valori: 'N' per indicare che non si vuole che venga effettuato alcun controllo di quadratura, 'L' se si vuole che venga effettuato il controllo della quadratura in Lire, 'E' se si vuole che venga effettuato il controllo della quadratura in Euro). Il file di log, che viene comunque creato indipendentemente dal valore dei parametri, verrà memorizzato nella directory nella quale si trova la dll.

PARAMETRI:

* **bDiProva** importazione "di prova"
* **bStampaErrori** evidenzia errori nel file di log

* **bControlloSequenza** controllo del fuori sequenza nei protocolli
* **szControlloQuadratura** puntatore al buffer che contiene il tipo di controllo di quadratura che si vuole venga effettuato sui movimenti importati.

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **94** inizializzazione già effettuata e non conclusa
- **96** errore nell'importazione dei movimenti contabili

int SPPSRVEndImpMovContabiliTransaction()

Release 43 **SCOPO:** termina una transazione di importazione dei movimenti contabili.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **94** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

Funzioni contenute in SPPSERV.DLL: ricalcolo saldi lotti

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di ricalcalo dei saldi di magazzino.

int SPPSRVInitRicLottiTransaction()

Release 56 **SCOPO:** inizia una nuova transazione di ricalcolo dei saldi lotti

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **100** inizializzazione già effettuata

La funzione esegue le seguenti operazioni:

7. Vengono inizializzate le procedure di ricalcolo per i saldi dei lotti

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndRicLottiTransaction() non appena possibile.**

int SPPSRVRicalcolaLotti()

Release 56 **SCOPO:** Effettua il ricalcolo dei saldi lotti.

ATTENZIONE: questa funzione può elaborare un'elevata quantità di dati ed è stata pensata per essere utilizzata quando non è eseguita altra attività da parte degli utenti, l'uso improprio di questa funzione può comportare notevoli problemi di performance.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **101** inizializzazione non effettuata
- **102** errore nel ricalcolo dei saldi lotti

La funzione esegue le seguenti operazioni:

7. La funzione effettua la procedura di ricalcolo dei saldi lotti

int SPPSRVEndRicLottiTransaction()

Release 56 **SCOPO:** termina una transazione di ricalcolo dei saldi lotti.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **101** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

Funzioni contenute in SPPSERV.DLL: ricalcolo fido

Le funzioni di seguito dettagliate sono state disegnate per facilitare l'attività di ricalcolo del fido clienti/fornitori.

int SPPSRVInitRicFidoTransaction()

Release 56 **SCOPO:** inizia una nuova transazione di ricalcolo del fido

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **97** inizializzazione già effettuata

La funzione esegue le seguenti operazioni:

8. Vengono inizializzate le procedure di ricalcolo per il fido.

La funzione inizia una nuova transazione. Durante la transazione alcune risorse vengono bloccate **è quindi fondamentale terminare la transazione usando la funzione SPPSRVEndRicFidoTransaction() non appena possibile.**

int SPPSRVRicalcolaFido(LPSTR szCliForIniziale, LPSTR szCliForFinale)

Release 56 **SCOPO:** Effettua il ricalcolo del fido clienti/fornitori.

ATTENZIONE: questa funzione può elaborare un'elevata quantità di dati ed è stata pensata per essere utilizzata quando non è eseguita altra attività da parte degli utenti, l'uso improprio di questa funzione può comportare notevoli problemi di performance.

PARAMETRI:

* **szCliForIniziale:** puntatore al buffer contenente il codice iniziale del cliente/fornitore NULL terminated

* **szCliForfinale:** puntatore al buffer contenente il codice finale del cliente/fornitore NULL terminated

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata (non invocata SPPSRVInit)
- **7** non connesso al database aziendale o al database dati comuni
- **98** inizializzazione non effettuata
- **99** errore nel ricalcolo del fido

La funzione esegue le seguenti operazioni:

8. La funzione effettua la procedura di ricalcolo del fido clienti/fornitori.

int SPPSRVEndRicFidoTransaction()

Release 56 **SCOPO:** termina una transazione di ricalcolo del fido clienti/fornitori.

PARAMETRI: nessuno

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

- **0** successo
- **6** DLL non inizializzata
- **7** non connesso al database aziendale o al database dati comuni
- **98** inizializzazione della transazione non effettuata

La funzione esegue una Commit sul database aziendale.

int SPPSRVCalcolaFidoDocumento(char *Numero, double *MerceDaConsegnare, double * MerceDaFatturare)

Release 65 **SCOPO:** Effettua il ricalcolo del fido clienti/fornitori per un solo documento. La funzione è parte del ricalcolo fido e in particolare permette il ricalcolo del valore FIDO di un singolo documento.**La funzione non esegue nessun tipo di aggiornamento del database.**

PARAMETRI:

* **Numero** puntatore a carattere che deve contenere la chiave univoca del documento in TESDOCUM (TESDOCUM.NUMERO).
* **Merce da consegnare** puntatore al buffer dove la funzione deve memorizzare l'importo relativo al totale della merce ordinata. Il dato viene gestito come double float
* **Merce da fatturare** puntatore al buffer dove la funzione deve memorizzare l'importo relativo al totale della merce spedita. Il dato viene gestito come double float**.**

VALORI DI RITORNO: la funzione genera un codice di ritorno intero che può assumere i seguenti valori:

* **0** successo
* **6** DLL non inizializzata (non invocata SPPSRVInit)
* **7** non connesso al database aziendale o al database dati comuni
* **103** documento non trovato. "Numero" non corrisponde a nessun record in TESDOCUM.

Per ogni documento solo uno dei due importi di ritorno **Merce da consegnare, Merce da fatturare** può essere diverso da zero:

- se NUMERO individua un ordine "Merce da consegnare" contiene il totale della merce ancora da consegnare nel documento e "Merce da Fatturare" vale zero;
- se NUMERO individua una bolla di scarico "Merce da consegnare" vale zero e "Merce da Fatturare" contiene il valore complessivo della bolla.

Per valore del fido calcolato si intende l'importo che sarebbe calcolato dalla funzione ricalcolo del FIDO se esiste quel solo documento da ricalcolare. **Il documento deve essere presente nella base dati**.
Questo significa che una personalizzazione che esegue l'immissione di un documento può, dopo il salvataggio, chiamare la funzione e aggiornare di conseguenza i valori relativi al fido in anagrafica clienti/fornitori.
Questa funzione è destinata all'uso in fase di immissione di un nuovo documento ma, qualora si intendesse usarla per implementare l'attività di revisione è, ovviamente, necessario eseguire un calcolo preventivo del documento in fase di revisione in modo da conoscere il valore attuale per il fido del documento prima che la personalizzazione effettui modifiche al documento stesso. Dopo che la personalizzazione ha registrato il documento modificato si dovrà rieseguire la funzione per conoscere il valore attuale del documento modificato. La somma algebrica in segno fra i due valori deve essere usata per aggiornare in anagrafica clienti/fornitori i valori del fido.
La stessa cosa deve essere fatta nel caso la personalizzazione implementi anche funzioni di evasione. In questo caso ogni documento coinvolto nell'evasione deve essere calcolato prima e dopo l'evasione e l'anagrafica deve essere aggiornata con la somma algebrica dei valori ottenuti.

Esempi

Di seguito sono riportati alcuni esempi di utilizzo delle funzioni contenute nella DLL SPPSERV.

Registrazione di movimenti di prima nota

Il codice C che segue realizza la registrazione di una scrittura in partita doppia cassa a banca:

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
// testa la versione
if(SPPSRVVersion()!=1) return(invalid_version)
// stabilisce la connessione al database
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
SPPSRVSetEsercizio("2012");
SPPSRVSetUtente("SIGLA   ");
SPPSRVSetTodayDate("20120412");
SPPSRVSetEuro(true);
// inizializza la transazione
SPPSRVInitMovcoTransaction();
// registra il primo record
SPPSRVSetMovcoField("DATAREG","20120412");
SPPSRVSetMovcoField("ESEREGISTR","2012");
SPPSRVSetMovcoField("DATACOMPET","20120412");
SPPSRVSetMovcoField("DTCOMPCONT","20120412");
SPPSRVSetMovcoField("ESECOMPET","2012");
SPPSRVSetMovcoField("SOTTOCONTO","cassa     ");
SPPSRVSetMovcoField("CONTROPART","banca     ");
SPPSRVSetMovcoField("C_F","A");
SPPSRVSetMovcoField("SEGNO","D");
SPPSRVSetMovcoField("CAUSALE","PAG");
SPPSRVSetMovcoField("EIMPORTO","100.00");
// imposta il campo CASO a "prima nota non iva"
SPPSRVSetMovcoField("CASO","4");
```

```
SPPSRVCheckMovcoRecord();
SPPSRVAppendMovcoRecord();
// imposta il secondo record
// (i campi impostati non vengono resettati da SPPSRVAppendNovcoRecord)
SPPSRVSetMovcoField("SOTTOCONTO","banca     ");
SPPSRVSetMovcoField("CONTROPART","cassa     ");
SPPSRVSetMovcoField("SEGNO","A");
SPPSRVCheckMovcoRecord();
SPPSRVAppendMovcoRecord();
// termina la transazione
SPPSRVEndMovcoTransaction();
// resetta la DLL
SPPSRVExit();
```

Registrazione di una fattura di vendita

Il codice C che segue realizza la registrazione di una fattura di vendita. Viene gestita anche l'acquisizione del numero di protocollo iva.

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
// testa la versione
if(SPPSRVVersion()!=1) return(invalid_version)
// stabilisce la connessione al database
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
// attivazione delle transazioni
SPPSRVActivateTransaction();
#endif
SPPSRVSetEsercizio("2012");
SPPSRVSetUtente("SIGLA   ");
SPPSRVSetTodayDate("20120412");
SPPSRVSetEuro(true);
// acquisisce il protocollo iva dal numeratore FV e lo usa sia come protocollo che come numero
// documento (N.B.: SPPSRVGetNum esegue automaticamente una Commit)
nprot=SPPSRVGetNum("FV");
// riporta il numero acquistito in formato ASCII
sprintf(numeroprot,"%07ld",nprot);
sprintf(numerodoc,"%010ld",nprot);
// inizializza la transazione
SPPSRVInitMovcoTransaction();
// registra il primo record sul cliente
SPPSRVSetMovcoField("DATAREG","20120412");
SPPSRVSetMovcoField("ESEREGISTR","2012");
SPPSRVSetMovcoField("DATACOMPET","20120412");
SPPSRVSetMovcoField("DTCOMPCONT","20120412");
```

```
SPPSRVSetMovcoField("ESECOMPET","2012");
SPPSRVSetMovcoField("SOTTOCONTO","deltaphi  ");
SPPSRVSetMovcoField("C_F","C");
SPPSRVSetMovcoField("SEGNO","D");
// imposta la causale contabile
SPPSRVSetMovcoField("CAUSALE","FV1");
SPPSRVSetMovcoField("EIMPORTO","121.00");
SPPSRVSetMovcoField("EIMPFATTUR","121.00");
// imposta data e numero del documento
SPPSRVSetMovcoField("DATAPROTOC","20120412");
SPPSRVSetMovcoField("DATADOCUM","20120412");
SPPSRVSetMovcoField("NUMDOCUM",numerodoc);
SPPSRVSetMovcoField("NUMPROTOC",numeroprot);
// imposta il codice del registro IVA (dipende dalla causale)
SPPSRVSetMovcoField("REGIVA","FV");
// imposta il tipo documento IVA (dipende dalla causale)
SPPSRVSetMovcoField("TIPODOCIVA","FV");
// imposta il campo CASO a "prima nota iva"
SPPSRVSetMovcoField("CASO","0");
// appende il record
SPPSRVCheckMovcoRecord();
SPPSRVAppendMovcoRecord();
// registra il secondo record sul ricavo
SPPSRVSetMovcoField("SOTTOCONTO","vendite   ");
SPPSRVSetMovcoField("C_F","A");
SPPSRVSetMovcoField("SEGNO","A");
SPPSRVSetMovcoField("EIMPORTO","100.00");
// i campi non ipostati mantengono il valore del record precedente
// appende il record
SPPSRVCheckMovcoRecord();
SPPSRVAppendMovcoRecord();
// registra il terzo record sull'iva
SPPSRVSetMovcoField("SOTTOCONTO","ivavend   ");
SPPSRVSetMovcoField("SEGNO","A");
SPPSRVSetMovcoField("EIMPORTO","21.00");
SPPSRVSetMovcoField("EIMPORTOIV","21.00");
SPPSRVSetMovcoField("MOV_IVA_SN","S");
// appende il movimento contabile
SPPSRVCheckMovcoRecord();
SPPSRVAppendMovcoRecord();
// registra un record su MOVIVA
SPPSRVSetMovIvaField("SOTTOCONTO","deltaphi  ");
SPPSRVSetMovIvaField("C_F","C");
SPPSRVSetMovIvaField("TIPODOCIVA","FV");
SPPSRVSetMovIvaField("REGISTRO","FV");
SPPSRVSetMovIvaField("NUMPROTOC",numeroprot);
SPPSRVSetMovIvaField("NUMERODOC",numerodoc);
SPPSRVSetMovIvaField("DATAREG","20120412");
SPPSRVSetMovIvaField("DATADOC","20120412");
SPPSRVSetMovIvaField("EIMPONIBIL","100.00");
SPPSRVSetMovIvaField("CODIVA","0021");
```

```
SPPSRVSetMovIvaField("EIMPOSTA","21.00");
SPPSRVSetMovIvaField("ETOTFATTUR","121.00");
// appende il movimento iva
SPPSRVCheckMovIvaRecord();
SPPSRVAppendMovIvaRecord();
// termina la transazione
SPPSRVEndMovcoTransaction();
// resetta la DLL
SPPSRVExit();
```

Registrazione di un movimento di magazzino

Il codice C che segue realizza la registrazione di un movimento di magazzino su un articolo gestito a taglie:

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
// testa la versione
if(SPPSRVVersion()!=1) {
     return(invalid_version)
}
// stabilisce la connessione al database
#ifdef STARTSUITE
if(SPPSRVComuniConnect("C:\\SIGLAPP\\DATICOM")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("C:\\SIGLAPP\\DITTA1")!=0) return(invalid_connection);
#endif
#ifdef ODBC
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
#endif
SPPSRVSetEsercizio("1997");
SPPSRVSetUtente("SIGLA   ");
SPPSRVSetTodayDate("19970513");
// inisia la transazione
SPPSRVInitMovmagTransaction();
// imposta i campi
SPPSRVSetMovmagField("DATA","19970513");
SPPSRVSetMovmagField("ARTICOLO","SCARPE");
SPPSRVSetMovmagField("CAUSALE","ACQ");
SPPSRVSetMovmagField("TIPO","MM");
SPPSRVSetMovmagField("MAGAZZINO","001");
SPPSRVSetMovmagField("PRNETTO","90000");
SPPSRVSetMovtagQnt(1,"1");
SPPSRVSetMovtagQnt(2,"5");
SPPSRVSetMovtagQnt(3,"3");
// appende il record
SPPSRVAppendMovmagRecord(FALSE);
// chiude la transazione
SPPSRVEndMovmagTransaction()
// resetta la DLL
SPPSRVExit();
```

Registrazione di un movimento per lotto

Il codice C che segue realizza la registrazione di un movimento per lotto collegato ad una riga di bolla di vendita identificata dalla coppia (numero,riga) con il seguente valore(‘0000234’,’00001’)

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
// testa la versione
if(SPPSRVVersion()!=13) {
     return(invalid_version)
}
// stabilisce la connessione al database (solo versione Client/Server)
if(SPPSRVComuniConnect(“DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO”)!=0) return(invalid_connection);
if(SPPSRVDittaConnect(“DSN=DITTA1;UID=DDD;PWD=DDD”)!=0) return(invalid_connection);

SPPSRVSetEsercizio(“1997”);
SPPSRVSetUtente("SIGLA   ");
SPPSRVSetTodayDate("19970513");

// inisia la transazione
SPPSRVInitMovLotTransaction();

// imposta i campi il codice articolo e il tipo effettivo
SPPSRVSetMovLotArticolo(“ART1”);
SPPSRVSetMovLotTipoEffettivo(“G”);

// imposta i campi
SPPSRVSetMovLotField("CODICE",”0000000022”);
SPPSRVSetMovLotField("MAGAZZINO",”001”);
SPPSRVSetMovLotField("UBICAZIONE",”0000000111”);
SPPSRVSetMovLotField("DTREGISTRA",”19991225”);
SPPSRVSetMovLotField("NUMDOCCF",”0000263”);
SPPSRVSetMovLotField("DTDOCCLIFO",”19991218”);
SPPSRVSetMovLotField("TIPOOP",”S”);
SPPSRVSetMovLotField("OPGIACENZA",”-”);
SPPSRVSetMovLotField("OPIMPEGNAT",”N”);
SPPSRVSetMovLotField("NUMMOVIMAG",”0000234”);
SPPSRVSetMovLotField("RIGMOVIMAG",”00001”);
SPPSRVSetMovLotQnt(“1.2”);
     // appende il record
SPPSRVAppendMovLotRecord();
// chiude la transazione
SPPSRVEndMovLotTransaction();
// resetta la DLL
SPPSRVExit();
```

Registrazione di più movimenti per lotto durante un'unica transazione

Il codice Delphi che segue realizza la registrazione di due movimenti per lotto collegati ad una riga di bolla di vendita identificata dalla coppia (numero,riga) con il seguente valore(‘0000234’,’00001’)

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
```

```
// testa la versione
if(SPPSRVVersion()!=13) then
     return(invalid_version);

// stabilisce la connessione al database (solo versione Client/Server)
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);
SPPSRVSetEsercizio("1997");
SPPSRVSetUtente("SIGLA   ");
SPPSRVSetTodayDate("19970513");
// inizia la transazione
SPPSRVInitMovLotTransactionPlus(2);
// legge il primo numero restituito da SPPSRVInitMovLotTransactionPlus
num:=SPPSRVReadMovLotNumber();
i:=0;
while (i<2) do begin
     // imposta i campi il codice articolo e il tipo effettivo
     SPPSRVSetMovLotArticolo("ART1");
     SPPSRVSetMovLotTipoEffettivo("G");
     // imposta i campi
     SPPSRVSetMovLotField("CODICE","0000000022");
     SPPSRVSetMovLotField("MAGAZZINO","001");
     SPPSRVSetMovLotField("UBICAZIONE","0000000111");
     SPPSRVSetMovLotField("DTREGISTRA","19991225");
     SPPSRVSetMovLotField("NUMDOCCF","0000263");
     SPPSRVSetMovLotField("DTDOCCLIFO","19991218");
     SPPSRVSetMovLotField("TIPOOP","S");
     SPPSRVSetMovLotField("OPGIACENZA","-");
     SPPSRVSetMovLotField("OPIMPEGNAT","N");
     SPPSRVSetMovLotField("NUMMOVIMAG","0000234");
     SPPSRVSetMovLotField("RIGMOVIMAG","00001");
     SPPSRVSetMovLotQnt("1.2");
     SPPSRVAppendMovLotRecord();
     i:=i+1;
     SPPSRVSetMovLotNumber(num+i);
end;
// chiude la transazione
SPPSRVEndMovLotTransaction();
// resetta la DLL
SPPSRVExit();
```

Registrazione di un movimento per matricola

Il codice C che segue realizza la registrazione di un movimento per matricola collegato ad una riga di bolla di vendita identificata dal numero ('0000234')

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
// testa la versione
if(SPPSRVVersion()!=13) {
     return(invalid_version)
```

```
}
// stabilisce la connessione al database (solo versione Client/Server)
if(SPPSRVComuniConnect("DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD")!=0) return(invalid_connection);

SPPSRVSetEsercizio("1997");
SPPSRVSetUtente("SIGLA   ");
SPPSRVSetTodayDate("19970513");

// inizia la transazione
SPPSRVInitMovMatTransaction();

// imposta i campi il codice articolo e riga
SPPSRVSetMovMatArticolo('ART1');
SPPSRVSetMovMatRiga('00000');

// imposta i campi
SPPSRVSetMovMatField("MATRICOLA","00000000000000000022");
SPPSRVSetMovMatField ("MAGAZZINO","001");
SPPSRVSetMovMatField ("UBICAZIONE","0000000111");
SPPSRVSetMovMatField ("TIPOOP","-");
SPPSRVSetMovMatField ("C_F","C");
SPPSRVSetMovMatField ("CODCLIFOR","CLIENTE");
SPPSRVSetMovMatField ("NUMMOVIMAG","0000234");
SPPSRVSetMovMatField ("NUMDOCUM","0000333");
SPPSRVSetMovMatField ("DATADOCUM","20051010");
SPPSRVSetMovMatField ("TIPOPROTOC","BV");
SPPSRVSetMovMatField ("DATAPROTOC","20051010");
SPPSRVSetMovMatField ("NUMPROTOC","0000111");
    // appende il record
SPPSRVAppendMovMatRecord();
// chiude la transazione
SPPSRVEndMovMatTransaction();
// resetta la DLL
SPPSRVExit();
```

Registrazione di più movimenti per matricola durante un'unica transazione

Il codice Delphi che segue realizza la registrazione di due movimenti per matricola collegati a due righe di bolla di vendita identificata dal numero ('0000234')

```
// inizializza la DLL
SPPSRVInit(hMainWnd);
// testa la versione
if(SPPSRVVersion()!=13) then
    return(invalid_version);

// stabilisce la connessione al database (solo versione Client/Server)
if(SPPSRVComuniConnect('DSN=SIGLAPP;UID=PIPPO;PWD=PIPPO")!=0) return(invalid_connection);
if(SPPSRVDittaConnect("DSN=DITTA1;UID=DDD;PWD=DDD')!=0) return(invalid_connection);
SPPSRVSetEsercizio('1997');
SPPSRVSetUtente("SIGLA   ');
```

```
SPPSRVSetTodayDate('19970513');
// inizia la transazione
SPPSRVInitMovMatTransactionPlus(2);
// legge il primo numero restituito da SPPSRVInitMovLotTransactionPlus
num:=SPPSRVReadMovMatNumber();
i:=0;
while (i<2) do begin
    // imposta i campi il codice articolo e il tipo effettivo
    SPPSRVSetMovMatArticolo('ART1');
    SPPSRVSetMovMatRiga('00000');
    // imposta i campi
    SPPSRVSetMovMatField('MATRICOLA','00000000000000000022');
    SPPSRVSetMovMatField ('MAGAZZINO','001');
    SPPSRVSetMovMatField ('UBICAZIONE','0000000111');
    SPPSRVSetMovMatField ('TIPOOP','-');
    SPPSRVSetMovMatField ('C_F','C');
    SPPSRVSetMovMatField ('CODCLIFOR','CLIENTE');
    SPPSRVSetMovMatField ('NUMMOVIMAG','0000234');
    SPPSRVSetMovMatField ('NUMDOCUM','0000333');
    SPPSRVSetMovMatField ('DATADOCUM','20051010');
    SPPSRVSetMovMatField ('TIPOPROTOC','BV');
    SPPSRVSetMovMatField ('DATAPROTOC','20051010');
    SPPSRVSetMovMatField ('NUMPROTOC','0000111');
    SPPSRVAppendMovLotRecord();
    i:=i+1;
    SPPSRVSetMovMatNumber(num+i);
end;
// chiude la transazione
SPPSRVEndMovMatTransaction();
// resetta la DLL
SPPSRVExit();
```

Ricalcolo dei saldi di magazzino normale e per taglia.

Il codice C che segue realizza il ricalcolo dei saldi di magazzino normale e per taglia per tutti i codici articolo compresi fra APTIVA e TAST. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL *lpSrvInitRicMagazzino)();
int (FAR PASCAL *lpSrvRicMagazzino)(LPSTR articoloiniziale, LPSTR articolofinale);
int (FAR PASCAL *lpSrvEndRicMagazzino)();

void DLLCALL SIGLAPPInit(LPSTR utente,
                    LPSTR todaydate,
```

```
                    LPSTR codditta,
                    LPSTR ragsoc,
                    LPSTR termname,
                    LPSTR dittaconnectstring,
                    LPSTR comuniconnectstring,
                    DPDbase *ditta,
                    DPDbase *comuni)
{
    if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
            return;
    }
    if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
            return;
    }
    if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
            return;
    }
    if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
            return;
    }
    if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
            return;
    }
    (* lpSrvInternalInit)();
    (* lpSrvInternalComuniConnect)(comuni);
    (* lpSrvInternalDittaConnect)(ditta);
}


BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
    char termine[132];


    switch(theActionID) {
            case xxx:
                    if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                            return TRUE;
                    }
                    if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetUtente"))==NULL) {
                            return TRUE;
                    }
                    if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
                            return TRUE;
                    }

    if((lpSrvInitRicMagazzino=GetProcAddress(Il,"SPPSRVInitRicMagazzinoTransaction"))==NULL) {
                            return TRUE;
                    }
```

```
                    if((lpSrvRicMagazzino=(int(_stdcal*)(LPSTR,LPSTR))
GetProcAddress(Il,"SPPSRVRicalcolaMagazzino"))==NULL) {
                            return TRUE;
                    }

    if((lpSrvEndRicMagazzino=GetProcAddress(Il,"SPPSRVEndRicMagazzinoTransaction"))==NULL) {
                            return TRUE;
                    }

                    (* lpSrvSetEsercizio)("2005");
                    (* lpSrvSetUtente)("SIGLA  ");
                    (* lpSetToDayDate)("20051027");
                    (* lpSrvInitRicMagazzino)();
                    (* lpSrvRicMagazzino)("APTIVA", "TAST");
                    (* lpSrvEndRicMagazzino)();
                    (* lpSrvInternalExit)();
                    strcpy(termine, "fine");
                    MessageBox(NULL,termine,"SIGPPDLL",MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
                    return TRUE;
                    break;
    }
    return TRUE;
}
```

Ricalcolo dei saldi contabili.

Il codice C che segue realizza il ricalcolo dei saldi contabili. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL *lpSrvInitRicContabilita)();
int (FAR PASCAL *lpSrvRicContabilita)();
int (FAR PASCAL *lpSrvEndRicContabilita)();

void DLLCALL SIGLAPPInit(LPSTR utente,
                     LPSTR todaydate,
                     LPSTR codditta,
                     LPSTR ragsoc,
                     LPSTR termname,
                     LPSTR dittaconnectstring,
                     LPSTR comuniconnectstring,
                     DPDbase *ditta,
                     DPDbase *comuni)
{
    if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
```

```
            return;
        }
        if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
            return;
        }
        if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
            return;
        }
        if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
            return;
        }
        if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
            return;
        }
        (* lpSrvInternalInit)();
        (* lpSrvInternalComuniConnect)(comuni);
        (* lpSrvInternalDittaConnect)(ditta);
}


BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        char termine[132];


        switch(theActionID) {
            case xxx:
                if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                    return TRUE;
                }
                if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                    return TRUE;
                }
                if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
                    return TRUE;
                }

        if((lpSrvInitRicContabilita=GetProcAddress(Il,"SPPSRVInitRicContabilitaTransaction"))==NULL)
{
                    return TRUE;
                }

        if((lpSrvRicContabilita=GetProcAddress(Il,"SPPSRVRicalcolaContabilita"))==NULL) {
                    return TRUE;
                }

        if((lpSrvEndRicContabilita=GetProcAddress(Il,"SPPSRVEndRicContabilitaTransaction"))==NULL) {
                    return TRUE;
                }


                (* lpSrvSetEsercizio)("2005");
```

```
                        (* lpSrvSetUtente)("SIGLA  ");
                        (* lpSetToDayDate)("20051027");
                        (* lpSrvInitRicContabilita)();
                        (* lpSrvRicContabilita)();
                        (* lpSrvEndRicContabilita)();
                        (* lpSrvInternalExit)();
                        strcpy(termine, "fine");
                        MessageBox(NULL,termine,"SIGPPDLL",MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
                        return TRUE;
                        break;
        }
        return TRUE;
    }
```

Importazione dei documenti

Il codice che segue realizza l'importazione dei documenti di prova e prendendo il magazzino e i flag relativi ai totalizzatori dal movimento. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL * lpSrvInitImpDocumentiTransaction)();
int (FAR PASCAL * lpSrvImportazioneDocumenti)(BOOL diprova, BOOL damovimento);
int (FAR PASCAL * lpSrvEndImpDocumentiTransaction)();

void DLLCALL SIGLAPPInit(LPSTR utente,
                         LPSTR todaydate,
                         LPSTR codditta,
                         LPSTR ragsoc,
                         LPSTR termname,
                         LPSTR dittaconnectstring,
                         LPSTR comuniconnectstring,
                         DPDbase *ditta,
                         DPDbase *comuni)
{
        if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
                return;
        }
        if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
                return;
        }
        if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
                return;
        }
```

```
	if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
		return;
	}
	if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
		return;
	}
	(* lpSrvInternalInit)();
	(* lpSrvInternalComuniConnect)(comuni);
	(* lpSrvInternalDittaConnect)(ditta);
}


BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
	char termine[322];


	switch(theActionID) {
		case xxx:
			if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
				return TRUE;
			}
			if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetUtente"))==NULL) {
				return TRUE;
			}
			if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
				return TRUE;
			}
		if((lpSrvInitImpDocumentiTransaction=GetProcAddress(Il,"
SPPSRVInitImpDocumentiTransaction"))==NULL) {
				return TRUE;
			}
			if((lpSrvImportazioneDocumenti=(int(_stdcal*)(BOOL,BOOL)) GetProcAddress(Il,"
SPPSRVImportazioneDocumenti"))==NULL) {
				return TRUE;
			}
			if((lpSrvEndImpDocumentiTransaction=GetProcAddress(Il,"
SPPSRVEndImpDocumentiTransaction"))==NULL) {
				return TRUE;
			}

			(* lpSrvSetEsercizio)("2006");
			(* lpSrvSetUtente)("SIGLA  ");
			(* lpSetToDayDate)("20060323");
			(*lpSrvInitImpDocumentiTransaction)();
			(*lpSrvImportazioneDocumenti)(TRUE,TRUE);
			(*lpSrvEndImpDocumentiTransaction)();
			(* lpSrvInternalExit)();
			return TRUE;
			break;
	}
```

```
        return TRUE;
}
```

Importazione dei movimenti di magazzino

Il codice C che segue realizza l'importazione dei movimenti di magazzino di prova e prendendo il magazzino e i flag relativi ai totalizzatori dal movimento. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL * lpSrvInitImpMovMagazzinoTransaction)();
int (FAR PASCAL * lpSrvImportazioneMovMagazzino)(BOOL diprova, BOOL damovimento);
int (FAR PASCAL * lpSrvEndImpMovMagazzinoTransaction)();

void DLLCALL SIGLAPPInit(LPSTR utente,
                         LPSTR todaydate,
                         LPSTR codditta,
                         LPSTR ragsoc,
                         LPSTR termname,
                         LPSTR dittaconnectstring,
                         LPSTR comuniconnectstring,
                         DPDbase *ditta,
                         DPDbase *comuni)
{
        if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
                return;
        }
        if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
                return;
        }
        if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
                return;
        }
        if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
                return;
        }
        if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
                return;
        }
        (* lpSrvInternalInit)();
        (* lpSrvInternalComuniConnect)(comuni);
        (* lpSrvInternalDittaConnect)(ditta);
```

```
}

BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        char termine[322];

        switch(theActionID) {
                case xxx:
                        if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetUtente"))==NULL) {
                                return TRUE;
                        }
                        if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
                                return TRUE;
                        }
                if((lpSrvInitImpMovMagazzinoTransaction=GetProcAddress(Il,"
SPPSRVInitImpMovMagazzinoTransaction"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvImportazioneMovMagazzino=(int(_stdcal*)(BOOL,BOOL))
GetProcAddress(Il," SPPSRVImportazioneMovMagazzino"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvEndImpMovMagazzinoTransaction=GetProcAddress(Il,"
SPPSRVEndImpMovMagazzinoTransaction"))==NULL) {
                                return TRUE;
                        }

                        (* lpSrvSetEsercizio)("2006");
                        (* lpSrvSetUtente)("SIGLA  ");
                        (* lpSetToDayDate)("20060323");
                        (*lpSrvInitImpMovMagazzinoTransaction)();
                        (*lpSrvImportazioneMovMagazzino)(TRUE,TRUE);
                        (*lpSrvEndImpMovMagazzinoTransaction)();
                        (* lpSrvInternalExit)();
                        return TRUE;
                        break;
        }
        return TRUE;
}
```

Importazione dei movimenti contabili.

Il codice C che segue realizza l'importazione dei movimenti contabili di prova, stampando gli errori e effettuando il controllo di sequenza e di quadratura in Euro. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
```

```
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL * lpSrvInitImpMovContabiliTransaction)();
int (FAR PASCAL * lpSrvImportazioneMovContabili)(BOOL diprova, BOOL stampaerrori,BOOL
controlloseq,LPSTR controlloquad);
int (FAR PASCAL * lpSrvEndImpMovContabiliTransaction)();

void DLLCALL SIGLAPPInit(LPSTR utente,
                         LPSTR todaydate,
                         LPSTR codditta,
                         LPSTR ragsoc,
                         LPSTR termname,
                         LPSTR dittaconnectstring,
                         LPSTR comuniconnectstring,
                         DPDbase *ditta,
                         DPDbase *comuni)
{
        if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
                return;
        }
        if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
                return;
        }
        if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
                return;
        }
        if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
                return;
        }
        if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
                return;
        }
        (* lpSrvInternalInit)();
        (* lpSrvInternalComuniConnect)(comuni);
        (* lpSrvInternalDittaConnect)(ditta);
}

BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        char termine[322];

        switch(theActionID) {
                case xxx:
                        if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
```

```
                                return TRUE;
                        }
                        if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetUtente"))==NULL) {
                                return TRUE;
                        }
                        if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
                                return TRUE;
                        }
                if((lpSrvInitImpMovContabiliTransaction=GetProcAddress(Il,"
SPPSRVInitImpMovContabiliTransaction"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvImportazioneMovContabili=(int(_stdcal*)(BOOL,BOOL,BOOL,LPSTR))
GetProcAddress(Il," SPPSRVImportazioneMovContabili"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvEndImpMovContabiliTransaction=GetProcAddress(Il,"
SPPSRVEndImpMovContabiliTransaction"))==NULL) {
                                return TRUE;
                        }

                        (* lpSrvSetEsercizio)("2006");
                        (* lpSrvSetUtente)("SIGLA  ");
                        (* lpSetToDayDate)("20060323");
                        (*lpSrvInitImpMovContabiliTransaction)();
                        (*lpSrvImportazioneMovContabili)(TRUE,TRUE,TRUE,'E');
                        (*lpSrvEndImpMovContabiliTransaction)();
                        (* lpSrvInternalExit)();
                        return TRUE;
                        break;
        }
        return TRUE;
}
```

Ricalcolo dei saldi lotti

Il codice che segue realizza il ricalcolo dei saldi lotti per tutti i codici articolo gestiti a lotti. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL *lpSrvInitRicLotti)();
int (FAR PASCAL *lpSrvRicLotti)();
int (FAR PASCAL *lpSrvEndRicLotti)();
```

```
void DLLCALL SIGLAPPInit(LPSTR utente,
                         LPSTR todaydate,
                         LPSTR codditta,
                         LPSTR ragsoc,
                         LPSTR termname,
                         LPSTR dittaconnectstring,
                         LPSTR comuniconnectstring,
                         DPDbase *ditta,
                         DPDbase *comuni)
{
        if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
                return;
        }
        if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
                return;
        }
        if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
                return;
        }
        if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
                return;
        }
        if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
                return;
        }
        (* lpSrvInternalInit)();
        (* lpSrvInternalComuniConnect)(comuni);
        (* lpSrvInternalDittaConnect)(ditta);
}

BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        char termine[132];

        switch(theActionID) {
                case xxx:
                        if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetUtente"))==NULL) {
                                return TRUE;
                        }
                        if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
                                return TRUE;
                        }

        if((lpSrvInitRicLotti=GetProcAddress(Il,"SPPSRVInitRicLottiTransaction"))==NULL) {
                                return TRUE;
```

```
                        }
                        if((lpSrvRicLotti=(int(_stdcal*)())
GetProcAddress(Il,"SPPSRVRicalcolaLotti"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvEndRicLotti=GetProcAddress(Il,"SPPSRVEndRicLottiTransaction"))==NULL)
{
                                return TRUE;
                        }

                        (* lpSrvSetEsercizio)("2005");
                        (* lpSrvSetUtente)("SIGLA  ");
                        (* lpSetToDayDate)("20051027");
                        (* lpSrvInitRicLotti)();
                        (* lpSrvRicLotti)();
                        (* lpSrvEndRicLotti)();
                        (* lpSrvInternalExit)();
                        strcpy(termine, "fine");
                        MessageBox(NULL,termine,"SIGPPDLL",MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
                        return TRUE;
                        break;
        }
        return TRUE;
}
```

Ricalcolo del fido clienti/fornitori.

Il codice C che segue realizza il ricalcolo del fido clienti/fornitori. Il codice è pensato per essere eseguito all'interno della sigppdll le funzioni verranno chiamate dalla siglappsaveobject potrà essere utilizzata una qualsiasi uscita della stessa all'interno del programma.

```
HANDLE Il;
int (FAR PASCAL *lpSrvInternalInit)();
int (FAR PASCAL *lpSrvInternalExit)();
int (FAR PASCAL *lpSrvInternalComuniConnect)(DPDbase *);
int (FAR PASCAL *lpSrvInternalDittaConnect)(DPDbase *);
int (FAR PASCAL *lpSrvSetEsercizio)(LPSTR lpszesercizio);
int (FAR PASCAL *lpSrvSetUtente)(LPSTR lpszutente);
int (FAR PASCAL *lpSetToDayDate)(LPSTR lpszdata);
int (FAR PASCAL *lpSrvInitRicFido)();
int (FAR PASCAL *lpSrvRicFido)();
int (FAR PASCAL *lpSrvEndRicFido)();

void DLLCALL SIGLAPPInit(LPSTR utente,
                         LPSTR todaydate,
                         LPSTR codditta,
                         LPSTR ragsoc,
                         LPSTR termname,
                         LPSTR dittaconnectstring,
                         LPSTR comuniconnectstring,
                         DPDbase *ditta,
                         DPDbase *comuni)
```

```
{
        if((Il=LoadLibrary("SPPSERV.DLL"))==NULL) {
                return;
        }
        if((lpSrvInternalInit=GetProcAddress(Il,"SPPSRVInternalInit"))==NULL) {
                return;
        }
        if((lpSrvInternalExit=GetProcAddress(Il,"SPPSRVInternalExit"))==NULL) {
                return;
        }
        if((lpSrvInternalComuniConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalComuniConnect"))==NULL) {
                return;
        }
        if((lpSrvInternalDittaConnect=(int(_stdcall *)(DPDbase *))
GetProcAddress(Il,"SPPSRVInternalDittaConnect"))==NULL) {
                return;
        }
        (* lpSrvInternalInit)();
        (* lpSrvInternalComuniConnect)(comuni);
        (* lpSrvInternalDittaConnect)(ditta);
}


BOOL DLLCALL SIGLAPPSaveObject(DPObject *theObject,int theActionID)
{
        char termine[132];


        switch(theActionID) {
                case xxx:
                        if((lpSrvSetEsercizio=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvSetUtente=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetEsercizio"))==NULL) {
                                return TRUE;
                        }
                        if((lpSetToDayDate=(int(_stdcall *)(LPSTR))
GetProcAddress(Il,"SPPSRVSetTodayDate"))==NULL) {
                                return TRUE;
                        }
                        if((lpSrvInitRicFido=GetProcAddress(Il,"SPPSRVInitRicFidoTransaction"))==NULL)
{
                                return TRUE;
                        }
                        if((lpSrvRicFido=GetProcAddress(Il,"SPPSRVRicalcolaFido"))==NULL) {
                                return TRUE;
                        }

        if((lpSrvEndRicContabilita=GetProcAddress(Il,"SPPSRVEndRicFidoTransaction"))==NULL) {
                                return TRUE;
                        }
```

```
                        (* lpSrvSetEsercizio)("2005");
                        (* lpSrvSetUtente)("SIGLA  ");
                        (* lpSetToDayDate)("20051027");
                        (* lpSrvInitRicFido)();
                        (* lpSrvRicFido)();
                        (* lpSrvEndRicFido)();
                        (* lpSrvInternalExit)();
                        strcpy(termine, "fine");
                        MessageBox(NULL,termine,"SIGPPDLL",MB_ICONEXCLAMATION|MB_OK|MB_APPLMODAL);
                        return TRUE;
                        break;
        }
        return TRUE;
}
```

Uso della DLL da Visual Basic

La libreria SPPSERV può essere utilizzata da Visual Basic . L'esempio che segue illustra la dichiarazione delle funzioni e la loro chiamata:

```
Public Declare Function SPPSRVInit Lib "sppserv" (ByVal hwnd As Integer) As Integer
Public Declare Function SPPSRVVersion Lib "sppserv" () As Integer
Public Declare Function SPPSRVSetEsercizio Lib "sppserv" (ByVal lpstr As String) As Integer
Public Declare Function SPPSRVComuniConnect Lib "sppserv" (ByVal lpstr As String) As Integer
Public Declare Function SPPSRVDittaConnect Lib "sppserv" (ByVal lpstr As String) As Integer
Public Declare Function SPPSRVExit Lib "sppserv" () As Integer
…
SPPSRVInit(Form1.hwnd)
Text2.Text = SPPSRVVersion
…
Text1.Text = SPPSRVComuniConnect("DSN=SIGLAPP")
Text2.Text = SPPSRVDittaConnect("DSN=DITTA1")
```

Uso della DLL da Borland Delphi

La libreria SPPSERV può essere utilizzata da Delphi . Nel seguito vengono indicati due esempi che illustrano la dichiarazione delle funzioni e la loro chiamata.

Esempio 1.

```
function SPPSRVInit(hwnd: longint): word; stdcall; far; external 'sppserv.dll';
function SPPSRVComuniConnect(connect: pchar): word; stdcall; far; external 'sppserv.dll';
function SPPSRVDittaConnect(connect: pchar): word; stdcall; far; external 'sppserv.dll';
function SPPSRVExit: word; stdcall; far; external 'sppserv.dll';
…
Edit1.Text := inttostr(SPPSRVInit (self.handle));
Edit2.Text := IntToStr(SPPSRVComuniConnect ('DSN=SIGLAPP'));
Edit3.Text := IntToStr(SPPSRVDittaConnect ('DSN=NATUR'));
```

Esempio 2.

```
hmod:integer;
initspp: function(hwnd: integer):word; stdcall;
```

```
comunispp: function(connstr: Pchar):word; stdcall;
dittaspp: function(connstr: Pchar):word; stdcall;
…
hmod := LoadLibrary('SPPSERV.DLL');
initspp := GetProcAddress(hmod,'SPPSRVInit');
comunispp := GetProcAddress(hmod,'SPPSRVComuniConnect');
dittaspp := GetProcAddress(hmod,'SPPSRVDittaConnect');
exitspp := GetProcAddress(hmod,'SPPSRVExit');
Edit1.Text := inttostr(initspp(self.handle));
Edit2.Text := IntToStr(comunispp('DSN=SIGLAPP'));
Edit3.Text := IntToStr(dittaspp('DSN=NATUR'));
…
FreeLibrary(hmod);
```

Un esempio in delphi di uso delle funzioni principali della libreria è scaricabile all'indirizzo http://www.deltaphi.it/go?sv100 in forma eseguibile (chiamasppserv.e) e sorgente (chiamasppserv.s).

Tabelle di configurazione

La procedura di configurazione di SIGLA memorizza i parametri di configurazione generali e quelli per ciascuna ditta gestita all'interno di opportune tabelle.

Tabella di configurazione applicazione

I dati di configurazione dell'applicazione vengono memorizzati da SIGLA all'interno della tabella **DPAPPCON** contenuta nel database dati comuni. Di seguito vengono documentati i records presenti all'interno della tabella.

Campo CODICE	Campo STATO	Campo NDATA	Descrizione
15000	'N' **sempre**	non significativo	**non gestito - ignorare**
15001	'S' se è attiva la gestione utenti, 'N' altrimenti	non significativo	gestione utenti
15002	'S' se è attiva la clausola 'Order By'sui dati comuni	non significativo	clausola order by
15003	'S' se devono essere usati statements SQL preparati, 'N' altrimenti	non significativo	statements preparati
15004	'S' se è attiva la conversione automatica in caratteri maiuscoli, 'N' altrimenti	non significativo	conversione in maiuscolo
15005	'S' se è attivo un controllo di *robustezza* sulle password utente, 'N' altrimenti	non significativo	password sicure
16000	non significativo	Numero (0-3)	default per tipo ricerca 0 - maggiore od uguale 1 - minore od uguale 2 - inizia per 3 - contiene
30000	non significativo	non significativo	Subdirectory di riferimento per gli aggiornamenti
50000	'S' se è attivo il modulo EuroKit per i dati comuni, 'N' altrimenti	non significativo	eurokit dati comuni attivo
80000	non significativo	Numero	numeratore tabella valute
90000	non significativo	Numero	progressivo di numerazione dei file del modulo Generatore Report
99998	non significativo	release database dati comuni	copia del numero di release database dati comuni
99999	non significativo	release database dati comuni	numero di release database dati comuni (gestito dalla procedura di configurazione per eseguire l'allineamento automatico degli archivi)

Tabella di configurazione ditte

I dati di configurazione di ciascuna ditta vengono memorizzati da SIGLA all'interno della tabella **DPCONFIG** contenuta nel database dati aziendali. Di seguito vengono documentati i records presenti all'interno della tabella.

Campo CODICE	Campo STATO/DATA	Campo NDATA	Descrizione
1	'S' se sono attive le transazioni sui dati comuni, 'N' altrimenti	non significativo	transazioni su dati comuni
2	'S' se sono attive le transazioni sui dati ditta, 'N' altrimenti	non significativo	transazioni su dati ditta
3	'S' se è attivo il locking pessimistico, 'N' altrimenti	non significativo	locking pessimistico
4	'S' se è attivo il lock cursori sui dati comuni,'N' altrimenti	non significativo	lock cursori dati comuni
5	S' se è attivo il lock cursori sui dati ditta,'N' altrimenti	non significativo	lock cursori dati ditta
6	'S' se è attiva la clausola 'Order By'sui dati ditta	non significativo	clausola order by su dati ditta
7	'S' se devono essere usati statements SQL preparati, 'N' altrimenti	non significativo	statements preparati
8	'S' se in stampa le migliaia devono essere separate con il punto anziche' con la virgola	non significativo	separare migliaia con punto
10	'S' se per i dati comuni vale l'ordinamento EBCDIC	non significativo	ordinamento EBCDIC su dati comuni
11	'S' se per i dati ditta vale l'ordinamento EBCDIC	non significativo	ordinamento EBCDIC su dati ditta
12	'S' se devono essere generati automaticamente i codici clienti/fornitori	non significativo	generazione automatica codici clienti/fornitori
10000	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione raggruppamenti magazzino
10001	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione magazzini multipli
10002	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione indirizzo su tabella magazzini
10003	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione raggruppamenti fiscali
10004	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione tabella imballi
10005	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione gruppi merceologici
10006	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione famiglie merceologiche
10007	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione sottofamiglie merceologiche
10008	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione ubicazioni
10009	'S' se la funzione è attiva,'N' altrimenti	non significativo	gestione taglie

10010	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione tabella sconti
10011	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione tabella raggruppamento causali
10012	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione offerte
10013	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione listini
10014	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione codici a barre
10015	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione unita’ di misura tecnica
10016	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione vuoti
10017	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione contrassegno iva
10018	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione numero pagina listino e progressivo in pagina
10019	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione descrizioni in lingua su articoli di magazzino
10020	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione codifica articoli cliente/fornitore
10021	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione condizioni particolari di acquisto/vendita
10022	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione provvigioni agente
10023	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione provvigioni capozona
10024	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	controllo EAN su barcode
10025	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	controllo giacenze negative
10026	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	controllo prezzi a zero in immissione documenti
10027	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	controllo sottoscorta
10028	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione marchi
10029	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	attivazione contabilita’ analitica
10030	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	attivazione classi di costo
10031	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	attivazione varianti su distinte base
10032	‘S’ se la funzione è attiva,’N’ altrimenti	non significativo	gestione aggregazione articoli per listini
10033	‘S’ se la funzione è attiva,’N’	non significativo	proposta ora e data in

	altrimenti		documenti
10501	non significativo	numero	numero dimensioni ubicazioni
10502	descrizione	non significativo	descrizione prima dimensione ubicazioni
10503	descrizione	non significativo	descrizione seconda dimensione ubicazioni
10504	descrizione	non significativo	descrizione terza dimensione ubicazioni
10505	descrizione	non significativo	descrizione quarta dimensione ubicazioni
10506	descrizione	non significativo	descrizione quinta dimensione ubicazioni
10550	non significativo	numero	numero massimo di taglie per riga
10570	non significativo	numero	trasporto di default per le bolle: 0=mittente 1=destinatario 2=vettore
10600	non significativo	numero	lunghezza codice articolo di magazzino
10601	non significativo	numero	numero decimali su prezzi in lire
10602	non significativo	numero	numero decimali su prezzi in valuta
10603	non significativo	numero	numero decimali su quantità di magazzino
10604	‘S’ se la chiave è numerica,’N’ altrimenti	non significativo	chiave di magazzino numerica
10605	non significativo	numero	lunghezza codici a barre
10606	non significativo	numero	numero sconti riga
10607	non significativo	numero	numero decimali su importi in lire **ANCORA NON GESTITO**
10608	non significativo	numero	numero maggiorazioni riga
10700	codice causale	non significativo	causale apertura magazzino
10800	‘S’ se la chiave è numerica,’N’ altrimenti	non significativo	codice piano dei conti numerico
10801	non significativo	numero	lunghezza codice piano dei conti
10850	non significativo	numero	numero classificazioni centri di costo
10851	descrizione	non significativo	descrizione prima classificazione centri di costo
10852	descrizione	non significativo	descrizione seconda classificazione centri di

			costo
10853	descrizione	non significativo	descrizione terza classificazione centri di costo
10854	descrizione	non significativo	descrizione quarta classificazione centri di costo
10855	descrizione	non significativo	descrizione quinta classificazione centri di costo
10856	descrizione	non significativo	descrizione sesta classificazione centri di costo
10870	non significativo	numero	numero classificazioni immagini aziendali
10871	descrizione	non significativo	descrizione prima classificazione immagini aziendali
10872	descrizione	non significativo	descrizione seconda classificazione immagini aziendali
10873	descrizione	non significativo	descrizione terza classificazione immagini aziendali
10874	descrizione	non significativo	descrizione quarta classificazione immagini aziendali
10875	descrizione	non significativo	descrizione quinta classificazione immagini aziendali
10876	descrizione	non significativo	descrizione sesta classificazione immagini aziendali
10900	‘S’ se il regime iva è trimestrale, ‘N’ se è mensile	non significativo	regime iva
90000	non significativo	numero	numeratore per immagni e documenti office
90001	non significativo	numero	numeratore per movimenti di magazzino
90002	non significativo	numero	numeratore per movimenti contabili
90003	non significativo	numero	numeratore per distinte effetti
90004	non significativo	numero	numeratore per documenti fax
90005	non significativo	numero	**non gestito**
90006	non significativo	numero	**non gestito**
90007	non significativo	numero	numeratore per modelli INTRA
90010	stringa	non significativo	path per file office

90100	stringa	non significativo	città' intestazione lettere
91050	'S' se la funzione è attiva,'N' altrimenti	non significativo	aggacio write(WIN) / editor testi (OS/2)
91051	'S' se la funzione è attiva,'N' altrimenti	non significativo	aggancio paintbrush (WIN) / icon edit (OS/2)
91052	'S' se la funzione è attiva,'N' altrimenti	non significativo	aggancio Word (WIN) / Ami Pro (OS/2)
91053	'S' se la funzione è attiva,'N' altrimenti	non significativo	aggancio Excel (WIN) /1,2,3 (OS/2)
91054	'S' se la funzione è attiva,'N' altrimenti	non significativo	aggancio a registratore di suoni (solo WIN)
91055	'S' se la funzione è attiva,'N' altrimenti	non significativo	aggancio a acquisizione e visualizzazione immagini (solo WIN)
91056	'S' se la funzione è attiva,'N' altrimenti	non significativo	attivazione telefonia assistita (solo WIN)
91057	'S' se la funzione è attiva,'N' altrimenti	non significativo	attivazione integrazione fax
91058	stringa	non significativo	prefisso telefonico locale
91059	stringa	non significativo	uscita centralino
91060	stringa	non significativo	pathname file copertina per fax
91100	'S' se la funzione è attiva,'N' altrimenti	non significativo	attivazione stored procedures **NON ANCORA GESTITO**
90007	non significativo	numero	numeratore per elenchi INTRA
99999	non significativo	numero	numero release archivi

Attività sul database

Questa parte della documentazione illustra alcuni esempi di attività sul database gestito da SIGLA. Lo scopo è quello di chiarire i concetti guida necessari ad implementare le operazioni fondamentali.

Gestione numeratori

L'acquisizione di un numeratore deve avvenire garantendo l'univocita' del dato. Cio' avviene sfruttando le capacita' transazionali del DBMS usato e la gestione del locking delle righe attuata durante la transazione.

Il seguente esempio mostra come SIGLA acquisice il numero da assegnare al prossimo movimento di magazzino (l'acquisizione avviene **prima** di memorizzare il movimento). La struttura della tabella **DPCONFIG è** illustrata nel file di documentazione TECNOTES.CHM mentre il contenuto della tabella è documentato nel presente manuale (tabella di configurazione ditte).

```
begin transaction
        UPDATE DPCONFIG SET NDATA=NDATA+1 WHERE CODICE=90001
        SELECT NDATA FROM DPCONFIG WHERE CODICE=90001
        /* il valore contenuto in NDATA deve essere decrementato di 1 prima di essere usato */
end transaction
```

Registrazione movimenti di magazzino

La registrazione nel database aziendale di movimenti di magazzino deve avvenire nel seguente modo:

* recuperare il numero da assegnare al raggruppamento di movimenti da registrare dal record con codice **90001** della tabella DPCONFIG (vedi "Gestione numeratori" nel presente manuale)
* inserire il record relativo al movimento da registrare nella tabella **MOVIMAG** avendo cura di impostare nel campo RIGA il progressivo della riga all'interno del raggruppamento di movimenti (zero filled a partire da '00000'). Se l'articolo movimentato è gestito a taglie deve essere inserito un record anche nella tabella **MOVITAG**
* se la causale utilizzata lo prevede aggiornare il prezzo ultimo acquisto e la data ultimo acquisto nella tabella **ANAMAGA** (anagrafica di magazzino)
* se la causale utilizzata prevede la movimentazione di ordinato cliente/fornitore, attesa di collaudo o attesa di spedizione aggiornare la tabella **GIAC** (saldi per articolo/magazzino indipendenti dall'esercizio)
* se la causale utilizzata prevede la movimentazione di campi di carico/scarico/impegnato aggiornare la tabella **GIACESE** (saldi per articolo/magazzino dipendenti dall'esercizio)
* se l'articolo di magazzino è gestito a taglie aggiornare la tabella **GIACTAG** (saldi per articolo/magazzino/taglia)

Le tabelle **GIAC,GIACESE,GIACTAG** devono essere aggiornate nel seguente modo:

GIAC

La tabella contiene un record per ogni coppia magazzino/articolo movimentata. E' inoltre presente un record per articolo (codice magazzino impostato a NULL) di riepilogo dei dati articolo nei vari magazzini. Per aggiornare la tabella è necessario operare come segue:

* verificare se esiste il record relativo alla coppia magazzino/articolo movimentata. Se il record non esiste crearlo.
* verificare se esiste il record di riepilogo dell'articolo (magazzino=NULL). Se il record non esiste crearlo.
* aggiornare i dati sui due records in funzione dei flags riportati sulla causale di magazzino usata

N.B.: se la gestione dei magazzini multipli non è attivata la tabella contiene solo i records di riepilogo (magazzino=NULL)

GIACESE

La tabella contiene un record per ogni terna esercizio/magazzino/articolo movimentata. E' inoltre presente un record per esercizio/articolo (codice magazzino impostato a NULL) di riepilogo dei dati articolo nei vari magazzini. Per aggiornare la tabella è necessario operare come segue:

* verificare se esiste il record relativo alla terna esercizio/magazzino/articolo movimentata. Se il record non esiste crearlo.
* verificare se esiste il record di riepilogo dell'articolo (magazzino=NULL). Se il record non esiste crearlo.
* aggiornare i dati sui due records in funzione dei flags riportati sulla causale di magazzino usata

N.B.: se la gestione dei magazzini multipli non è attivata la tabella contiene solo i records di riepilogo per ogni esercizio (magazzino=NULL)

GIACTAG

La tabella contiene un record per ogni terna esercizio/magazzino/articolo movimentata. E' inoltre presente un record per esercizio/articolo (codice magazzino impostato a NULL) di riepilogo dei dati articolo nei vari magazzini. Per aggiornare la tabella è necessario operare come segue:

* verificare se esiste il record relativo alla terna esercizio/magazzino/articolo movimentata. Se il record non esiste crearlo.
* verificare se esiste il record di riepilogo dell'articolo (magazzino=NULL). Se il record non esiste crearlo.
* aggiornare i dati sui due records in funzione dei flags riportati sulla causale di magazzino usata

N.B.: se la gestione dei magazzini multipli non è attivata la tabella contiene solo i records di riepilogo per ogni esercizio (magazzino=NULL)

Si rammenta che la giacenza di un articolo è determinata nel seguente modo:

Giacenza=Inventario+Carico da Acquisto+Altri Carichi-Scarico per Vendita-Altri Scarichi

mentre la consistenza è determinata come:

Consistenza=Giacenza-Impegnato-Ordinato Cliente+Ordinato Fornitore

Registrazione movimenti contabili

La registrazione nel database aziendale di movimenti contabili deve avvenire nel seguente modo:

* recuperare il numero da assegnare al raggruppamento di movimenti da registrare dal record con codice **90002** della tabella DPCONFIG (vedi "Gestione numeratori" nel presente manuale)
* inserire il record relativo al movimento da registrare nella tabella **MOVCO** avendo cura di impostare nel campo RIGA il progressivo della riga all'interno del raggruppamento di movimenti (zero filled a partire da '00000'). Se il movimento contabile generato è relativo ad un sottoconto IVA devono essere aggiunti uno o più records nella tabella **MOVIVA**.
* se il movimento inserito comporta un aggiornamento del saldo dare/avere del sottoconto movimentato aggiornare la tabella **TASALSOT**

La tabella **TASALSOT** deve essere aggiornata nel seguente modo:

TASALSOT

La tabella contiene un record per ogni sottoconto movimentato in ogni esercizio contabile. Per aggiornare la tabella è necessario operare come segue:

* verificare se esiste il record relativo alla coppia esercizio/sottoconto movimentata. Se il record non esiste è necessario crearlo
* aggiornare i progressivi dare/avere sul record

Estrazione movimenti di scadenzario

La tabella MOVCO nel database dati aziendali contiene i movimenti contabili registrati (sia fiscali che extracontabili) e i movimenti di scadenzario. In generale ogni record della tabella può essere **contemporaneamente** un movimento contabile fiscale, un movimento di contabilita' analitica e un movimento di scadenzario in funzione del valore di opportuni flags.

Ci soffermeremo nel presente paragrafo sulle query necessarie ad estrarre dal database di SIGLA i movimenti di scadenzario sia attivo che passivo.

Il campo **SCADENZ_SN** della tabella MOVCO contiene il valore “S” nei record che fanno parte dello scadenzario. Lo statement SQL

SELECT * FROM MOVCO WHERE SCADENZ_SN=’S’

consente di estrarre dalla tabella tutti i record che fanno parte, in generale, dello scadenzario[34].

L’appartenenza del record allo scadenzario attivo o passivo è determinata dal valore del campo **ATT_PASSIV** che contiene il valore “A” per lo scadenzario attivo o il valore “P” per lo scadenzario passivo. Il seguente statement estrae tutti i record che fanno parte dello scadenzario passivo:

SELECT * FROM MOVCO WHERE SCADENZ_SN=’S’ AND ATT_PASSIV=’P’

Il tipo di scadenza è individuato dal valore del campo **TIPOEFFETT** che può contenere i seguenti valori:

- “0” per Rimessa Diretta o Contanti
- “1” per Tratta
- “2” per Ricevuta Bancaria
- “3” per Cessione
- “4” per Pagherò
- “5” per Lettera di Credito
- “6” per Tratta Accettata
- “7” per RiBa
- “8” per Ritardato Pagamento
- “9” per Cambiale
- “A” per Altro Pagamento
- “B” per Bonifico Bancario

Volendo estrarre dallo scadenzario passivo tutti i movimenti relativi,ad esempio, a tratte accettate si deve eseguire il seguente statement:

SELECT * FROM MOVCO WHERE SCADENZ_SN=’S’ AND ATT_PASSIV=’P’ AND TIPOEFFETT=’6’

SIGLA divide i movimenti di scadenzario in due categorie:

movimenti sospesi, cioe’ movimenti che non hanno natura fiscale (non vengono quindi stampati nel giornale contabile). Sono movimenti sospesi tutti i movimenti dello scadenzario passivo e quelli dello scadenzario attivo che non corrispondono ad un pagamento certo

movimenti non sospesi cioe’ movimenti di scadenzario che sono contemporaneamente anche movimenti contabili. Rientrano in questa categoria i record dello scadenzario attivo relativi all’emissione di ricevute bancarie, RiBa, tratte, tratte accettate e cambiali.

Un movimento è sospeso se il campo **SOSPESO_SN** contiene il valore “S”.

Lo statement che segue estrae dallo scadenzario attivo tutti i movimenti sospesi:

SELECT * FROM MOVCO WHERE SCADENZ_SN=’S’ AND ATT_PASSIV=’A’ AND SOSPESO_SN=’S’

In SIGLA possono esistere movimenti di scadenzario in lire e in valuta. I record relativi ad operazioni in valuta riportano comunque anche l’importo in lire dell’operazione. Il campo CODVALUTA contiene il codice della valuta usata o il valore NULL o blank (tre bianchi) se il movimento è in lire. Per tutti i movimenti, sia in valuta che in lire, il campo IMPORTO contiene il valore in lire dell’operazione. Per i soli movimenti in valuta (CODVALUTA diverso da NULL e diverso da blank) il campo IMPVALUTA contiene l’importo in valuta estera.

[34] Vengono marcati come appartenenti allo scadenzario anche i record registrati sui sottoconti di chiusura della registrazione sul cliente/fornitore. SIGLA++ genera **sempre** i movimenti di scadenzario in partita doppia (registrazioni tipo cliente/fornitore a portafoglio) sia che siano sospesi che non sospesi. Il campo ATT_PASSIV viene riempito invece solo per i record relativi ai clienti/fornitori. Ne consegue che lo statement illustrato estrae anche i movimenti di giro relativi alle scadenze gestite che vengono invece filtrati dalla specifica <ATT_PASSIV=[valore]> aggiunta sulla clausola WHERE dello statement.

Il seguente statement SQL estrae i movimenti dello scadenzario passivo in dollari USA di importo superiore a $100 (si assume che il dollaro USA sia stato codificato nella tabella valute con il codice USD):

SELECT * FROM MOVCO WHERE SCADENZ_SN='S' AND ATT_PASSIV='P' AND CODVALUTA='USD' AND IMPVALUTA>100

Se si volessero estrarre le righe di scadenzario attivo gestite in lire per importo inferiore a 50,000 si dovrebbe operare come segue:

SELECT * FROM MOVCO WHERE SCADENZ_SN='S' AND ATT_PASSIV='A' AND (CODVALUTA IS NULL OR CODVALUTA=' ') AND IMPORTO<50000

Nella gestione dello scadenzario assume particolare importanza la data di scadenza prevista che viene memorizzata da SIGLA nel campo **DATASCAD**. Ricordiamo che SIGLA gestisce le date in campi di tipo carattere lunghi 8 bytes in formato ANSI (AAAAMMGG).

Il seguente comando SQL estrae i movimenti dello scadenzario attivo relativi a RiBa in scadenza fra il 1 febbraio 1997 e il 31 marzo 1997:

SELECT * FROM MOVCO WHERE SCADENZ_SN='S' AND ATT_PASSIV='A' AND TIPOEFFETT='7' AND DATASCAD>='19970201' AND DATASCAD<='19970331'

Il codice del soggetto intestatario del movimento (cliente/fornitore) viene memorizzato nel campo **SOTTOCONTO**. Così, se al fornitore DeltaPhi è stato attribuito nel piano dei conti il codice '0000000001" il seguente statement estrare dallo scadenzario passivo tutte le RiBa in scadenza fra il 1 febbraio 1997 e il 31 marzo 1997 per il solo fornitore DeltaPhi:

SELECT * FROM MOVCO WHERE SCADENZ_SN='S' AND ATT_PASSIV='P' AND TIPOEFFETT='7' AND DATASCAD>='19970201' AND DATASCAD<='19970331' AND SOTTOCONTO='0000000001'

Normalmente i movimenti sullo scadenzario attivo sono in avere mentre quelli dello scadenzario passivo sono in dare. I record generati dalla registrazione di note di credito cliente/fornitore hanno ovviamente segno opposto rispetto alla norma. SIGLA **non** gestisce nei campi IMPORTO e IMPVALUTA numeri negativi (i valori sono sempre positivi). Ne consegue che per determinare il segno dell'operazione è necessario verificare se il movimento è in dare o in avere. Il campo **SEGNO** contiene il valore "D" per i movimenti in dare e il valore "A" per i movimenti in avere.

TABELLE COLLEGATE

L'estrazione di movimenti di scadenzario operata con una query può essere particolarmente utile se la query stessa è in grado di operare la decodifica delle principali informazioni tabellate che sono riportate in MOVCO solo a livello di codice.
Può essere quindi opportuno "arricchire" lo statement di estrazione con una o più join sulle tabelle connesse. L'esempio che segue decodifica i dati del sottoconto cliente/fornitore movimentato e della causale contabile usata per la registrazione mettendo in join MOVCO,PIACON (tabella piano dei conti) e CAUSCONT (tabella causali contabili). Vengono estratti tutti i record sospesi dallo scadenzario attivo:
SELECT * FROM MOVCO,PIACON,CAUSCONT WHERE MOVCO.SCADENZ_SN='S' AND MOVCO.ATT_PASSIV='A' AND MOVCO.SOSPESO_SN='S' AND MOVCO.SOTTOCONTO=PIACON.SOTTOCONTO AND MOVCO.CAUSALE=CAUSCONT.CODICE

Estrazione movimenti di insoluto

Gli insoluti sono registrati sulla tabella MOVCO. Più precisamente su MOVCO:

- **INSOLUT_SN='S**', individua un record di insoluto
- **INSPAGA_SN='S'**, indica che l'insoluto è stato saldato
- **RIEMISS_SN='S'**, indica che l'insoluto è stato riemesso
- **AGENTE**, contiene il codice agente inserito al momento della registrazione di un insoluto
- **CAPOZONA**, contiene il codice capozona inserito al momento della registrazione di un insoluto

Si possono pertanto effettuare le seguenti interrogazioni:

1. dettaglio di tutti gli insoluti di un cliente ordinati per cliente, agente

SELECT * FROM MOVCO,CLIFO WHERE MOVCO.INSOLUT_SN='S' AND
CLIFO.CODICE=MOVCO.SOTTOCONTO
ORDER BY MOVCO.SOTTOCONTO,MOVCO.AGENTE

2. dettaglio di tutti gli insoluti di un cliente non saldati ordinati per cliente, agente

SELECT * FROM MOVCO,CLIFO WHERE MOVCO.INSOLUT_SN='S' AND MOVCO.INSPAGA_SN<>'S'
AND CLIFO.CODICE=MOVCO.SOTTOCONTO ORDER BY MOVCO.SOTTOCONTO,MOVCO.AGENTE

3. dettaglio di tutti gli insoluti di un agente non saldati

SELECT * FROM MOVCO
WHERE MOVCO.INSOLUT_SN='S' AND MOVCO.INSPAGA_SN<>'S' ORDER BY MOVCO.AGENTE

4. dettaglio di tutti gli insoluti di un agente

SELECT * FROM MOVCO WHERE MOVCO.INSOLUT_SN='S' ORDER BY MOVCO.AGENTE

Inoltre per un insoluto sono indicativi i seguenti campi:

- **SPPAGAT_SN = 'S**', indica che le spese bancarie devono essere pagate dal cliente
- **INTPAGA_SN = 'S'**, indica che gli interessi sono stati pagati dal cliente
- **IMPRICEVUT**, contiene l'importo totale dei movimenti di acconto o saldo relativi versati dal cliente
- **DATADECINT**, contiene la data di partenza del calcolo interessi
- **DATASCAD**, contiene la data di scadenza dell'effetto insoluto
- **IMPORTO**, contiene il valore dell'effetto insoluto.

I record di tipo acconto, saldo o riemissione legati ad un insoluto sono indicativi i seguenti campi:

- **INSOLUTNUM**, contiene il numero univoco della registrazione contabile che contiene il record di insoluto
- **INSOLUTRIG**, contiene il numero di riga della registrazione contabile che contiene, il record di insoluto

I due precedenti campi sono la chiave univoca per recuperare un insoluto.

Quindi si può recuperare un record di insoluto con la seguente query:
SELECT * FROM MOVCO WHERE NUMERO=INSOLUTNUM AND RIGA=INSOLUTRIG

Inoltre per tutti i record di acconto o saldo legati all'insoluto sono riempiti:

- **INSPAGA_SN='S'**, indica che questo è il movimento di saldo dell'insoluto
- **RIEMISS_SN='S'**, indica che questa è la scadenza/effetto di riemissione dell'insoluto.

Collegamento insoluti/provvigioni

Dalle provvigioni è possibile risalire alla scadenza/effetto in contabilità tramite i campi della tabella PROVAGEN:

- **NUMEROFATT**, contiene il numero fattura
- **ESERCIZIO**, contiene il codice dell'esercizio contabile
- **SCADENZAAG**, contiene la data della scadenza/effetto.
- **TIPOPROTOC**, contiene il codice del tipo documento di magazzino

Da questo codice, cioè TIPOPROTOC è possibile trovare sulla tabella TIPODOCUM il campo CAUCONTABI che contiene il codice della causale contabile con cui si è registrata la fattura.
Da questo codice, cioè CAUCONTABI, è possibile trovare sulla tabella CAUSCONT i campi:

- **REGISTRIVA**, che contiene il codice del registro I.V.A. della fattura
- **TIPODOCIVA**, che contiene il codice del tipo documento I.V.A. della fattura.

Inoltre i record di movco che sono anche scadenze/effetti hanno il campo SCADENZ_SN='S'.

Quindi per recuperare il record di effetto/scadenza della fattura interessata:
SELECT * FROM MOVCO WHERE ESEREGISTR=<ESERCIZIO_FATTURA_DA_PROVVIGIONI> AND
NUMPROTOC=<NUMERO_FATTURA_DA_PROVVIGIONI> AND
REGIVA=<REGISTRO_IVA_DA_PROVVIGIONI> AND
TIPODOCIVA=<TIPO_DOCUMENTO_IVA_DA_PROVVIGIONI> AND
DATASCAD=<DATA_SCADENZA_DA_PROVVIGIONI> AND SCADENZ_SN='S'

Per ulteriori ricerche o agganci con gli altri record che costituiscono la fattura si potrà usare il campo RIFENUMERO, che in una fattura registrata da SIGLA lega i movimenti contabili del documento con le sue scadenze/effetti.

Pertanto per recuperare tutti i movimenti della fattura in MOVCO:
SELECT * FROM MOVCO WHERE
RIFENUMERO=<QUELLO_RICAVATO_DAL_RECORD_SCADENZA>

Mentre per recuperare l'effetto/scadenza della fattura che ci interessa:
SELECT * FROM MOVCO WHERE
RIFENUMERO=<QUELLO_RICAVATO_DAL_RECORD_SCADENZA>
AND SCADENZ_SN='S' AND DATASCAD=<DATA_SCADENZA_DELLA PROVVIGIONE>

La seguente query invece consente di recuperare tutte le scadenze della fattura in MOVCO:

SELECT * FROM MOVCO WHERE
RIFENUMERO=<QUELLO_RICAVATO_DAL_RECORD_SCADENZA> AND SCADENZ_SN='S'

Dall'effetto/scadenza, usati per generare automaticamente l'insoluto, sono copiati in MOVCO (se presenti) alcuni valori.

- DATASCAD
- IMPORTO
- AGENTE
- CAPOZONA

Inoltre l'insoluto è registrato nella stessa partita dell'effetto, ed in particolare il campo NUMRIFSALD contiene il numero riferimento saldaconto e quindi individua i record appartenenti ad una partita.
E' quindi possibile recuperare l'insoluto legato alla scadenza:

SELECT * FROM MOVCO WHERE
NUMRIFSALD=<NUMERO_RIFERIMENTO_PARTITA_DELLA_SCADENZA> AND INSOLUT_SN='S'
AND DATASCAD=<DATA_DI_SCADENZA_EFFETTO>

Gestione Fax

SIGLA utilizza Delrina WinFax Pro per spedire in automatico fax. La procedura DeltaFax mantiene sotto controllo la directory specificata nella procedura di configurazione in attesa che compaiano delle richieste di invio fax. Quando una richiesta viene rilevata (tramite la creazione da parte di SIGLA degli appositi files di seguito descritti), DeltaFax agisce su WinFax Pro via DDE sottomettendo la richiesta di inoltro.

Una richiesta fax viene effettuata dal pacchetto creando nella directory monitorata da DeltaFax due files aventi lo stesso nome e postfisso diverso.

Il primo file, di tipo TXT, contiene il testo del messaggio da inviare.

Il secondo file, di tipo INF, contiene le specifiche di invio. Il file *.INF è a lunghezza fissa. Le informazioni al suo interno sono organizzate come nella seguente struttura C:

```
typedef struct {
               char coddes[6];       // non usato
               char ragsoc[25];      // ragione sociale destinatario
               char numtel[20];      // numero telefono con prefisso
               char data[8];         // data prevista d'invio
               char ora[4];          // ora prevista d'invio
               char dachi[40];       // descrizione del mittente
               char achi[40];        // descrizione del destinatario
               char ogget1[40];      // primo rigo oggetto messaggio
               char ogget2[40];      // secondo rigo oggetto messaggio
               char ogget3[40];      // terzo rigo oggetto messaggio
               char ogget4[40];      // quarto rigo oggetto messaggio
               char stampa[1];       //'S' se seve essere stampata una copia del fax su carta
               char invio[1];        // 'S' se il fax deve essere inviato
               char grigli[12];      // nome del file *.BMP da usare come sfondo di ciascuna
                                     // pagina
               char copert[12];      // nome del file *.BMP da usare come copertina
               char rigpag[2];       // numero di righe testo da usare per formattare una pagina
                                     // ( se 0 si assume il default di 66)
               char respons[3];      // non usato
               char firma[12];       // nome del file *.BMP contenente la firma da accodare al fax
               char logo[12];        // nome del file *.BMP contenetente il logo da anteporre a
                                     // ciascuna pagina
               char font[1];         // 0 per 80 caratteri per righa, 1 per 132 caratteri per riga
               char docum[12];       // non usato
               char lista[1];        // non usato (contiene il valore fisso 'N')
               char end[1];          // identificatore di file file (0x1A)
} INFFILE;
```

APPENDICI

Header Files

Di seguito sono riportati i sorgenti di alcuni header files distribuiti per consentire la ricompilazione della DLL SIGPPDLL.DLL.

DPDBASE.H - Versione client/server 16 bit

```
#ifndef DPDBASE
        #define DPDBASE
        #define DPODBC
        #ifdef DPDLL
                #define DPCALL _export
        #else
                #define DPCALL
        #endif
        #define MS_ODBC_DRIVER
        #ifdef MS_ODBC_DRIVER
                #define MAX_HSTMT       300             // dimensione tabella handles per statement
        #endif

        #define DP_INVALID_ENV_ALLOCATION               1
        #define DP_INVALID_ALLOC_CONNECT                2
        #define DP_INVALID_CONNECT                      3
        #define DP_INVALID_INFO                         4
        #define DP_INVALID_STAT_ALLOC                   5
        #define DP_INVALID_DB_TYPE                      6
        #define DP_EXECUTESQL_ERROR                     7
        #define DP_SCROLL_ERROR                         8
        #define DP_RESULT_COLS_ERROR                    9
        #define DP_NOCOLUMNS                           10
        #define DP_DESCRIBE_ERROR                      11
        #define DP_COLUMN_NAME_NOT_FOUND               12
        #define DP_BIND_ERROR                          13
        #define DP_BAD_CURSOR_NAME                     14
        #define DP_TOO_MANY_TABLES                     15
        #define DP_TOO_MANY_FIELDS                     16
        #define DP_INVALID_TABLE_POINTER               17
        #define DP_NO_TABLE_SPECIFIED                  18
        #define DP_NO_UNIQUE_KEY_SPECIFIED             19
        #define DP_MAX_UNIQUE_IDENTIFIER              256

        #define DP_GREATER_OR_EQUAL                   0
        #define DP_LESS_OR_EQUAL                      1
        #define DP_BEGINS_WITH                        2
        #define DP_CONTAINS                           3
        #define DP_MAX_TABLE                          5
        #define DP_MAX_FIELDS                       256

        /*-------------------------------------------------------------------------------------------
                definizione delle stringhe di uso generale
        -------------------------------------------------------------------------------------------*/
        extern char *DP_CODICE;
        extern char *DP_DESCRIZIONE;
        extern char *DP_DESCR_PICT;
        extern char *DP_SELECT;
        extern char *DP_STATO;
        extern char *DP_UTENTE;
        extern char *DP_ULT_AGG;

        #include <WINDOWS.H>
        #include <SQL.H>
        #include <SQLEXT.H>
        #include <STDIO.H>
        #include <STRING.H>
        #include <STDLIB.H>
        #include <TIME.H>

        /*-------------------------------------------------------------------------------------------
                Definizione della classe DPODBCEnvironment
        -------------------------------------------------------------------------------------------*/
        class DPODBCEnvironment {
        public:
                HENV henv;
                int errcode;
                DPODBCEnvironment(void);
                ~DPODBCEnvironment(void);
        };

        /*-------------------------------------------------------------------------------------------
                Definizione della classe DPDbase
        -------------------------------------------------------------------------------------------*/
        class DPDbase {
        public:
                DPODBCEnvironment *theEnv;
                BOOL IsValidated;
                int errcode;
                HDBC hdbc;
                HDBC hdbc1,hdbc2,hdbc3,hdbc4,hdbc5;
                BOOL IsConnected;
```

```
                BOOL IsCommittable;
                BOOL IsTransactionPending;
                short int sql_active_statements;
                int ObjectInUse;
                char DPCHAR[128];
                char DPNUM[128];
                BOOL NCreateParam;
                BOOL TransactionOn;
                BOOL LockingOn;
                BOOL LockingCursor;
                SWORD NumericType;
                BOOL NeedOrderBy;
                BOOL CanPrepareStatement;
                time_t LastUpdate;
                BOOL IsEBCDIC;
                char DbmsType[128];
                char DPFIRST[2];
                char DPLAST[2];
                char DPLastBuf[128],DPFirstBuf[128];
                // tabella degli handles
                #ifdef MS_ODBC_DRIVER
                        int DPObjInUse[MAX_HSTMT];
                        HSTMT odbchandle[MAX_HSTMT];
                #endif
                BOOL OldNeedOrderBy;
              int IsSubString;

                DPDbase(DPODBCEnvironment *DPEnv);
                ~DPDbase();
                void GetTypeInfo(void);
                BOOL ExecuteSQL(char *SQLStm);
                BOOL Connect(char *SourceName);
                BOOL IsValid(void);
              int RetError(void);
              BOOL BeginTrans(void);
              BOOL CommitTrans(void);
              BOOL RollBack(void);
              void SetTransaction(BOOL status);
              void SetLocking(BOOL status);
              void SetLockingCursor(BOOL status);
              void SetOrderBy(BOOL status);
              void SetPrepare(BOOL status);
              char *DPFirst(int len=1);
              char *DPLast(int len=1);
              void SetEBCDIC(BOOL status=FALSE);
      };

      /*-------------------------------------------------------------------------------------------
               Definizione della classe DPObject
      -------------------------------------------------------------------------------------------*/
      class DPObject {
      public:
                      int ObjectID;
                      BOOL IsValidated;
                      char Esercizio[5];
                      DPDbase *theDB;
                      HSTMT hstmt;
                      int errcode;
                      BOOL bBOF;
                      BOOL bEOF;
                      BOOL Locked;
                      SWORD resultCol;
                      int TableNo,FieldNo;
                      char *Table[DP_MAX_TABLE];
                      BOOL IsToUpdate[DP_MAX_TABLE];
                      short int TablePoint[DP_MAX_FIELDS];
                      char *Fields[DP_MAX_FIELDS];
                      void *Variables[DP_MAX_FIELDS];
                      short int VarSize[DP_MAX_FIELDS];
                      short int FieldsType[DP_MAX_FIELDS];
                      BOOL FldIsToUpdate[DP_MAX_FIELDS];
                      char theUniqueIdentifier[DP_MAX_UNIQUE_IDENTIFIER];

                      DPObject(DPDbase *theDbase,int theObjectID=0,char far *theEsercizio=NULL);
                      ~DPObject(void);
                      BOOL IsValid(void);
                      int RetError(void);
                      BOOL SetTableName(char *TableName,BOOL MustBeUpdate=TRUE);
                      BOOL SetField(short int TablePointer,char *FieldName,short int FieldType,void
*variable,short int varsize);
                      virtual void Bind(void);
                      BOOL ExecuteSQL(void);
                      BOOL ExecuteSQL(char *SQLStm);
                      virtual BOOL MoveNext(void);
                      BOOL IsEOF(void);
                      BOOL IsBOF(void);
                      void Lock(void);
                      void Unlock(void);
                      BOOL IsLocked(void);
                      SWORD RetResultColNo(void);
```

```
                        BOOL BindCol(int ColumnNumber,char *buffer,long int buf_len,short int type=SQL_C_CHAR);
                        BOOL RetCursorName(unsigned char far *CursorName);
                        BOOL Delete(BOOL UseDBStmt=FALSE);
                        BOOL ExcludeFromUpdate(char *TableName,char *FieldName);
                        BOOL Update(BOOL UseDBStmt=FALSE);
                        BOOL Append(BOOL UseDBStmt=FALSE);
                        void SetUniqueIdentifier(char *theIdentifier);
                        void SetUniqueIdentifier(void);
                        BOOL SeekObj(char *theUniqueIdentifier,BOOL SetUniqueKey=FALSE);
                        virtual BOOL CreateTable(int theTableNo=0);
                        virtual BOOL StoreData(FILE *theFile);
                        virtual BOOL LoadData(FILE *theFile);
                        virtual BOOL DoPrint(char *buffer,int thePrintType=0);
                        virtual BOOL CreateIndex(void);
                        /*-------------------------------------------------------------------------
                                funzioni virtuali pure
                        -------------------------------------------------------------------------*/
                        virtual int RetFindCriteriaNo()=0;
                        virtual char far *RetFindTitle()=0;
                        virtual char far *RetFindCriteriaDescr(int CriteriaNo=0)=0;
                        virtual char far *RetFindCriteriaPict(int CriteriaNo=0)=0;
                        virtual BOOL SeekFindCriteria(char *theKey,int CriteriaNo)=0;
                        virtual BOOL RetFindRow(char *buffer,int CriteriaNo=0)=0;
                        virtual BOOL Seek(char *theUniqueKey)=0;
                        virtual BOOL SeekRange(char *theInitialKey,char *theFinalKey,int CriteriaNo=0)=0;
                        virtual void Clear()=0;
        };
#endif
```

DPDBASE.H - Versione client/server 32 bit

```
#ifndef DPDBASE
        #define DPDBASE
        /*-----------------------------------------------------------------------------------------
                Compilation switch: definire le macro necessario
                                        commentare le altre
        -----------------------------------------------------------------------------------------*/
        #define MS_ODBC_DRIVER
        /*-----------------------------------------------------------------------------------------
                Gli switch di compilazione devono essere defini dall'esterno
        -----------------------------------------------------------------------------------------*/
        // #define DPODBC
        // #define DPQELIB
        // #define DPDLL
        #ifdef DPODBC
                #ifdef DPQELIB
                        #error - Sono stati definiti entrambi gli switch DPODBC e DPQELIB
                #endif
                #ifdef DPDB2CLI
                        #error - Sono stati definiti entrambi gli switch DPODBC e DPDB2CLI
                #endif
        #endif
        #ifdef DPQELIB
                #ifdef DPDB2CLI
                        #error - Sono stati definiti entrambi gli switch DPQELIB e DPDB2CLI
                #endif
        #endif
        #ifndef DPODBC
                #ifndef DPQELIB
                        #ifndef DPDB2CLI
                                #error - Almeno uno degli switch DPODBC, DPQELIB o DPDB2CLI deve essere definito
                        #endif
                #endif
        #endif
        #ifdef DPQELIB
                #define NOT_NATIVE_ODBC
        #endif
        #ifdef DPDLL
                #define DPCALL _export
        #else
                #define DPCALL
        #endif

        #ifdef DPQELIB
                #define SQL_C_LONG (short int) 0
                #define SQL_C_CHAR (short int) 1
                #define SQL_C_DOUBLE (short int) 2
                #define BOOL unsigned char
                #define FALSE 0
                #define TRUE  1
                #define far
                #define HSTMT qeHANDLE
                #define NOT_NATIVE_ODBC
                #define LPSTR char*
        #endif
        #ifdef MS_ODBC_DRIVER
                #define MAX_HSTMT       300             // dimensione tabella handles per statement
        #endif
        #define DP_INVALID_ENV_ALLOCATION               1
        #define DP_INVALID_ALLOC_CONNECT                2
        #define DP_INVALID_CONNECT                      3
```

```
#define DP_INVALID_INFO                         4
#define DP_INVALID_STAT_ALLOC                   5
#define DP_INVALID_DB_TYPE                      6
#define DP_EXECUTESQL_ERROR                     7
#define DP_SCROLL_ERROR                         8
#define DP_RESULT_COLS_ERROR                    9
#define DP_NOCOLUMNS                           10
#define DP_DESCRIBE_ERROR                      11
#define DP_COLUMN_NAME_NOT_FOUND               12
#define DP_BIND_ERROR                          13
#define DP_BAD_CURSOR_NAME                     14
#define DP_TOO_MANY_TABLES                     15
#define DP_TOO_MANY_FIELDS                     16
#define DP_INVALID_TABLE_POINTER               17
#define DP_NO_TABLE_SPECIFIED                  18
#define DP_NO_UNIQUE_KEY_SPECIFIED             19
#define DP_MAX_UNIQUE_IDENTIFIER              256
#define DP_GREATER_OR_EQUAL                                   0
#define DP_LESS_OR_EQUAL                                  1
#define DP_BEGINS_WITH                                        2
#define DP_CONTAINS                                           3
#define DP_MAX_TABLE                                      5
#define DP_MAX_FIELDS                                     256
/*-------------------------------------------------------------------------------------------
        definizione delle stringhe di uso generale
-------------------------------------------------------------------------------------------*/
extern char *DP_CODICE;
extern char *DP_DESCRIZIONE;
extern char *DP_DESCR_PICT;
extern char *DP_SELECT;
extern char *DP_STATO;
extern char *DP_UTENTE;
extern char *DP_ULT_AGG;
#ifdef DPQELIB
        #include <QELIB.H>
#endif
#ifdef DPODBC
        #ifndef WIN32
                #include <WINDOWS.H>
        #else
                #define BOOL unsigned char
                #define HWND unsigned long int
                #define FALSE 0
                #define TRUE  1
                #define far
                #define LPSTR char*
                #define WORD SQLUSMALLINT
                #define DWORD SQLUINTEGER
                #define SDWORD SQLINTEGER
        #endif
        #include <SQL.H>
        #include <SQLEXT.H>
        #include <TIME.H>
#endif

#ifdef DPDB2CLI
        #include <SQLCLI1.H>
        #define DPODBC
#endif

#ifdef DB2OS2
        #define far
        #define LPSTR char*
        #define HENV SQLHENV
        #define HDBC SQLHDBC
        #define HSTMT SQLHSTMT
        #define WORD SQLUSMALLINT
        #define DWORD SQLUINTEGER
        #define SDWORD SQLINTEGER
#endif
#include <STDIO.H>
#include <STRING.H>
#include <STDLIB.H>
/*-------------------------------------------------------------------------------------------
        Definizione della classe DPODBCEnvironment
-------------------------------------------------------------------------------------------*/
class DPODBCEnvironment {
public:
        #ifdef DPODBC
                HENV henv;
        #endif
        int errcode;
        DPODBCEnvironment(void);
        ~DPODBCEnvironment(void);
};
/*-------------------------------------------------------------------------------------------
        Definizione della classe DPDbase
-------------------------------------------------------------------------------------------*/
class DPDbase {
public:
```

```
        DPODBCEnvironment *theEnv;
        BOOL IsValidated;
        int errcode;
        #ifdef DPODBC
                HDBC hdbc;
                HDBC hdbc1,hdbc2,hdbc3,hdbc4,hdbc5;
        #endif
        #ifdef DPQELIB
                qeHANDLE hdbc;
        #endif
        BOOL IsConnected;
        BOOL IsCommittable;
        BOOL IsTransactionPending;
        short int sql_active_statements;
        int ObjectInUse;
        char DPCHAR[128];
        char DPNUM[128];
        BOOL NCreateParam;
        BOOL TransactionOn;
        BOOL LockingOn;
        BOOL LockingCursor;
        #if defined(DPODBC)
             SWORD NumericType;
        #endif
        #if defined(DPODBC) && !defined(DPDB2CLI)
                time_t LastUpdate;
        #endif
        BOOL NeedOrderBy;
        BOOL CanPrepareStatement;
        BOOL IsEBCDIC;
        char DbmsType[128];
        char DPFIRST[2];
        char DPLAST[2];
        char DPLastBuf[128],DPFirstBuf[128];
        #ifdef MS_ODBC_DRIVER
                int DPObjInUse[MAX_HSTMT];
                #ifdef DPODBC
                        HSTMT odbchandle[MAX_HSTMT];
                #endif
                #ifdef DPQELIB
                        qeHANDLE odbchandle[MAX_HSTMT];
                #endif
        #endif
        BOOL OldNeedOrderBy;
        int IsSubString;

        DPDbase(DPODBCEnvironment *DPEnv);
        ~DPDbase();
        void GetTypeInfo(void);
        BOOL ExecuteSQL(char *SQLStm);
        BOOL Connect(char *SourceName);
        BOOL IsValid(void);
        int RetError(void);
        BOOL BeginTrans(void);
        BOOL CommitTrans(void);
        BOOL RollBack(void);
        void SetTransaction(BOOL status);
        void SetLocking(BOOL status);
        void SetLockingCursor(BOOL status);
        void SetOrderBy(BOOL status);
        void SetPrepare(BOOL status);
        char *DPFirst(int len=1);
        char *DPLast(int len=1);
        void SetEBCDIC(BOOL status=FALSE);
};

/*-------------------------------------------------------------------------------------------
        Definizione della classe DPObject
-------------------------------------------------------------------------------------------*/
class DPObject {
public:
                int ObjectID;
                BOOL IsValidated;
                char Esercizio[5];
                DPDbase *theDB;
                #ifdef DPODBC
                        HSTMT hstmt;
                #endif
                #ifdef DPQELIB
                        qeHANDLE hstmt;
                #endif
                int errcode;
                BOOL bBOF;
                BOOL bEOF;
                BOOL Locked;
                #ifdef DPODBC
                        SWORD resultCol;
                #endif
                #ifdef DPQELIB
                        int resultCol;
```

```
                        #endif
                        int TableNo,FieldNo;
                        char *Table[DP_MAX_TABLE];
                        BOOL IsToUpdate[DP_MAX_TABLE];
                        short int TablePoint[DP_MAX_FIELDS];
                        char *Fields[DP_MAX_FIELDS];
                        void *Variables[DP_MAX_FIELDS];
                        short int VarSize[DP_MAX_FIELDS];
                        short int FieldsType[DP_MAX_FIELDS];
                        #ifdef DPQELIB
                                long int FieldsLen[DP_MAX_FIELDS];
                        #endif
                        #ifdef DPODBC
                                SDWORD pcbValue[DP_MAX_FIELDS];
                        #endif
                        BOOL FldIsToUpdate[DP_MAX_FIELDS];
                        char theUniqueIdentifier[DP_MAX_UNIQUE_IDENTIFIER];

                        DPObject(DPDbase *theDbase,int theObjectID=0,char far *theEsercizio=NULL);
                        ~DPObject(void);
                        BOOL IsValid(void);
                        int RetError(void);
                        BOOL SetTableName(char *TableName,BOOL MustBeUpdate=TRUE);
                        BOOL SetField(short int TablePointer,char *FieldName,short int FieldType,void
*variable,short int varsize);
                        virtual void Bind(void);
                        BOOL ExecuteSQL(void);
                        BOOL ExecuteSQL(char *SQLStm);
                        virtual BOOL MoveNext(void);
                        BOOL IsEOF(void);
                        BOOL IsBOF(void);
                        void Lock(void);
                        void Unlock(void);
                        BOOL IsLocked(void);
                        #ifdef DPODBC
                                SWORD RetResultColNo(void);
                        #endif
                        #ifdef DPQELIB
                                int RetResultColNo(void);
                        #endif
                        BOOL BindCol(int ColumnNumber,char *buffer,long int buf_len,short int type=SQL_C_CHAR);
                        BOOL RetCursorName(unsigned char far *CursorName);
                        BOOL Delete(BOOL UseDBStmt=FALSE);
                        BOOL ExcludeFromUpdate(char *TableName,char *FieldName);
                        BOOL Update(BOOL UseDBStmt=FALSE);
                        BOOL Append(BOOL UseDBStmt=FALSE);
                        void SetUniqueIdentifier(char *theIdentifier);
                        void SetUniqueIdentifier(void);
                        BOOL SeekObj(char *theUniqueIdentifier,BOOL SetUniqueKey=FALSE);
                        virtual BOOL CreateTable(int theTableNo=0);
                        virtual BOOL StoreData(FILE *theFile);
                        virtual BOOL LoadData(FILE *theFile);
                        virtual BOOL DoPrint(char *buffer,int thePrintType=0);
                        virtual BOOL CreateIndex(void);
                        /*-------------------------------------------------------------------------
                                funzioni virtuali pure
                        -------------------------------------------------------------------------*/
                        virtual int RetFindCriteriaNo()=0;
                        virtual char far *RetFindTitle()=0;
                        virtual char far *RetFindCriteriaDescr(int CriteriaNo=0)=0;
                        virtual char far *RetFindCriteriaPict(int CriteriaNo=0)=0;
                        virtual BOOL SeekFindCriteria(char *theKey,int CriteriaNo)=0;
                        virtual BOOL RetFindRow(char *buffer,int CriteriaNo=0)=0;
                        virtual BOOL Seek(char *theUniqueKey)=0;
                        virtual BOOL SeekRange(char *theInitialKey,char *theFinalKey,int CriteriaNo=0)=0;
                        virtual void Clear()=0;
        };
#endif
```

DPDBASE.H - Versione Start Suite 32 bit

```
#ifndef DPDBASE
        #define DPDBASE
        /*-----------------------------------------------------------------------------------------
                Compilation switch: definire le macro necessario
                                        commentare le altre
        -----------------------------------------------------------------------------------------*/
        #define MS_ODBC_DRIVER
        /*-----------------------------------------------------------------------------------------
                Gli switch di compilazione devono essere definiti dall'esterno
        -----------------------------------------------------------------------------------------*/
        #ifdef DPODBC
                #ifdef DPQELIB
                        #error - Sono stati definiti entrambi gli switch DPODBC e DPQELIB
                #endif
                #ifdef DPDB2CLI
                        #error - Sono stati definiti entrambi gli switch DPODBC e DPDB2CLI
                #endif
                #ifdef DPCODEBASE
                        #error - Sono stati definiti entrambi gli switch DPODBC e DPCODEBASE
```

```
                #endif
        #endif
        #ifdef DPQELIB
                #ifdef DPDB2CLI
                        #error - Sono stati definiti entrambi gli switch DPQELIB e DPDB2CLI
                #endif
                #ifdef DPCODEBASE
                        #error - Sono stati definiti entrambi gli switch DPODBC e DPCODEBASE
                #endif
        #endif
        #ifndef DPODBC
                #ifndef DPQELIB
                        #ifndef DPDB2CLI
                                #ifndef DPCODEBASE
                                        #error - Almeno uno degli switch DPODBC, DPQELIB, DPDB2CLI o DPCODEBASE
deve essere definito
                                #endif
                        #endif
                #endif
        #endif
        #ifdef DPQELIB
                #define NOT_NATIVE_ODBC
        #endif
        #ifdef DPDLL
                #define DPCALL _export
        #else
                #define DPCALL
        #endif
        #ifdef DPODBC
                #pragma message("  ***  Utilizzo di DPDBASE versione ODBC")
        #endif
        #ifdef DPQELIB
                #pragma message("  ***  Utilizzo di DPDBASE versione Q+E")
        #endif
        #ifdef DPDB2CLI
                #pragma message("  ***  Utilizzo di DPDBASE versione DB/2 CLI")
        #endif
        #ifdef DPCODEBASE
                #pragma message("  ***  Utilizzo di DPDBASE versione CODEBASE")
        #endif
        #ifdef DPQELIB
                #define SQL_C_LONG (short int) 0
                #define SQL_C_CHAR (short int) 1
                #define SQL_C_DOUBLE (short int) 2
                #define BOOL unsigned char
                #define FALSE 0
                #define TRUE  1
                #define far
                #define HSTMT qeHANDLE
                #define NOT_NATIVE_ODBC
                #define LPSTR char*
        #endif
        #ifdef DPCODEBASE
                #define SQL_C_LONG (short int) 0
                #define SQL_C_CHAR (short int) 1
                #define SQL_C_DOUBLE (short int) 2
                /*-------------------------------------------------------------------------------
                        numero massimo di database aperti contemporaneamente
                -------------------------------------------------------------------------------*/
                #define CB4FILES 150
        #endif
        #ifdef MS_ODBC_DRIVER
                #define MAX_HSTMT       300             // dimensione tabella handles per statement
        #endif

        #define DP_INVALID_ENV_ALLOCATION               1
        #define DP_INVALID_ALLOC_CONNECT                2
        #define DP_INVALID_CONNECT                      3
        #define DP_INVALID_INFO                         4
        #define DP_INVALID_STAT_ALLOC                   5
        #define DP_INVALID_DB_TYPE                      6
        #define DP_EXECUTESQL_ERROR                     7
        #define DP_SCROLL_ERROR                         8
        #define DP_RESULT_COLS_ERROR                    9
        #define DP_NOCOLUMNS                           10
        #define DP_DESCRIBE_ERROR                      11
        #define DP_COLUMN_NAME_NOT_FOUND               12
        #define DP_BIND_ERROR                          13
        #define DP_BAD_CURSOR_NAME                     14
        #define DP_TOO_MANY_TABLES                     15
        #define DP_TOO_MANY_FIELDS                     16
        #define DP_INVALID_TABLE_POINTER               17
        #define DP_NO_TABLE_SPECIFIED                  18
        #define DP_NO_UNIQUE_KEY_SPECIFIED             19
        #define DP_MAX_UNIQUE_IDENTIFIER               256
        #define DP_MAX_TABLE    5
        #define DP_MAX_FIELDS   256
        #ifndef DPCODEBASE
                #define DP_GREATER_OR_EQUAL                                     0
                #define DP_LESS_OR_EQUAL                                1
```

```
                #define DP_BEGINS_WITH                                          2
                #define DP_CONTAINS                                                     3
        #endif

        /*-----------------------------------------------------------------------------------------
                definizione delle stringhe di uso generale
        -----------------------------------------------------------------------------------------*/
        extern char *DP_CODICE;
        extern char *DP_DESCRIZIONE;
        extern char *DP_DESCR_PICT;
        extern char *DP_SELECT;
        extern char *DP_STATO;
        extern char *DP_UTENTE;
        extern char *DP_ULT_AGG;
    #ifdef DPQELIB
                #include <QELIB.H>
    #endif
    #ifdef DPODBC
                #ifndef WIN32
                        #include <WINDOWS.H>
                #else
                        #define BOOL unsigned char
                        #define HWND unsigned long int
                        #define FALSE 0
                        #define TRUE  1
                        #define far
                        #define LPSTR char*
                        #define WORD SQLUSMALLINT
                        #define DWORD SQLUINTEGER
                        #define SDWORD SQLINTEGER
                #endif
                #include <SQL.H>
                #include <SQLEXT.H>
                #include <TIME.H>
        #endif
        #ifdef DPDB2CLI
                #include <SQLCLI1.H>
                #define DPODBC
        #endif
        #ifdef DB2OS2
                #define far
                #define LPSTR char*
                #define HENV SQLHENV
                #define HDBC SQLHDBC
                #define HSTMT SQLHSTMT
                #define WORD SQLUSMALLINT
                #define DWORD SQLUINTEGER
                #define SDWORD SQLINTEGER
        #endif
        #ifdef DPCODEBASE
                #ifndef WIN32
                        #include <WINDOWS.H>
                #endif
                extern "C" {
                        #include "d4all.h"
                        #include "x4filter.h"
                }
        #endif
        #include <STDIO.H>
        #include <STRING.H>
        #include <STDLIB.H>
        /*-----------------------------------------------------------------------------------------
                Definizione della classe DPODBCEnvironment
        -----------------------------------------------------------------------------------------*/
        class DPODBCEnvironment {
        public:
                #ifdef DPODBC
                        HENV henv;
                #endif
                int errcode;
                DPODBCEnvironment(void);
                ~DPODBCEnvironment(void);
        };

        /*-----------------------------------------------------------------------------------------
                Definizione della classe DPDbase
        -----------------------------------------------------------------------------------------*/
        class DPDbase {
        public:
                DPODBCEnvironment *theEnv;
                BOOL IsValidated;
                int errcode;
                #ifdef DPODBC
                        HDBC hdbc;
                        HDBC hdbc1,hdbc2,hdbc3,hdbc4,hdbc5;
                #endif
                #ifdef DPQELIB
                        qeHANDLE hdbc;
                #endif
                #ifdef DPCODEBASE
```

```
                char databasepath[128];
                DATA4 *dbdata[CB4FILES];
                X4FILTER xdb[CB4FILES];
                char TableNames[CB4FILES][15];
                int ObjectInUse[CB4FILES];
        #endif
        BOOL IsConnected;
        BOOL IsCommittable;
        BOOL IsTransactionPending;
        short int sql_active_statements;
        char DPFIRST[2];
        char DPLAST[2];
        char DPLastBuf[128],DPFirstBuf[128];
        #ifndef DPCODEBASE
                int ObjectInUse;
                char DPCHAR[128];
                char DPNUM[128];
                BOOL NCreateParam;
                BOOL TransactionOn;
                BOOL LockingOn;
                BOOL LockingCursor;
            #if defined(DPODBC)
                            SWORD NumericType;
            #endif
            #if defined(DPODBC) && !defined(DPDB2CLI)
                        time_t LastUpdate;
            #endif
            BOOL NeedOrderBy;
            BOOL CanPrepareStatement;
            BOOL IsEBCDIC;
            char DbmsType[128];
           #ifdef MS_ODBC_DRIVER
                        int DPObjInUse[MAX_HSTMT];
                        #ifdef DPODBC
                                HSTMT odbchandle[MAX_HSTMT];
                        #endif
                        #ifdef DPQELIB
                                qeHANDLE odbchandle[MAX_HSTMT];
                        #endif
                #endif
        #endif
        #ifndef DPCODEBASE
                BOOL OldNeedOrderBy;
                int IsSubString;
        #endif
        DPDbase(DPODBCEnvironment *DPEnv);
        ~DPDbase();
        BOOL Connect(char *SourceName);
        BOOL IsValid(void);
        int RetError(void);
        BOOL BeginTrans(void);
        BOOL CommitTrans(void);
        BOOL RollBack(void);
        char *DPFirst(int len=1);
        char *DPLast(int len=1);
        #ifdef DPCODEBASE
                int DB_open(char *filename,BOOL IsExclusive=FALSE);
                BOOL DB_close(int handle);
                BOOL DB_top(int handle);
                BOOL DB_bottom(int handle);
                BOOL DB_skip(int handle,long int recordno=1L);
                BOOL DB_lock(int handle);
                BOOL DB_unlock(int handle);
                long int DB_recno(int handle);
                BOOL DB_tagselect(int handle,char *ndxname);
                int DB_seek(int handle,char *key);
                int DB_seek(int handle,double key);
                BOOL DB_eof(int handle);
                BOOL DB_bof(int handle);
                BOOL DB_go(int handle,long int recno);
                BOOL DB_delete(int handle);
                BOOL DB_rewrite(int handle);
                void DB_appendstart(int handle);
                BOOL DB_append(int handle);
                BOOL DB_value(int handle,char *fldname,char *value);
                BOOL DB_replace(int handle,char *fldname,char *value);
                BOOL DropTable(char *TableName);
                BOOL ExecuteSQL(char *SQLStm);
        #else
                void GetTypeInfo(void);
                BOOL ExecuteSQL(char *SQLStm);
                void SetTransaction(BOOL status);
                void SetLocking(BOOL status);
                void SetLockingCursor(BOOL status);
                void SetOrderBy(BOOL status);
                void SetPrepare(BOOL status);
                void SetEBCDIC(BOOL status=FALSE);
        #endif
};
```

```
        /*-------------------------------------------------------------------------------------------
                Definizione della classe DPObject
        -------------------------------------------------------------------------------------------*/
        class DPObject {
        public:
                int ObjectID;
                BOOL IsValidated;
                char Esercizio[5];
                DPDbase *theDB;
                #ifdef DPODBC
                        HSTMT hstmt;
                #endif
                #ifdef DPQELIB
                        qeHANDLE hstmt;
                #endif
                int errcode;
                BOOL bBOF;
                BOOL bEOF;
                BOOL Locked;
                #ifdef DPODBC
                        SWORD resultCol;
                #endif
                #ifdef DPQELIB
                        int resultCol;
                #endif
                int TableNo,FieldNo;
                char *Table[DP_MAX_TABLE];
                BOOL IsToUpdate[DP_MAX_TABLE];
                short int TablePoint[DP_MAX_FIELDS];
                char *Fields[DP_MAX_FIELDS];
                void *Variables[DP_MAX_FIELDS];
                short int VarSize[DP_MAX_FIELDS];
                short int FieldsType[DP_MAX_FIELDS];
                #ifdef DPQELIB
                        long int FieldsLen[DP_MAX_FIELDS];
                #endif
                #ifdef DPODBC
                        SDWORD pcbValue[DP_MAX_FIELDS];
                #endif
                BOOL FldIsToUpdate[DP_MAX_FIELDS];
                char theUniqueIdentifier[DP_MAX_UNIQUE_IDENTIFIER];
                #ifdef DPCODEBASE
                        int TableHandle[DP_MAX_TABLE];
                        int actualtable;
                        long int actualrec;
                        int selectcriteria;
                        BOOL IsRelateOn;
                        RELATE4 *relate;
                        RELATE4 *lastslave;
                        BOOL IsFileCursor;
                        char itsCursorFile[64];
                        #ifndef WIN32
                                int itsFile;
                                OFSTRUCT oF;
                        #else
                                FILE *CursorFile;
                                #define far
                                #define HWND long int
                        #endif
                        long int RecordInCursor;
                #endif

                DPObject(DPDbase *theDbase,int theObjectID=0,char far *theEsercizio=NULL);
                ~DPObject(void);
                BOOL IsValid(void);
                int RetError(void);
                BOOL SetTableName(char *TableName,BOOL MustBeUpdate=TRUE);
          BOOL SetField(short int TablePointer,char *FieldName,short int FieldType,void *variable,short int
varsize);
                virtual void Bind(void);
                virtual BOOL MoveNext(void);
                BOOL IsEOF(void);
                BOOL IsBOF(void);
                void Lock(void);
                void Unlock(void);
                BOOL IsLocked(void);
                BOOL BindCol(int ColumnNumber,char *buffer,long int buf_len,short int type=SQL_C_CHAR);
                BOOL Delete(BOOL UseDBStmt=FALSE);
                BOOL ExcludeFromUpdate(char *TableName,char *FieldName);
                BOOL Update(BOOL UseDBStmt=FALSE);
                BOOL Append(BOOL UseDBStmt=FALSE);
                void SetUniqueIdentifier(char *theIdentifier);
                void SetUniqueIdentifier(void);
                virtual BOOL CreateTable(int theTableNo=0);
                virtual BOOL StoreData(FILE *theFile);
                virtual BOOL LoadData(FILE *theFile);
                virtual BOOL DoPrint(char *buffer,int thePrintType=0);
                virtual BOOL CreateIndex(void);
```

```
                /*-------------------------------------------------------------------------
                        funzioni virtuali pure
                -------------------------------------------------------------------------*/
                virtual int RetFindCriteriaNo()=0;
                virtual char far *RetFindTitle()=0;
                virtual char far *RetFindCriteriaDescr(int CriteriaNo=0)=0;
                virtual char far *RetFindCriteriaPict(int CriteriaNo=0)=0;
                virtual BOOL SeekFindCriteria(char *theKey,int CriteriaNo)=0;
                virtual BOOL RetFindRow(char *buffer,int CriteriaNo=0)=0;
                virtual BOOL Seek(char *theUniqueKey)=0;
                virtual BOOL SeekRange(char *theInitialKey,char *theFinalKey,int CriteriaNo=0)=0;
                virtual void Clear()=0;
                #ifdef DPCODEBASE
                        BOOL GoTop(void);
                        BOOL Go(long int recno=1L);
                        void SetSelectCriteria(int theCriteria);
                        int GetSelectCriteria(void);
                        BOOL Select(char *alias);
                        BOOL Select(int tableno);
                        BOOL SetOrderTo(char *ndxtag);
                        BOOL SSeek(char *key);
                        BOOL SSeek(double key);
                        BOOL Index(TAG4INFO *tag_info,int indexno=1);
                        BOOL Skip(long int recno=1L);
                        BOOL Pack(void);
                        BOOL Zap(void);
                        void RelateInit();
                        void RelateEnd();
                        void CreateSlave(char *masterexpr,char *slavetag,BOOL samemaster=FALSE);
                        void QuerySet(char *expression);
                        void SortSet(char *expression);
                        BOOL ExecuteQuery(void);
                        virtual BOOL SkipData(void);
                        BOOL SeekObj(char *theUniqueIdentifier,BOOL SetUniqueKey=FALSE);
                        void CreateCursor(void);
                        BOOL EndCursor(void);
                        void SaveToCursor(void);
                        BOOL ReadFromCursor(void);
            #else
                        BOOL ExecuteSQL(void);
                        BOOL ExecuteSQL(char *SQLStm);
                        BOOL RetCursorName(unsigned char far *CursorName);
                        BOOL SeekObj(char *theUniqueIdentifier,BOOL SetUniqueKey=FALSE);
                    #ifdef DPODBC
                                SWORD RetResultColNo(void);
                    #endif
                    #ifdef DPQELIB
                                int RetResultColNo(void);
                    #endif
            #endif
        };
#endif
```

Installazione di DB/2 NT

La seguente serie di note è stata concepita per facilitare l'attività d'installazione della versione NT di DB/2.

Si consiglia vivamente di installare la versione **in lingua inglese** del prodotto.

Installazione del DB2 Server WinNT Ver. 4.0.1

Tutte le operazioni dal punto 1 al punto 6 devono essere effettuate sul server.

1. Creazione di un nome utente per il gestore DB2

E' necessario creare un nome utente speciale che soddisfi i criteri di denominazione DB2. Questo utente verrà usato per installare ed eseguire attività di gestione DB2.

Il nome utente specificato deve avere gli stessi diritti dell'amministratore del server e deve rispettare le seguenti regole:

il nome può contenere **al massimo 8 caratteri**;

il nome non può iniziare con IBM, SYS, SQL o con un numero;

il nome non può essere un parola riservata DB2 (USERS, ADMINS, GUESTS, PUBLIC o LOCAL) o una parola chiave SQL;

il nome non può terminare con $;

il nome non può includere caratteri accentati.

Attenzione il nome utente "Administrator" assunto per default da WinNT per attività di installazione e gestione **non è valido** poichè contiene più di 8 caratteri.

2. Note

Accertarsi che tra i protocolli disponibili sia stato installato anche il protocollo **TCP/IP**.

Il programma di installazione tenta di sostituire alcuni file che potrebbero essere utilizzati da altre applicazioni o servizi, per questo motivo occorre chiudere eventuali altre applicazioni ed arrestare eventuali servizi in esecuzione prima di procedere con la installazione.

Il programma di installazione mantiene una registrazione che tiene traccia delle attività di installazione e disintallazione e delle informazioni relative agli eventuali errori incontrati. Il file DB2.LOG viene memorizzato nella subdirectory DB2LOG.

3. Installazione da CD-ROM

Dopo essersi collegati con il nome utente valido eseguire

```
<unità_cdrom>:\En\Disk1\setup.exe
```

dove `<unità_cdrom>` è la lettera che rappresenta l'unità CD-ROM.

Per l'esecuzione dei passi rimanenti è disponibile l'aiuto in linea.

Terminata l'installazione occorre riavviare il server.

4. Configurazione delle comunicazioni

Il protocollo di comunicazione consigliato è TCP/IP.
Impostazione della variabile di ambiente DB2COMM:

richiamare il '*Control panel*';

richiamare le impostazioni del sistema (icona '*System*');

fare click su una delle variabili di sistema (pagina '*Environment*' listbox '*System variables*');

modificare il nome nel campo '*Variable*' in DB2COMM;

modificare il valore nel campo '*Value*' in TCPIP;

premere '*Set*' e dopo '*Ok*' per uscire.

Attenzione è di fondamentale importanza che la variabile DB2COMM sia inserita tra le **variabili di sistema**, e non tra le variabili dell'utente, altrimenti il gestore del DB2 non utilizza il TCPIP come protocollo di comunicazione e dai client non si riesce a stabilire il collegamento.

Aggiornamento del file *services* sul server.

Occorre accedere ed editare il file *services*; questo file si trova nella subdirectory

`%SYSTEMROOT%\system32\drivers\etc`.

Il file *services* sul server deve contenere le due seguenti entry:

```
db2ins1     3700/tcp
db2ins1i    3701/tcp
```

dove `db2ins1` è il valore del parametro `SVCENAME` del DB2, 3700 e 3701 sono numeri di porta per la porta di collegamento e interruzione, `tcp` è il protocollo. Il numero di porta 3700 è arbitrario ma deve essere univoco all'interno del file sia sul server che sul client. Anche il secondo numero di porta deve essere univoco e uguale al primo numero più uno.

Aggiornamento del file di configurazione del *Database Director* sul server.

Utilizzare '*DB2 Command Window*'.

Digitare il seguente comando

```
DB2 UPDATE DATABASE MANAGER CONFIGURATION USING SVCENAME db2ins1
```

dove `db2ins1` è il nome del servizio.

Chiudere la finestra.

5. Avvio del DB2

Fare il '*Restart*' del server.

Collegarsi con il nome utente creato per la installazione del DB2. Richiamare il '*Control panel*' e avviare il il gestore dei servizi (icona '*Services*').

Avviare i servizi *DB2 - DB2* e *DB2 Security Server* (evidenziando le due linee una per volta e clickando su '*Start*'). Se si desidera che il DB2 si avvii automaticamente dopo un *Restart* evidenziare il servizio e fare click su '*Startup*' e sulla finestra successiva finestra fare click su '*Automatic*'.

6. Creazione dei database SIGLAPP e DITTA1

Dopo che il DB2 è stato fatto partire occorre creare i due database utilizzati da SIGLA. Il primo, che conterrà i dati comuni, deve necessariamente essere SIGLAPP, mentre il secondo, che conterrà i dati della ditta, può ad esempio essere DITTA1.

Utilizzare '*DB2 Command Window*'.

Digitare i seguenti comandi:

```
DB2 CREATE DATABASE SIGLAPP
DB2 CREATE DATABASE DITTA1
```

Per verificare la corretta creazione dei database
`DB2 CONNECT TO SIGLAPP`
in caso di successo il messaggio è

```
   Database Connection Information

 Database product        = DB2/NT 2.1.2
 SQL authorization ID    = <nome_utente>
 Local database alias    = SIGLAPP
```

Installazione dei client Win95, WinNT e Win311 (Workgroup) Ver. 2.1.2

I client sulla LAN che devono collegarsi al DB2 Server devono avere il *DB2 Client Application Enabler* installato. Prima di procedere occorre accertarsi che sia stato installato il protocollo di comunicazione TCP/IP.

Tutti gli utenti devono avere un nome utente che contenga **al massimo 8 caratteri**.
Si consiglia vivamente di installare la versione **in lingua inglese** del prodotto. Tutte le operazioni dal punto 1 al punto 4 devono essere effettuate su ciascun client.

1. Installazione del Client Application Enabler

Inserire il CDROM Client Pack ed eseguire

```
<unità_cdrom>:\Win32\I386\En\Disk1\setup.exe
```

dove `<unità_cdrom>` è la lettera che rappresenta l'unità CD-ROM.

Per l'esecuzione dei passi rimanenti è disponibile l'aiuto in linea.

2. Configurazione delle comunicazioni sul Client

Aggiungere un'entrata al file `HOSTS` sul client relativa all'hostname del server come segue:

```
9.21.15.235       servernt
```

dove `9.21.15.235` è l'indirizzo `IP` e `servernt` è l'hostname del server.

(N.B.: l'indirizzo numerico ed il nome simbolico del server sono qui riportati a titolo di esempio e devono essere sostituiticon quelli scelti in fase di installazione di WinNT Server).

Nel client WinNT il file `HOSTS` si trova nella subdirectory

```
%SYSTEMROOT%\system32\drivers\etc
```

mentre nel client Win95 si trova nella subdirectory

```
\windows
```

Se non si trova il file dovrebbe essere presente nella stessa subdirectory il file `HOSTS.SAM` che può essere copiato nel file `HOSTS`.

Aggiungere le seguenti entry al file `SERVICES` sul client:

```
db2ins1     3700/tcp
db2ins1i    3701/tcp
```

N.B.: i nomi dei servizi e i valori delle porte **devono corrispondere** a quelli impostati in precedenza sul server.
Nel client WinNT il file `SERVICES` si trova nella subdirectory
`%SYSTEMROOT%\system32\drivers\etc`

mentre nel client Win95 si trova nella subdirectory

```
\windows
```

3. Catalogazione del nodo TCP/IP sul client

Utilizzando il *DB2 Client Setup*:

selezionare il menù '*Node*'

selezionare '*New*'

inserire nel campo '*Name*' il nome da assegnare al nodo es. NODONT

inserire nel campo '*Comment*' un eventuale commento a piacere

inserire nel campo '*Hostname*' il nome del server (vd. file `HOSTS`) `servernt`

inserire nel campo '*Service name*' il nome del servizio (vd. file `SERVICES`) `db2ins1`

terminare selezionando '*Ok*'

4. Catalogazione dei database sul client

Per il client Windows for Workgroup occorre prima installare il driver ODBC clickando su *ODBC Installer.* Utilizzando il *DB2 Client Setup*:

selezionare il nodo che è stato creato al punto precedente (es. NODONT)

premere il bottone '*Databases*', si apre la nuova finestra *DB2 Client Setup - Databases*

selezionare il menù '*Databases*'

selezionare '*New*'

inserire SIGLAPP nel campo '*Name*'

inserire SIGLAPP nel campo '*Alias*' (questo è il nome utilizzato dal client per collegarsi al database)

premere *Test Database Connection* per provare la connessione al database DB2, nella successiva finestra inserire nel campo '*User Name*' il nome utente per il gestore DB2 e nel campo '*Password*' l'eventuale password

terminare selezionando '*Ok*'

Ripetere le stesse operazioni per il database DITTA1.

Verificare che nella finestra *DB2 Client Setup - Databases* sotto la colonna *ODBC Data Source* sia presente la scritta *Yes*. In caso contrario selezionare la riga che si vuole modificare, premere il tasto destro e fare click sulla voce *ODBC data source*.

A questo punto l'installazione del Database DB2 è completata le fasi successive (creazione delle tabelle, riempimento delle tabelle precaricate ecc.) devono essere effettuate dal programma di ***Configurazione*** di ***SIGLA.***

Tuning del database (per tutte le versioni)

La configurazione dei vari parametri del database non può ovviamente essere generalizzata poichè è strettamente dipendente dalla configurazione globale della installazione (memoria ram e fisica del server, volume dei dati da gestire, numero di utenti collegati ecc.). Pertanto non è possibile indicare la configurazione ottimale dei vari parametri del database, i quali sono tutti illustrati nella documentazione installata, o comunque presente nel CD-ROM, alla voce '*Administrator Guide - Configuring DB2*'.

In fase di creazione del database alcuni parametri vengono inizializzati con dei valori di default non adeguati alla normale attività con SIGLA, nel seguito saranno proposti dei valori indicativi per tali parametri.

Per modificarli utilizzare '*DB2 Command Window*'.

Il primo parametro da modificare è relativo al database manager ed è quindi generale per tutti i database che saranno creati; digitare il seguente comando:

```
db2 update dbm cfg using QUERY_HEAP_SZ 1000
```

Gli altri parametri da modificare sono invece relativi ai database creati. Devono essere modificati tre parametri del database creato per i dati della ditta (es. DITTA1); digitare i seguenti comandi:

```
db2 update db cfg for DITTA1 using APPLHEAPSZ 512
db2 update db cfg for DITTA1 using PCKCACHESZ 128
db2 update db cfg for DITTA1 using BUFFPAGE 1024
```

Altri parametri possono essere modificati per ottimizzare le performance (anche se il valore ottimale potrebbe non essere quello indicato); digitare i seguenti comandi:

```
db2 update db cfg for DITTA1 using NUM_IOCLEANERS 3
db2 update db cfg for DITTA1 using NUM_IOSERVERS 9
db2 update db cfg for DITTA1 using LOCKLIST 150
```

Altri due parametri riguardano il numero di applicazioni che si possono connettere al database (in questo caso occorre considerare anche la presenza di eventuali personalizzazioni o strumenti di office automation che accedano al database); digitare i seguenti comandi:

```
db2 update db cfg for DITTA1 using MAXAPPLS <almeno pari al numero di client * 2>
db2 update db cfg for DITTA1 using AVG_APPLS <numero di client mediamente attivi>
```

Altri parametri fondamentali sono legati ai file di log (utilizzati dal DB/2 per la gestione delle transazioni); digitare i seguenti comandi:

```
db2 update db cfg for DITTA1 using LOGFILSIZ 256
db2 update db cfg for DITTA1 using LOGPRIMARY <numero di utenti + 5>
db2 update db cfg for DITTA1 using LOGSECOND <logprimary / 2>
```

In questo caso è fondamentale disporre di sufficiente spazio su disco: tenere presente che la quantità di spazio su disco occupata dai file di log, in bytes, varia da:

```
(LOGPRIMARY * (LOGFILSIZ+2) * 4096) + 8192
```

a:

```
((LOGPRIMARY + LOGSECOND) * (LOGFILSIZ+2) * 4096) + 8192
```

Se durante l'attività risultasse necessario aumentare ulteriormente il numero dei file di log il database manager restituisce l'errore con codice **SQL0964C**. Poiché i file di log vengono generati alla prima connessione che viene effettuata da una applicazione al database, occorre effettuare il '*Restart*' del server per rendere effettive le modifiche.

Un parametro fondamentale da non modificare è `LOCKTIMEOUT` che deve essere lasciato al valore di default –1; per ripristinarlo digitare il seguente comando:

```
db2 update db cfg for DITTA1 using LOCKTIMEOUT -1
```

Dopo queste operazioni occorre fermare e poi far ripartire il database DB2 (utilizzare la gestione dei servizi del server WinNT).

I valori indicati in questa sezione sono riferiti ad una configurazione con almeno 64MB di ram; in linea di massima maggiore è la quantità di ram installata e più efficiente sarà il DB/2. Con una quantità di ram superiore è possibile, ad esempio, aumentare ulteriormente i valori attribuiti ai parametri `NUM_IOCLEANERS` e `NUM_IOSERVERS`.

Queste impostazioni potrebbero necessitare di ulteriori modifiche con il passare del tempo (e quindi con ad esempio con l'aumento del volume dei dati), percui occorre una verifica periodica delle performance del database.

Periodicamente è consigliabile anche calcolare le statistiche sulle caratteristiche fisiche delle tabelle e degli indici associati al fine di determinare se alcune tabelle necessitano di essere riorganizzate. L'ottimizzatore utilizza queste statistiche per determinare i percorsi di accesso ai dati. **N.B.**: queste operazioni devono essere effettuate quando nessuna applicazione è collegata al database.

Sempre da '*DB2 Command Window*', dopo essersi connessi al database, digitare il seguente comando:

```
db2 reorgchk update statistics on table user
```

oppure, per controllare le statistiche correnti (senza aggiornarle):

```
db2 reorgchk current statistics on table user
```

per controllare le statistiche anche delle tabelle di sistema:

```
db2 reorgchk current statistics on table all
```

Per una spiegazione dell'output del comando (che conviene redirigere in un file ascii), rimandiamo alla documentazione di DB/2. L'analisi di tale output consentirà di capire quali tabelle e/o indici dovranno essere interessate dai successivi comandi.

E' possibile aggiornare le statistiche di ciascuna tabella e dei suoi indici con il seguente comando da digitare per ogni tabella:

```
db2 runstats on table <utente>.<tabella> with distribution and detailed indexes all
```

Questo comando dovrebbe essere effettuato se la tabella è stata modificata significativamente a causa di un grande numero record modificati, inseriti o cancellati e se la tabella è stata riorganizzata.

Se una tabella deve essere riorganizzata occorre utilizzare il seguente comando:

```
db2 reorg table <utente>.<tabella>
```

per riorganizzare una tabella secondo un indice specifico occorre utilizzare il seguente comando:

```
db2 reorg table <utente>.<tabella> index <utente>.<indice>
```

Il database manager registra nel file DB2DIAG.LOG tutte le informazioni su errori o avvertimenti che si presentano durante il normale funzionamento, poiché tale file non viene mai svuotato occorre tenerne sotto controllo la dimensione e cancellarlo periodicamente per liberare spazio su disco. Nella installazione standard tale file è situato nella subdiretory:

```
\sqllib\db2.
```

Installazione di DB/2 Single User

La seguente serie di note è stata concepita per facilitare l'attività d'installazione della versione Single User per WinNT Workstation 4.0 e Win95 di DB/2.

Si consiglia vivamente di installare la versione **in lingua inglese** del prodotto.

Installazione del DB2 Single User Ver. 2.1.2 per WinNT-Wks 4.0 e Win95

1. Installazione da CD-ROM

Eseguire

<unità_cdrom>:\i386\En\Disk1\setup.exe

dove <unità_cdrom> è la lettera che rappresenta l'unità CD-ROM.
Per l'esecuzione dei passi rimanenti è disponibile l'aiuto in linea.

2. Note

E' necessario attivare la gestione degli utenti e creare un utente il cui nome soddisfi i criteri di denominazione DB2 illustrati in precedenza per la installazione del *DB2 Server* (**N.B.: max 8 caratteri**).

3. Avvio del DB2

Riavviare il computer.
Utilizzare '*DB2 Command Line Processor*' e digitare il seguente comando:
DB2START

Per fermare il gestore del database, sempre da '*DB2 Command Line Processor*', digitare il seguente comando:
DB2STOP

Alternativamente si può utilizzare '*DB2 Command Windows*', il tal caso i comandi da digitare sono:
DB2 DB2START
DB2 DB2STOP

4. Creazione dei database SIGLAPP e DITTA1

Si deve procedere esattamente come illustrato nel caso della installazione del *DB2 Server*.

5. Catalogazione dei database

Utilizzando il *DB2 Client Setup*:

selezionare il nodo '*This node*'.

premere il bottone '*Databases*', si apre la nuova finestra *DB2 Client Setup - Databases*

selezionare il menù '*Databases*'

selezionare '*New*'

inserire SIGLAPP nel campo '*Name*'

inserire SIGLAPP nel campo '*Alias*' (questo è il nome utilizzato per collegarsi al database)

premere *Test Database Connection* per provare la connessione al database DB2, nella successiva finestra inserire nel campo '*User Name*' il nome utente del personal e nel campo '*Password*' l'eventuale password

terminare selezionando '*Ok*'

Ripetere le stesse operazioni per il database DITTA1.

Verificare che nella finestra *DB2 Client Setup - Databases* sotto la colonna *ODBC Data Source* sia presente la scritta *Yes*. In caso contrario selezionare la riga che si vuole modificare, premere il tasto destro e fare click sulla voce *ODBC data source*.
Per le operazioni di tuning del database occorre riferirsi alla corrispondente sezione nelle note di installazione del DB2 Server. A questo punto l'installazione del Database DB2 è completata le fasi successive (creazione delle tabelle, riempimento delle tabelle precaricate ecc.) devono essere effettuate dal programma di ***Configurazione*** di ***SIGLA.***

Installazione di DB/2 NT UDB

La seguente serie di note è stata concepita per facilitare l'attività d'installazione della versione NT di DB2 Universal Database (versione 5.2).

Si consiglia vivamente di installare la versione **in lingua inglese** del prodotto.

Installazione del DB2 Universal Database Server WinNT Ver. 5.2

Tutte le operazioni dal punto 1 al punto 5 devono essere effettuate sul server.

1. Note

Accertarsi che tra i protocolli disponibili sia stato installato anche il protocollo **TCP/IP**.

Il programma di installazione tenta di sostituire alcuni file che potrebbero essere utilizzati da altre applicazioni o servizi, per questo motivo occorre chiudere eventuali altre applicazioni ed arrestare eventuali servizi in esecuzione prima di procedere con la installazione.

Il programma di installazione mantiene una registrazione che tiene traccia delle attività di installazione e disintallazione e delle informazioni relative agli eventuali errori incontrati. Il file DB2.LOG viene memorizzato nella subdirectory DB2LOG.

2. Installazione da CD-ROM

Una volta inserito il CDROM viene eseguito automaticamente il setup dalla procedura di autorun. In caso contrario nella directory principale del CDROM è presente il file SETUP.EXE per l'installazione.
Tra i prodotti da installare occorre selezionare 'DB2 Universal Database Workgroup Edition' e 'DB2 Client Application Enabler'.

Conviene selezione la Custom Installation come tipo di installazione ed accettare le impostazioni. La finestra denominata 'Customize Comunication Protocols' consente di verificare e configurare le impostazioni per i protocolli di comunicazione dell'istanza del database manager e l'utente abilitato alla gestione del database (cfr. Administration Server).
Il sistema di help in linea fornisce tutte le informazioni sulle varie fasi della installazione.

3. Configurazione delle comunicazioni

Il protocollo di comunicazione consigliato è TCP/IP.
Impostazione della variabile di ambiente DB2COMM:

- richiamare il '*Control panel*';
- richiamare le impostazioni del sistema (icona '*System*');
- fare click su una delle variabili di sistema (pagina '*Environment*' listbox '*System variables*');
- modificare il nome nel campo '*Variable*' in DB2COMM;
- modificare il valore nel campo '*Value*' in TCPIP;
- premere '*Set*' e dopo '*Ok*' per uscire.

Attenzione è di fondamentale importanza che la variabile DB2COMM sia inserita tra le **variabili di sistema**, e non tra le variabili dell'utente, altrimenti il gestore del DB2 non utilizza il TCPIP come protocollo di comunicazione e dai client non si riesce a stabilire il collegamento.

Richiamare il tool '*Control Center*' e per l'istanza DB2 configurare il nome del servizio TCP/IP da utilizzare per le comunicazioni. L'installazione standard utilizza il servizio db2cDB2 (con valore 50000), quindi nel folder 'Comunication' della finestra di configurazione della istanza attribuire il valore db2cDB2 al parametro svcename.

Verificare che nel file services siano presenti i dati relativi ai due servizi utilizzati dall'istanza DB2 per le comunicazioni. L'installazione standard utilizza il servizio db2cDB2 (con valore 50000), quindi nel file `SERVICES` (situato nella directory `%SYSTEMROOT%\system32\drivers\etc`) devono essere presenti le seguenti righe:

```
db2cDB2      50000/TCP
db2cDB2i     50001/TCP
```

4. Avvio del DB2 UDB

Fare il '*Restart*' del server.

Collegarsi con il nome utente creato per l'installazione di DB2 UDB. Richiamare il '*Control panel*' e avviare il gestore dei servizi (icona '*Services*').

Avviare i servizi *DB2 - DB2*, *DB2 – DB2DAS00* e *DB2 Security Server* (evidenziando le tre linee una per volta e clickando su '*Start*'). Se si desidera che il DB2 UDB si avvii automaticamente dopo un *Restart* evidenziare i servizi e fare click su '*Startup*' e sulla finestra successiva fare click su '*Automatic*'.

5. Creazione dei database SIGLAPP e DITTA1

Può essere seguita la stessa procedura indicata per la precedente versione di DB2. In alternativa è possibile utilizzare il tool '*Control Center*'.

Installazione dei client Win95, Win98, WinNT Workstation

I client sulla LAN che devono collegarsi al DB2 Server devono avere il *DB2 Client Application Enabler* installato. Prima di procedere occorre accertarsi che sia stato installato il protocollo di comunicazione TCP/IP.

Tutti gli utenti devono avere un nome utente che contenga **al massimo 8 caratteri**.

La procedura di autorun esegue automaticamente la procedura di setup che in base al sistema operativo consente di installare il software opportuno. Automaticamente viene installata la versione nella lingua corrispondente al sistema operativo, è comunque possibile installare la versione **in lingua inglese** del prodotto eseguendo il setup (`DB2INST.EXE`) direttamente dalla cartella `\DB2\EN\INSTALL` del CDROM di installazione.

Tutte le operazioni dal punto 1 al punto 2 devono essere effettuate su ciascun client.

1. Installazione del Client Application Enabler

Lo stesso CDROM impiegato per l'installazione del server contiene il software per il client. Una volta inserito il CDROM viene eseguito automaticamente il setup dalla procedura di autorun. In caso contrario nella directory principale del CDROM è presente il file SETUP.EXE per l'installazione.

Non è necessario installare i componenti per l'amministrazione remota del server in tutti i client.

Il sistema di help in linea fornisce tutte le informazioni sulle varie fasi della installazione.

2. Configurazione del Client

Il tool '*Client Configuration Assistant*' consente di configurare i collegamenti necessari per permettere alle applicazioni client la comunicazione con DB2.

La pressione del bottone '*Add*' attiva un'ulteriore finestra che mette a disposizione varie opzioni: scegliendo '*Search the network*' viene attivata una procedura automatica.

L'opzione '*Manually configure a connection to a DB2 database*' consente di eseguire la stessa procedura passo passo ed in forma non automatica: per prima cosa occorre scegliere il protocollo TCP/IP, successivamente indicare il nome del server e il numero della porta (50000 nella installazione standard) e il nome del servizio (db2cDB2 nella installazione standard), il nome del database e il suo alias (il database dei dati comuni deve comunque avere SIGLAPP come alias).

E' disponibile anche una terza opzione, '*Use an access profile*', che consente di utilizzare un file di configurazione generato dal server. Per ottenere il file occorre utilizzare il '*Control Center*', facendo click con il tasto destro del mouse sull'icona del sistema '*Local*' si attiva un menù dal quale occorre selezionare la voce '*Generate access profile*' (il file generato deve essere copiato in una directory accessibile a tutti i client). In fase di configurazione del client è sufficiente indicare il nome del file e selezionare i database.

Ottimizzazione della configurazione dei database

Per la modifica dei vari parametri di configurazione del database occorre seguire le istruzioni riportate nel presente manuale tecnico paragrafo relativo alla installazione del DB2 NT ver.4.0.1 alla voce 'Tuning del database'. La procedura indicata prevede di utilizzare la '*DB2 Command Window*', ma è comunque possibile operare le stesse modifiche anche utilizzando il tool '*Control Center*'.

Descrizione dei dataset XML delle stampe

Nel seguito sono illustrati gli schemi dei file xml contenenti i dataset delle varie stampe eseguite da SIGLA Ultimate e Start Edition.

Questa documentazione potrà subire, nel corso del tempo, aggiornamenti e/o modifiche volte a migliorarne la chiarezza e la precisione oppure necessarie a recepire le nuove implementazioni inserite nel prodotto.

I dataset xml si distinguono in due tipi: il primo, il cui nome inizia con "$S", contiene i dati delle stampe generiche (stampa di un report) mentre il secondo, il cui nome inizia con "$X" o "$P", contiene i dati della stampa dei documenti (se il nome inizia con "$X" il file contiene i dati di tutti i documenti interessati dalla singola sessione di stampa, mentre se il nome inizia con "$P" il file contiene i dati di un singolo documento). L'estensione del file è in tutti i casi ".tmp" anche se si tratta di un file XML.

Convenzioni utilizzate

I nomi degli elementi e degli attributi (tag) sono tutti in caratteri maiuscoli e sono espressi in lingua inglese.

I nomi dei campi delle delle varie tabelle del database sono espressi nella forma *nometabella.nomecampo* e sono sempre scritti in caratteri maiuscoli. Il tipo del valore di questi campi corrisponde esattamente al tipo definito nella tabella del database e può essere ricavato dalla definizione della tabella o dal file di documentazione della struttura del database tecnotes.chm.

I dati che non provengono dal database sono indicati come campi calcolati ed il come è nella forma *CALCOLATO.nomecampo* (in questo caso il nome del campo è espresso sia in caratteri maiuscoli che minuscoli). Per questi campi sono indicati, come attributi dell'elemento, anche il tipo e la lunghezza (per il tipo carattere) o in numero di decimali (per il tipo numerico). I possibili tipi di dato sono:

- carattere (indicato con la lettera 'C'),
- numero decimale (indicato con la lettera 'D'),
- numero intero (indicato con la lettera 'I'),
- data in formato ansi AAAAMMGG (indicata con la lettera 'A'),
- ora in formato HHMM (indicata con la lettera 'H').

Nei dataset relativi alla stampa dei documenti tutti i campi possibili sono sempre di tipo calcolato e sono contraddistinti, come ulteriore attributo, da un identificativo numerico (il codice del campo). I campi con codice 0 sono gestiti in forma automatica dalla procedura e non hanno un corrispondente campo nella lista dei campi configurabili mostrata dal programma di configurazione. Per un elenco dei campi disponibili è possibile consultare il file di documentazione cfgstampe.xls (si tenga presente che rispetto a quan to indicato in tale documento al codice dei campi del corpo è stato aggiunto il valore 1000 e a quelli del piede il valore 2000).

Dataset della stampa di un report

Il nome del file inizia per '$S' ed ha estensione .tmp. Lo schema generale è composto dall'elemento REPORTS, radice (root) del file xml, che contiene l'elemento obbligatorio INFO, uno o più elementi REPORT (un elemento REPORT è obbligatorio, quindi almeno uno è sempre presente) e l'elemento opzionale PARAMETERS.

Padre	*Figli*	*Note*
REPORTS		Root del file xml.
	INFO	Elemento obbligatorio.
	REPORT	Elemento obbligatorio.
	…	Eventuali ulteriori elementi REPORT.
	PARAMETERS	Elemento opzionale.

L'elemento INFO contiene gli elementi obbligatori APPLICATION e FILEVERSION.

Padre	*Figli*	*Note*
INFO		
	APPLICATION	Il valore è SIGLA se il dataset è stato generato da SIGLA Ultimate, START se è stato

		generato da SIGLA Start Edition.
	FILEVERSION	Indica la versione dell'applicazione che ha prodotto il dataset (es. 4.2.1.0).

L'elemento REPORT è caratterizzato da alcuni attributi e contiene l'elemento obbligatorio DATASET e l'elemento opzionale PARAMETERS.

Elemento	*Attributi*	*Note*
REPORT		
	ID	Identificativo univoco della stampa.
	TITLE	Titolo della stampa.
	ORIENT	Orientamento della pagina ('L' per orizzontale/landscape o 'P' per verticale/portrait).
	DATAMEMBER	Vale sempre 'Corpo'.
	REPORTINFOTABLE	Vale sempre 'Testata'.

Padre	*Figli*	*Note*
REPORT		
	DATASET	Elemento obbligatorio.
	PARAMETERS	Elemento opzionale.

L'elemento DATASET è caratterizzato dall'attributo NAME e contiene gli elementi obbligatori SCHEMA e DATA.

Elemento	*Attributi*	*Note*
DATASET		
	NAME	Nome del dataset.

Padre	*Figli*	*Note*
DATASET		
	SCHEMA	Contiene la definizione della struttura del dataset, ovvero l'elenco delle tabelle, con la descrizione dei campi, e le relazioni.
	DATA	Contiene i dati.

L'elemento SCHEMA contiene due o più elementi TABLE (due sono obbligatori) e uno o più elementi opzionali RELATION.

Padre	*Figli*	*Note*
SCHEMA		
	TABLE	Descrive la struttura della tabella. Gli elementi con attributo NAME uguale a 'Corpo' e 'Testata' sono sempre presenti.
	…	Eventuali ulteriori elementi TABLE.

	RELATION	Descrive la relazione tra le tabelle. La relazione tra le tabelle 'Testata', 'Corpo' e 'Piede' è implicita e non viene descritta con un elemento RELATION. Elemento opzionale
	…	Eventuali ulteriori elementi RELATION

L'elemento TABLE è caratterizzato dall'attributo NAME e contiene uno o più elementi COLUMN.

Elemento	*Attributi*	*Note*
TABLE		
	NAME	Nome della tabella.

Padre	*Figli*	*Note*
TABLE		
	COLUMN	Contiene la definizione di un campo. Per i campi estratti da tabelle del database di SIGLA si indica solo il nome mentre per i campi calcolati in fase di elaborazione della stampa si indica anche il tipo e la lunghezza o il numero dei decimali.
	…	Ulteriori elementi COLUMN.

L'elemento COLUMN è caratterizzato dall'attributo NAME (obbligatorio) più altri attributi opzionali e non contiene ulteriori elementi.

Elemento	*Attributi*	*Note*
COLUMN		
	NAME	Nome del campo. I campi delle delle varie tabelle del database hanno un nome nella forma *NOMETABELLA.NOMECAMPO* e sono scritti sempre in caratteri maiuscoli, mentre i campi calcolati in fase di elaborazione della stampa lo hanno nella forma *CALCOLATO.NomeCampo*.
	TYPE	Tipo del campo: • 'C' per carattere • 'D' per numero decimale • 'I' per numero intero • 'A' per data (AAAAMMGG) • 'H' per ora (HHMM) Presente solo per i campi calcolati.
	LEN	Lunghezza del campo. Presente solo per i campi calcolati di tipo carattere.

	DEC	Numero dei decimali. Presente solo per i campi calcolati di tipo numerico.

L'elemento RELATION è caratterizzato dagli attributi obbligatori NAME, MASTER, SLAVE e non contiene ulteriori elementi.

Elemento	*Attributi*	*Note*
RELATION		
	NAME	Nome della relazione.
	MASTER	Nome della tabella master.
	SLAVE	Nome della tabella slave.

L'elemento DATA contiene due o più elementi TABLE (due sono obbligatori).

Padre	*Figli*	*Note*
DATA		
	TABLE	Contiene le varie righe della tabella. Gli elementi con attributo NAME uguale a 'Corpo' e 'Testata' sono sempre presenti.
	…	Eventuali ulteriori elementi TABLE.

L'elemento TABLE è caratterizzato dall'attributo NAME e contiene uno o più elementi ROW.

Elemento	*Attributi*	*Note*
TABLE		
	NAME	Nome della tabella.

Padre	*Figli*	*Note*
TABLE		
	ROW	Contiene i valori di una riga della tabella.
	…	Ulteriori elementi ROW.

L'elemento ROW contiene uno o più elementi COLUMN.

Padre	*Figli*	*Note*
ROW		
	COLUMN	Contiene il valore del campo descritto dall'attributo NAME.
	…	Ulteriori elementi COLUMN.

L'elemento COLUMN è caratterizzato dall'attributo NAME e il suo valore contiene il valore del campo.

Elemento	*Attributi*	*Note*
COLUMN		Il valore dell'elemento corrisponde al valore assunto dal campo.
	NAME	Nome del campo.

L'elemento PARAMETERS contiene uno o più elementi PARAMETER. Questi elementi contengono informazioni che non hanno nessun collegamento con i dati della stampa ma che sono utilizzate nella fase di stampa del documento.

Padre	*Figli*	*Note*
PARAMETERS		
	PARAMETER	Contiene i valori di una riga della tabella.

L'elemento PARAMETER è caratterizzato da alcuni attributi e da un valore.

Elemento	*Attributi*	*Note*
PARAMETER		
	ID	Identificativo numerico univoco dell'elemento.
	NAME	Descrizione
	TYPE	Tipo del valore dell'elemento: • 'C' per carattere • 'D' per numero decimale • 'I' per numero intero • 'A' per data (AAAAMMGG) • 'H' per ora (HHMM)
	KIND	Tipo dell'elemento

Le tabelle 'Testata' e 'Corpo' sono sempre presenti. La tabella 'Testata' contiene un solo record e tra le sue colonne sono presenti nuovamente il titolo e l'identificativo numerico univoco della stampa.

Dataset della stampa di un documento

Il nome del file inizia per '$P' ed ha estensione .tmp. Lo schema generale è composto dall'elemento REPORTS, radice (root) del file xml, che contiene l'elemento obbligatorio INFO e l'elemento obbligatorio REPORT.

Padre	*Figli*	*Note*
REPORTS		Root del file xml.
	INFO	Elemento obbligatorio.
	REPORT	Elemento obbligatorio.
	PARAMETERS	Elemento opzionale.

L'elemento INFO contiene gli elementi obbligatori APPLICATION e FILEVERSION.

Padre	*Figli*	*Note*
INFO		
	APPLICATION	Il valore è SIGLA se il dataset è stato generato da SIGLA Ultimate, START se è stato generato da SIGLA Start Edition.
	FILEVERSION	Indica la versione dell'applicazione che ha prodotto il dataset (es. 4.2.1.0).

L'elemento REPORT è caratterizzato da alcuni attributi e contiene l'elemento obbligatorio DATASET e l'elemento opzionale PARAMETERS.

Elemento	*Attributi*	*Note*

REPORT		
	ID	Vale sempre 0.
	TITLE	Descrizione del tipo effettivo del documento.
	ORIENT	Orientamento della pagina (‘L’ per orizzontale/landscape o ‘P’ per verticale/portrait).
	DATAMEMBER	Vale sempre ‘Testata’.
	REPORTINFOTABLE	Vale sempre ‘Testata’.
	TYPE	Codice del tipo effettivo del documento: • ‘B’ per bolla di carico • ‘G’ per bolla di scarico • ‘F’ per fattura • ‘R’ per fattura riepilogativa • ‘P’ per fattura proforma • ‘A’ per fattura accompagnatoria • ‘M’ per movimento di magazzino • ‘N’ per nota di credito • ‘O’ per ordine cliente • ‘T’ per ordine fornitore • ‘V’ per preventivo • ‘U’ per ricevuta fiscale • ‘S’ per scontrino • ‘C’ per documento generico
	LAYOUT	Tipo di fincato: • ‘0’ per tipo A • ‘1’ per tipo B • ‘2’ per tipo C • ‘3’ per tipo D • ‘4’ per tipo E • ‘5’ per tipo F

Padre	*Figli*	*Note*
REPORT		
	DATASET	Elemento obbligatorio.
	PARAMETERS	Elemento opzionale.

L’elemento DATASET è caratterizzato dall’attributo NAME e contiene gli elementi obbligatori SCHEMA e DATA.

Elemento	*Attributi*	*Note*
DATASET		
	NAME	Vale sempre ‘Dataset0’.

Padre	*Figli*	*Note*
DATASET		
	SCHEMA	Contiene la definizione della struttura del dataset, ovvero

		l'elenco delle tabelle, con la descrizione dei campi, e le relazioni.
	DATA	Contiene i dati.

L'elemento SCHEMA contiene tre o più elementi TABLE (tre sono obbligatori) e due o più elementi obbligatori RELATION.

Padre	*Figli*	*Note*
SCHEMA		
	TABLE	Descrive la struttura della tabella. Gli elementi con attributo NAME uguale a 'Testata', 'Corpo' e 'Piede' sono sempre presenti.
	…	Eventuali ulteriori elementi TABLE.
	RELATION	Descrive la relazione tra le tabelle. Le relazioni tra le tabelle 'Testata' e 'Corpo' e tra 'Testata' e 'Piede' sono sempre presenti.
	…	Eventuali ulteriori elementi RELATION.

L'elemento TABLE è caratterizzato dall'attributo NAME e contiene uno o più elementi COLUMN.

Elemento	*Attributi*	*Note*
TABLE		
	NAME	Nome della tabella.

Padre	*Figli*	*Note*
TABLE		
	COLUMN	Contiene la definizione di un campo.
	…	Ulteriori elementi COLUMN.

L'elemento COLUMN è caratterizzato da alcuni attributi tutti obbligatori e non contiene ulteriori elementi.

Elemento	*Attributi*	*Note*
COLUMN		
	CODE	Codice del campo. I campi con codice 0 sono gestiti in forma automatica dalla procedura e non hanno un corrispondente campo nella lista dei campi configurabili mostrata dal programma di configurazione.
	NAME	Nome del campo (nella forma *CALCOLATO.NomeCampo*).
	TYPE	Tipo del campo: • 'C' per carattere • 'D' per numero decimale • 'I' per numero intero

		• ‘A’ per data (AAAAMMGG) • ‘H’ per ora (HHMM).
	LEN	Lunghezza del campo. Presente solo per i campi di tipo carattere.

L’elemento RELATION è caratterizzato dagli attributi obbligatori NAME, MASTER, SLAVE e non contiene ulteriori elementi.

Elemento	*Attributi*	*Note*
RELATION		
	NAME	Nome della relazione.
	MASTER	Nome della tabella master.
	SLAVE	Nome della tabella slave.

L’elemento DATA contiene tre o più elementi TABLE (‘Testata’, ‘Corpo’ e ‘Piede’ sono obbligatori).

Padre	*Figli*	*Note*
DATA		
	TABLE	Contiene le varie righe della tabella. Gli elementi con attributo NAME uguale a ‘Testata’, ‘Corpo’ e ‘Piede’ sono sempre presenti.
	…	Eventuali ulteriori elementi TABLE.

L’elemento TABLE è caratterizzato dall’attributo NAME e contiene uno o più elementi ROW.

Elemento	*Attributi*	*Note*
TABLE		
	NAME	Nome della tabella.

Padre	*Figli*	*Note*
TABLE		
	ROW	Contiene i valori di una riga della tabella.
	…	Ulteriori elementi ROW.

L’elemento ROW contiene uno o più elementi COLUMN.

Padre	*Figli*	*Note*
ROW		
	COLUMN	Contiene il valore del campo descritto dall’attributo NAME.
	…	Ulteriori elementi COLUMN.

L’elemento COLUMN è caratterizzato da alcuni attributi obbligatori e il suo valore contiene il valore del campo.

Elemento	*Attributi*	*Note*
COLUMN		Il valore dell’elemento corrisponde

		al valore assunto dal campo.
	CODE	Codice del campo.
	NAME	Nome del campo.

L'elemento PARAMETERS contiene uno o più elementi PARAMETER. Questi elementi contengono informazioni che non hanno nessun collegamento con i dati della stampa ma che sono utilizzate nella fase di stampa del documento.

Padre	*Figli*	*Note*
PARAMETERS		
	PARAMETER	Contiene i valori di una riga della tabella.

L'elemento PARAMETER è caratterizzato da alcuni attributi e da un valore.

Elemento	*Attributi*	*Note*
PARAMETER		
	ID	Identificativo numerico univoco dell'elemento.
	NAME	Descrizione
	TYPE	Tipo del valore dell'elemento: • 'C' per carattere • 'D' per numero decimale • 'I' per numero intero • 'A' per data (AAAAMMGG) • 'H' per ora (HHMM)
	KIND	Tipo dell'elemento

Le tabelle 'Testata', 'Corpo' e 'Piede' sono sempre presenti.

Il file xml contenete il dataset di una serie di documenti (quelli interessati dalla sessione dei stampa) è analogo a questo ma ovviamente contiene i dati relativi a più di un documento (nelle tabelle 'Testata' e 'Piede' è presente più di un record). I singoli documenti sono identificati attraverso un numero univoco inserito in ogni tabella nel campo 'CALCOLATO.ID_Documento'. Il nome del file contenente il dataset di una serie di documenti inizia con "$X" ed ha estensione ".tmp".

Problem Report

Di seguito sono elencati alcuni dei problemi che sono stati riscontrati durante l'attività di installazione o di uso del pacchetto e i consigli per la soluzione degli stessi. Poiché le variabili in gioco (sistema operativo del client, versione di SIGLA usata, tipo e versione del DBMS, supporto di rete usato etc.) consentono la generazione di un numero elevatissimo di possibilità, i problemi sono stati catalogati fornendo, per quanto possibile, la descrizione dell'environment di utilizzo della procedura e sono stati raggruppati in ragione del DBMS utilizzato.

Oracle

Problem ID	Environment	Descrizione	Soluzione
#0001	RDBMS vers. 7.1, Clients Win16 driver ODBC per 7.0	In alcune stampe il primo rigo viene ripetuto in ultima posizione	Installare il driver ODBC per la versione 7.1 del DBMS
#0002	RDBMS per Netware usato da clients con Win32 e supporto SQL Net a 16 bit	L'esecuzione di statements SQL avviene in modo incompleto senza la generazione di errori se le righe coinvolte superano un numero critico	Installare la versione a 32 bit di SQL Net e la versione a 32 bit di SIGLA
#0016	RDBMS vers. 7.3 o succ.	La procedura si blocca con un codice d'errore tutte le volte che deve essere creata una nuova tabella (anche temporanea)	Installare la versione 1.64 o successiva

xBase

Problem ID	Environment	Descrizione	Soluzione
#0003	Win16/Win32 con SIGLA a 16 bit	SIGLA genera un errore di protezione generale durante l'attività sul magazzino	Disattivare gli statements SQL preparati utilizzando la procedura di configurazione
#0004	Win32 con SIGLA a 32 bit	Il driver ODBC genera un errore operando sulla tabella NOTE	SIGLA 32 non è certificato per l' uso con driver ODBC Microsoft a 32 bit. Il nome della tabella NOTE viene interpretato in modo anomalo dai drivers elencati che generano un errore di sintassi SQL. Non esiste al momento alcuna soluzione
#0005	Win16/Win32 con SIGLA a 16 bit, driver Microsoft per FoxPro versione 1.01.2115	I records cancellati continuano ad essere visibili dalla procedura	Cambiare driver (si consiglia di usare il driver per dBIV distribuito insieme alla procedura)

SQL Server

Problem ID	Environment	Descrizione	Soluzione
#0006	Win16/Win32	La procedura di ricalcolo saldi di magazzino produce un messaggio d'errore del server	SQL Server non è in grado di eseguire una clausola GROUP BY su più di quindici campi. **La versione 1.60b di SIGLA fissa il problema** evitando, solo per SQL Server, di far eseguire raggruppamenti di dati al server. Ne consegue che su questa particolare piattaforma la procedura di ricalcolo saldi può

			richiedere più tempo per essere eseguita. **Installare la versione 1.60b o successive.**
#0007	Win16/Win32	I saldi in linea degli articoli di magazzino non vengono aggiornati correttamente.	SQL Server non esegue correttamente statements SQL con clausole WHERE complesse [a.e UPDATE GIACESE SET CARICOQ=CARICOQ+100. WHERE (ARTICOLO='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' AND ESERCIZIO='1997' AND MAGAZZINO IS NULL) OR (ARTICOLO='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' AND ESERCIZIO='1997' AND MAGAZZINO='001')] La versione 1.60b del pacchetto fissa il problema splittando gli statements troppo complessi in più statements semplificati. **Installare la versione 1.60b o successive ed eseguire un ricalcolo saldi di magazzino.**
#0008	Win32	Il DBMS rileva la probabilita' di una situazione di stallo (deadlock) ed abortisce la transazione	Il fenomeno è sporadico e abbastanza raro ed avviene nonostante che SIGLA gestisca correttamente l'attività transazionale sul DBMS. La versione 1.60b è in grado di intercettare l'errore generato dal server e di rilanciare, dopo aver avvertito l'utente, lo statement SQL abortito. **Installare la versione 1.60b o successive.**
#0009	Win16/WFW	Usando la procedura da più posti di lavoro simultaneamente di possono generare fenomeni di lock.(Il problema è dipendente dall'attivazione delle transazioni e non si manifesta operando in modalita AUTOCOMMIT)	Usare driver ODBC a 32 bit (e conseguentemente la versione a 32 bit dell'applicazione)
#0012	Win16/Win32	SIGLA funziona correttamente solo su un un posto di lavoro	Poiche' SQL Server non consente la contemporanea esecuzione di più statements SQL sulla stessa connessione, SIGLA è costretto ad allocare fino a 8 (otto) connessioni da ogni workstation. E' necessario impostare quindi il numero delle connessioni attivabili su SQL Server ad un valore appropriato.
#0014	Win16/Win32	Viene generato il messaggio d'errore:'SQL Server has run out	aumentare il valore del parametro LOCKS sul DBMS

		of LOCKS'	
#0015	Win16/Win32	MS Query locca in modo esclusivo le tabelle convolte in una select fino a che tutti i records selezionati non vengono fetchati in memoria	Minimizzare la durata della query e leggere immediatamente tutti i records selezionati allo scopo di non bloccare l'attività degli altri utenti
#0017	Win16/Win32	S++ si comporta in modo anomalo generando informazioni incongruenti	Il DBMS è stato installato in modalita' 'case insensitive'. Rigenerare l'installazione selezionando la modalita' 'case sensitive'

DB2/400

Problem ID	Environment	Descrizione	Soluzione
#0011	Win16/Win32	Le stampe generate non contengono tutti i record previsti	Attivare la gestione dell'ordinamento EBCDIC tramite la procedura di configurazione.

INTERBASE

Problem ID	Environment	Descrizione	Soluzione
#0013	Win16/Win32	Durante la creazione delle tabelle nel database dati comuni viene segnalato un errore sulla tabella UTENTI	Installare la versione 1.60g o successiva