consip

Standard di programmazione

Giugno 2001

INDICE

Standard di programmazione

Gli standard di programmazione indicati devono essere adottati per tutti gli oggetti software di nuova realizzazione.

Quando non diversamente specificato, l'applicazione degli standard è riferita anche agli oggetti creati nell'ambito di evoluzioni di progetti esistenti. Non si richiede pertanto l'adeguamento massivo degli oggetti esistenti, fermo restando che, durante le attività di modifica a oggetti esistenti, vada valutata la possibilità di adeguamento alle nuove regole.

Le indicazioni fornite non sostituiscono la documentazione tecnica dei singoli prodotti citati, cui si rimanda per le informazioni di dettaglio.

1 COPIE E STAMPE DI ARCHIVI

Le seguenti indicazioni sono valide per JCL in ambiente MVS.

- **Archivi sequenziali (PS):**

  Possono essere copiati o stampati mediante l'utility IBM **DFSORT** (utilizzando OPTION COPY se non è necessario ordinare diversamente l'archivio). Si possono selezionare record e colonne mediante l'utility associata ICETOOL, si realizzano così più operazioni in uno stesso step.
  Sono disponibili anche altre utility adatte allo scopo (IEBDG, IEBGENER etc.) ma, per motivi prestazionali, si ritiene preferibile l'utilizzo di DFSORT (insieme a ICEGENER e ICETOOL).
  L'utility **DFSORT** è disponibile, per alcune sue funzionalità, in modalità on-line sul Menù generalizzato dei "Prodotti", opzione "I".

- **Archivi VSAM KSDS:**

  Possono essere copiati, stampati, scaricati e ricaricati, mediante la funzione REPRO dell' AMS oppure utilizzando la stessa utility **DFSORT** presa in considerazione per i dataset sequenziali. La funzione LISTCAT dell'AMS fornisce informazioni esaurienti su questo tipo di archivi.

- **Archivi Direct Access:**

  Le operazioni di copia di questi archivi possono essere effettuate tramite le funzioni presenti nel prodotto Compuware "File Aid for MVS", disponibile in modalità on-line sul Menù generalizzato dei "Prodotti", opzione "9.FM". Tali funzioni consentono la copia batch o interattiva fra archivi DA e fra archivi DA e PS: permettono inoltre di effettuare selezioni particolari che vengono specificate in una sessione interattiva del prodotto.
  Per quanto riguarda le informazioni sugli archivi di questa tipologia, il prodotto File Aid le fornisce interattivamente.

La seguente tabella presenta un quadro riepilogativo delle funzioni utilizzabili per i differenti tipi di archivi:

	Copia	Scarico su PS	Creazione con input da PS	Info
Archivi PS	DFSORT per copie su PS e KSDS (anche selettive)	Non previsto	Non previsto	Non previsto
Archivi KSDS	DFSORT o REPRO (AMS) Per copie su PS e KSDS (anche selettive) Necessità di inserire un doppio step.	DFSORT o REPRO (AMS)	DFSORT o REPRO (AMS) Il caricamento deve essere preceduto da uno step di IDCAMS con il comando DEFINE CLUSTER	Comando LISTCAT (AMS)
Archivi DA	File Aid for MVS per copie su PS e DA (le selezioni possono essere specificate mediante una sessione interattiva del prodotto)	File Aid for MVS (le selezioni possono essere specificate mediante una sessione interattiva del prodotto)	File Aid for MVS (le selezioni possono essere specificate mediante una sessione interattiva del prodotto)	Sessione interattiva di File Aid for MVS

L'applicazione dello standard descritto consente la sostituzione di codice sviluppato ad hoc, presente nei sistemi informativi gestiti da Consip.

1.1 HSELCOPY – NOTE PER GLI UTILIZZATORI DELL'UTILITY

Le funzioni sopra descritte consentono di sostituire quelle offerte dall'utility HSELCOPY, che forniva le seguenti funzionalità:

1. Copia di archivi organizzati come PS, IS, DA, e VSAM KSDS;
2. Stampa esadecimale e decimale di un archivio o di una sua porzione;
3. Scarico di un archivio IS, VSAM KSDS, DA o di loro porzioni su un dataset PS;
4. Creazione di un archivio IS o VSAM KSDS a partire da un sequenziale o da una porzione di esso;
5. Informazioni su archivi IS, DA, VSAM KSDS;
6. Perforazione di un archivio o di una sua porzione.

Non si sono fornite indicazioni su utility standard da utilizzare per archivi IS, in quanto questi non sono più presenti in ambiente PLT e in esercizio.

Non si sono fornite indicazioni su come effettuare la perforazione di un archivio o di una sua porzione, in quanto la perforazione non è più ritenuto requisito necessario.

La sostituzione del prodotto HSELCOPY con le utility standard indicate, comporta diverse modalità di scrittura di JCL e SYSIN, rispetto ai procedimenti adottati fino a questo momento: può essere richiesto, ad esempio, l'effettuazione di più step per operazioni realizzate precedentemente in un unico step.

2 STRUTTURAZIONE E PROGRAMMI IN COBOL

Relativamente alle modalità di strutturazione dei programmi Cobol, i programmi devono utilizzare esclusivamente le regole sintattiche del linguaggio Cobol (standard ANSI 85), secondo quanto previsto dalla documentazione di prodotto.

Per quanto riguarda i requisiti di qualità, per la strutturazione dei programmi, questi sono elencati nei piani di qualità e/o nei contratti che regolano i singoli progetti.

L'applicazione dello standard descritto consente la sostituzione di codice sviluppato ad hoc presente nei sistemi informativi gestiti da Consip.

2.1 WASP – NOTE PER GLI UTILIZZATORI DELLA MACRO

La macro WASP garantisce un livello minimo di strutturazione, obbligando però all'utilizzo di particolari procedure di compilazione e rendendo il codice sorgente di difficile comprensione e non di tipo standard.

Al fine di rendere più semplice l'applicazione dello standard sopra descritto, si inserisce un esempio di codifica WASP, a cui fa seguito l'output del traduttore dopo la compilazione.

Si riporta poi lo stesso blocco di istruzioni codificato seguendo istruzioni proprie del Cobol, da cui è evidente come le potenzialità del linguaggio, sfruttate opportunamente, rendono non solo più leggibile il sorgente ma permettono di essere indipendenti da qualunque tipo di personalizzazione.

□ Esempio WASP.

Il codice sorgente presenta le seguenti keyword, proprie di WASP: BEGIN NSQUARE, END NSQUARE, BLOCKS, END BLOCKS, REPEAT, WHILE, ENDREPEAT, DO, DO DUMMY, ENDIF:

```
C1-010-ELA-MSG.
        BEGIN NSQUARE
        BLOCKS TR-INI
                TR-ENT REP
                TR-TIP REP
                TR-CAT REP
                TR-COM REP
                TR-FIN
        END BLOCKS
        DO TR-INI
        REPEAT
                DO TR-COM
          IF TR-COM-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                 REPEAT
                        DO TR-ENT
                  WHILE TR-ENT-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                   ENDREPEAT
                IF TR-FIN-CC EQUAL 0
                    REPEAT
                        DO TR-TIP
                    WHILE TR-TIP-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                    ENDREPEAT
                ELSE
                        DO DUMMY
                ENDIF
                IF TR-FIN-CC EQUAL 0
                    REPEAT
                        DO TR-CAT
                    WHILE TR-CAT-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                    ENDREPEAT
                ELSE
                        DO DUMMY
                 ENDIF
           ELSE
                    DO DUMMY
           ENDIF
        WHILE TR-COM-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
        ENDREPEAT
        DO TR-FIN
        END NSQUARE
Cl-010-ELA-MSG-EX.
```

□ Esempio dell'output del traduttore:

Tale output è stato scaricato ad hoc per mostrare il tipo di decodifica che deve essere realizzata sul sorgente, generalmente non è preso in considerazione dal programmatore:

```
*******************************************************************************
** **
**                              BEGIN NSQUARE
** **
*******************************************************************************
C1-010-ELA-MSG.
        PERFORM TR-INI THRU TR-INI-EXIT
        IF CICS-FLG NOT = ZERO
                GO TO Cl-010-ELA-MSG-EX.
NSQUARE-LP-1.
        PERFORM TR-COM THRU TR-COM-EXIT
        IF CICS-FLG NOT = ZERO
                GO TO CL-010-ELA-MSG-EX.
        IF TR-COM-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                NEXT SENTENCE
        ELSE
                GO TO NSQUARE-IF-2.
NSQUARE-LP-3.
        PERFORM TR-ENT THRU TR-ENT-EXIT
        IF CICS-FLG NOT = ZERO
                GO TO Cl-010-ELA-MSG-EX.
        IF TR-ENT-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                NEXT SENTENCE
        ELSE
                GO TO NSQUARE-LX-3.
                GO TO NSQUARE-LP-3.
NSQUARE-LX-3.
        IF TR-FIN-CC EQUAL 0
                NEXT SENTENCE
        ELSE
                GO TO NSQUARE-IF-4.
NSQUARE-LP-5.
PERFORM TR-TIP THRU TR-TIP-EXIT
IF CICS-FLG NOT = ZERO
        GO TO Cl-010-ELA-MSG-EX.
IF TR-TIP-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
        NEXT SENTENCE
ELSE
        GO TO NSQUARE-LX-5.
GO TO NSQUARE-LP-5.




NSQUARE-LX-5.
                GO TO NSQUARE-IX-4.
        NSQUARE-IF-4.
```

```
        PERFORM MODULE-DUMMY THRU MODULE-DUMMY-EXIT.
NSQUARE-IX-4.
IF TR-FIN-CC EQUAL 0
                NEXT SENTENCE
        ELSE
                GO TO NSQUARE-IF-6.
NSQUARE-LP-7.
        PERFORM TR-CAT THRU TR-CAT-EXIT
        IF CICS-FLG NOT = ZERO
                GO TO Cl-01Ò-ELA-MSG-EX.
        IF TR-CAT-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                NEXT SENTENCE
        ELSE
                GO TO NSQUARE-LX-7.
                GO TO NSQUARE-LP-7.
NSQUARE-LX-7.
        GO TO NSQUARE-IX-6.
NSQUARE-IF-6.
        PERFORM MODULE-DUMMY THRU MODULE-DUMMY-EXIT.
NSQUARE-IX-6.
        GO TO NSQUARE-IX-2.
NSQUARE-IF-2.
        PERFORM MODULE-DUMMY THRU MODULE-DUMMY-EXIT.
NSQUARE-IX-2.
        IF TR-COM-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
                NEXT SENTENCE
        ELSE
                GO TO NSQUARE-LX-1.
        GO TO NSQUARE-LP-1.
NSQUARE-LX-1.
        PERFORM TR-FIN THRU TR-FIN-EXIT
        IF CICS-FLG NOT = ZERO
                GO TO Cl-010-ELA-MSG-EX.
        GO TO Cl-010-ELA-MSG-EX.
MODULE-DUMMY.
MODULE-DUMMY-EXIT.
EXIT.
```

```
*****************************************************************************************
** **
**                                        END NSQUARE
** **
*****************************************************************************************
Cl-010-ELA-MSG-EX.
```

□ Esempio conversione WASP – COBOL:
Si riporta un'ipotesi di traduzione in PROCEDURE DIVISION relativa al blocco principale di elaborazione. Gl statement WASP sono asteriscati, quelli COBOL , introdotti in loro sostituzione, sono in grassetto:

```
C1-010-ELA-MSG.
*--* BEGIN NSQUARE
*--* BLOCKS TR-INI
*--*       TR-ENT REP
*--*       TR-TIP REP
*--*       TR-CAT REP
*--*       TR-COM REP
*--*       TR-FIN
*--* END BLOCKS
*--* DO TR-INI
          PERFORM TR-INI THRU TR-INI-EX
*--* REPEAT
          PERFORM UNTIL NOT (TR-COM-CC EQUAL O AND TR-FIN-CC EQUAL O)
*--* DO TR-COM
                PERFORM TR-COM THRU TR-COM-EX
          IF TR-COM-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
*--* REPEAT
                    PERFORM UNTIL NOT (TR-ENT-CC EQUAL 0 AND
                              TR-FIN-CC EQUAL 0)
*--* DO TR-ENT
                         PERFORM TR-ENT THRU TR-ENT-EX
*--* WHILE TR-ENT-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
*--* ENDREPEAT
                    END-PERFORM
               IF TR-FIN-CC EQUAL 0
*--* REPEAT
```

```
                    PERFORM UNTIL NOT (TR-TIP-CC EQUAL 0 AND
                                       TR-FIN-CC EQUAL 0)
*--* DO TR-TIP
                        PERFORM TR-TIP THRU TR-TIP-EX
*--* WHILE TR-TIP-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
*--* ENDREPEAT
                    END-PERFORM
                ELSE
*--* DO DUMMY
                        CONTINUE
*--* ENDIF
                END-IF
                IF TR-FIN-CC EQUAL 0
*--* REPEAT
                    PERFORM UNTIL NOT (TR-CAT-CC EQUAL 0 AND
                                       TR-FIN-CC EQUAL 0)
*--* DO TR-CAT
                        PERFORM TR-CAT THRU TR-CAT-EX
*--* WHILE TR-CAT-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
*--* ENDREPEAT
                    END-PERFORM
                 ELSE
*--* DO DUMMY
                        CONTINUE
*--* ENDIF
                END-IF
            ELSE
*--* DO DUMMY
                    CONTINUE
*--* ENDIF
            END-IF
*--* WHILE TR-COM-CC EQUAL 0 AND TR-FIN-CC EQUAL 0
*--* ENDREPEAT
            END-PERFORM
*--* DO TR-FIN
            PERFORM TR-FIN THRU TR-FIN-EX.
*--* END NSQUARE
CI-010-ELA-MSG-EX.
```

3 COMMAND LIST O OGGETTI REXX

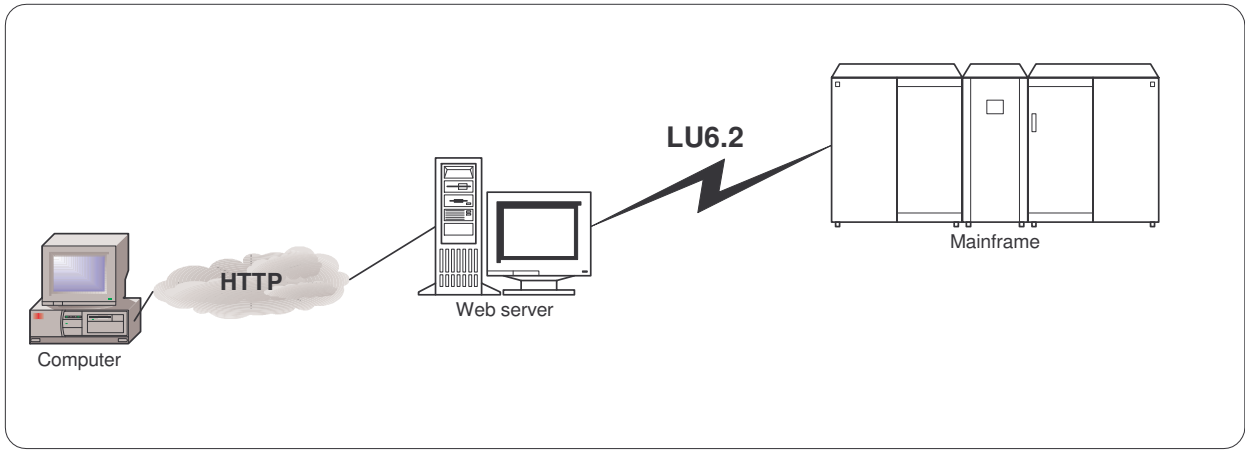
Le seguenti indicazioni sono valide in ambiente MVS.

Nell'ambito dello sviluppo di progetti applicativi, non è consentito realizzare oggetti utilizzando Command List o linguaggio Rexx. Non è previsto il rilascio e la successiva gestione in esercizio di oggetti creati da tali linguaggi di programmazione interpretati.

Per espletare le funzionalità generalmente realizzate da tali oggetti, si richiede di utilizzare esclusivamente programmi, JCL, utility di sistema o utility fornite dai prodotti eventualmente acquisiti.

4 ACCESSO A PROGRAMMI CICS MAINFRAME IN ARCHITETTURA CLIENT/SERVER

4.1 ARCHITETTURA



Nella figura precedente è riportato un tipico esempio di architettura a più livelli in cui l'applicazione è realizzata secondo protocolli di tipo client-server.

Ad un primo livello si pone un personal computer dotato di browser, che gestisce l'interfaccia con l'utente.

Ad un secondo livello è presente un server, su cui sono presenti un web server, un application server e un software di comunicazione con il mainframe.

Di seguito sia il primo livello, browser, che il secondo, web/application server, saranno identificati come un'unica sezione, "*parte presentation*".

Al terzo livello è presente la parte di logica elaborativa dell'applicazione e di accesso ai dati realizzata secondo tradizionali standard cobol/CICS/DB2 che risiedono sul mainframe.
Tale sezione sarà identificata come "*parte host*".

A prescindere dallo schema fisico presentato quale esempio, è rilevante il fatto che il colloquio tra la *parte presentation e la parte host* dell'applicazione è realizzato secondo il protocollo Distributed Program Link del CICS (DPL).

Il protocollo DPL consente ad un programma client, residente su *parte presentation*, di comunicare con un programma CICS, residente su *parte host* ("remoto"), utilizzando le stesse modalità di colloquio comunemente utilizzate per la chiamata tra programmi CICS su mainframe ("locali"); il programma CICS chiamato, quindi, non deve contenere istruzioni particolari per comunicare con il client, ma può essere realizzato come una comune routine di un'applicazione CICS tradizionale.

Come esempio si riporta la chiamata effettuata da un programma client java, che utilizza il metodo ECIRequest fornito dal CICS Transaction Gateway.

```
MyECI =new ECIRequest(ECIRequest.ECI_SYNC_TPN, SERVER, // CICS Server
                      USERID,                          // UserId, null for none
                      PASSWORD,                        // Password, null for
none
                      PRGNAME,                         // Program name
                      TRANSID,
                      abCommarea,                      // Commarea
                     abCommarea.length,
                      ECIRequest.ECI_NO_EXTEND,
                      ECIRequest.ECI_LUW_NEW);
```

La chiamata suddetta, dal punto di vista applicativo, è equivalente alla:

EXEC CICS LINK PROGRAM(program name)
COMMAREA(Commarea) **LENGTH**(Commarealength)

nel senso che il programma chiamato non sa di essere invocato via **ECIRequest** piuttosto che via **EXEC CICS LINK**.

Nel seguito si descrivono in dettaglio le caratteristiche della parte host dell'applicazione, ovvero della componente realizzata con programmi CICS residenti su mainframe, rimandando a documenti specifici la descrizione delle diverse modalità di sviluppo delle componenti client.

Di seguito si indicano:

- Modalità di comunicazione e passaggio di dati;
- Modalità da seguire nella stesura del sorgente;
- Standard di nomenclatura dei programmi mainframe;
- Le note per gli utilizzatori dei servizi e delle funzioni SGC, fornendo, ove possibile, indicazione di come procedere per ottenere gli stessi servizi.

4.2 MODALITÀ DI COMUNICAZIONE E PASSAGGIO DATI

Il passaggio di parametri e dati tra programmi client e programmi mainframe si realizza esclusivamente attraverso DFHCOMMAREA, utilizzando le funzioni proprie degli standard di comunicazione DPL.
Si consideri che il limite massimo della DFHCOMMAREA è pari a 32.000 bytes, ma per motivi prestazionali, è opportuno limitare il più possibile la dimensione di quest'area, evitando di definire strutture non completamente utilizzate per l'effettivo scambio di dati.
Sulla *parte host* possono essere codificate differenti routine: si definisce *prima routine* quella richiamata da client. Non esistono vincoli per il passaggio di dati su mainframe tra questa *prima routine* e le successive: questo si realizza con DFHCOMMAREA e/o utilizzando code di Temporary storage o altre funzionalità proprie del Cics (Getmain, etc).
La comunicazione DPL non prevede l'esistenza di terminali 3270, quindi non è possibile utilizzare come area di comunicazione la TCTUA.
Risultano inoltre inibite la gestione di mappe BMS e la comunicazione tramite APPC.

E' opportuno fare alcune precisazioni:

- I dati su mainframe sono in formato EBCDIC; i dati su client sono in formato ASCII: devono quindi essere convertiti opportunamente nel passaggio da un ambiente all'altro. Questa funzione di conversione è realizzata in modo automatico dal software di base presente nei sistemi; è però opportuno considerare l'impatto che questa conversione può generare nel trattamento dei dati.
- Il trattamento dei campi numerici deve avere una particolare gestione: quando si trattano valori negativi, decimali, etc., si devono utilizzare opportuni algoritmi di trascodifica.
  Si verificano diverse possibilità:
  - Campi numerici negativi: deve essere previsto, per ogni campo numerico negativo, un byte per il segno; questo deve essere impostato dal programma client. Il programma cobol ricevente deve prevedere un campo definito:
    Campo-num pic S9(n) SIGN IS LEADING SEPARATE CHARACTER.
  - Campi numerici decimali: un campo decimale può essere trasmesso dal programma client come una unica stringa costituita dalla parte intera e dalla parte decimale senza alcuna separazione. Il programma cobol ricevente deve prevedere un campo definito come segue:
    Campo-num pic 9(n)V9(m) se il campo è senza segno.
    Campo-num pic S9(n)V9(m) SIGN IS LEADING SEPARATE CHARACTER se il campo trasferito è con segno e virgola.
  - I campi Cobol COMP o COMP-3 non possono essere utilizzati come dati di passaggio, occorre definirli in formato DISPLAY e successivamente muoverli in campi COMP o COMP-3.

4.3 MODALITÀ DI STESURA DEL SORGENTE

I programmi realizzati su mainframe devono contenere esclusivamente istruzioni CICS native.
Le istruzioni utilizzabili sono soggette ad alcune restrizioni, dovute alla modalità di comunicazione DPL; in particolare sono inibiti i comandi relativi a:

- comunicazioni APPC;
- signon/signoff;
- comandi BMS;
- comandi di terminal control.

Per i dettagli si rimanda al manuale “CICS Application Programming Reference”.

I sorgenti devono essere routine di tipo CICS nativo.

4.3.1 DEFINIZIONE TRANSAZIONI CICS

La componente server dell’applicazione è composta da:

- una prima routine, che riceve i dati di input dal client e trasmette a questo i dati di output;
- eventuali routine secondarie, richiamate dalla precedente in cascata.

In questa struttura, la *prima routine* non è un vero e proprio main program, ma essendo la prima routine richiamata su mainframe, deve avere un transid associato.

4.4 STANDARD DI NOMENCLATURA

La prima routine, richiamata in modalità remota dal programma client, deve rispettare il seguente standard di nomenclatura: primi 4 caratteri = transid + 3 caratteri numerici.

Le successive routine mainframe richiamate saranno caratterizzate dai primi 4 caratteri uguali al transid e da un progressivo numerico successivo a quello attribuito alla prima routine.

4.5 SGC – NOTE PER GLI UTILIZZATORI DELLE FUNZIONI E DEI SERVIZI SGC

Nei programmi CICS nativi non devono essere utilizzati comandi SGC e funzioni/servizi SGC.

I comandi SGC si identificano con istruzione: **"EXEC SGC …".**
L'utilizzo di funzioni e servizi SGC può però essere nascosto all'interno di routine SGC generalizzate del tipo YLOG200 o CXP…, richiamate mediante comandi apparentemente non SGC come "CALL" o "EXEC CICS LINK…". Dette routine SGC richiamano servizi o contengono istruzioni SGC. Ciò significa che tali routine non devono essere richiamate e i servizi che esse offrono devono essere gestiti diversamente.

Si fornisce di seguito un elenco completo di tutte le funzioni ed i servizi SGC richiamabili.
I programmi realizzati su mainframe non devono contenere questo tipo di istruzioni: si indica, caso per caso, quali funzionalità devono essere gestite con altra modalità oppure di quali attività non serve proprio preoccuparsi.

□ **Gestione SGC Mappe BMS**

Servizio SGC non necessario a questo tipo di applicazioni.

I programmi mainframe realizzati e richiamati via DPL non possono gestire mappe; l'interfaccia grafica e le problematiche ad essa connesse sono gestite su client.

□ **Vettore di terminale e ufficio associato**

Servizio SGC non necessario a questo tipo di applicazioni.

La comunicazione DPL non prevede l'esistenza di terminali 3270, quindi non è possibile utilizzare l'area di memoria TCTUA, in cui SGC memorizza vettore di terminale e codice ufficio.

In questo tipo di applicazione l'unica informazione utile è l'identificazione dell'ufficio operante: tale informazione può essere reperita in modo alternativo con due operazioni:
- 1. Funzione CICS ASSIGN, per reperire la userid dell'utente;
- 2. Routine assembler "YUFF000", a cui si fornisce in input la userid e un tipo di richiesta..

Tale routine generalizzata accede all'archivio di RACF, dove è definita un'associazione logica tra utenti, identificati dalla userid , e gruppo/i logico/i (ufficio/i) di appartenenza.
In risposta si potrà ottenere, a seconda del tipo di richiesta formulato, un ufficio di default o un gruppo di uffici, per i quali l'utente risulta abilitato a connettersi.

□ **Gestione della riga di stato di una mappa BMS**

Servizio SGC non necessario a questo tipo di applicazioni.

I programmi mainframe non gestiscono mappe.

- **Gestione dell'immagine**

Servizio SGC non necessario a questo tipo di applicazioni.

I programmi mainframe non gestiscono mappe.

- **Controlli sui messaggi di input**

Servizio SGC non necessario a questo tipo di applicazioni

Tali controlli sono effettuati sui campi delle mappe BMS di input; le mappe non sono presenti in questo tipo di applicazioni. Tutti i controlli di questo tipo sono effettuati sul client.

- **Gestione stampanti**

Servizio SGC non necessario a questo tipo di applicazioni.

Le stampe, nell'ambito di questo tipo di applicazioni, vengono attivate e gestite su client.

- **Passaggio parametri**

Servizio SGC non necessario a questo tipo di applicazioni.

Il passaggio di parametri di cui si parla, è quello gestito dalla pre e post-elaborazione di SGC. Il passaggio di tali parametri viene gestito direttamente dai programmi secondo le modalità sopra descritte (par. 1.4.2).

- **Archivi e moduli tabella**

Servizio SGC non necessario a questo tipo di applicazioni.

Gli archivi e i moduli tabella sono strettamente legati all'utilizzo di comando SGC.

□ **Instradamento**

Servizio SGC non necessario a questo tipo di applicazioni.

L'interfaccia grafica e le problematiche ad essa connesse, vedi instradamento, sono gestite su client.

□ **Gestione vocabolario e dei diagnostici**

Servizio SGC da non utilizzare. La gestione dei diagnostici va realizzata con altra modalità.

La gestione dei possibili messaggi diagnostici viene effettuata da funzioni SGC riconoscibili dalle seguenti istruzioni:

- EXEC SGC VOC, con cui si acquisisce il testo dei diagnostici inseriti in un archivio centralizzato, identificato come "vocabolario";
- EXEC LINK PGM "CXPVODRV", espansione dell'istruzione precedente. CXPVODRV non è altro che la routine che legge l'archivio "vocabolario".

Nessuna delle istruzioni sopra descritte va utilizzata per la gestione dei possibili messaggi di errore.

I programmi CICS devono impostare esclusivamente il codice di ritorno, sulla base delle possibili situazioni di errore. Tale codice va poi decodificato su client tramite una routine/funzione generalizzata specifica dell'applicazione, questa provvede alla trascodifica dei codici ricevuti in messaggi di errore testuali, adeguati all'interfaccia grafica utilizzata.

□ **Gestione condizioni anomale**

Servizio SGC da non utilizzare. La gestione di condizioni anomale va realizzata con altra modalità.

Le condizioni anomale sono individuate e gestite da una macro SGC, attivata dall'istruzione EXEC SGC BEGIN. Inoltre in caso di errore logico, nei programmi CICS si prevede la scrittura di un archivio di logerr, attraverso l'istruzione EXEC LINK PGM 'YLOG200'. La routine YLOG200 contiene istruzioni SGC, non va quindi utilizzata in questo tipo di applicazioni.

In alternativa, i programmi mainframe CICS devono individuare e gestire le situazioni di errore applicativo, inserendo opportunamente funzioni di 'Handle Condition' per la gestione di situazioni

anomale e realizzando un test esplicito sul codice di ritorno (DFHRESP) di funzioni Cics richiamate. Tali istruzioni consentono di individuare e controllare tutti i possibili casi di errore ed impostare opportunamente un codice di ritorno decodificato poi su client o, se necessario, far terminare la transazione in abend con un opportuno codice.

Si valuti l'opportunità di definire e scrivere autonomamente un archivio, proprio dell'applicazione, che sostituisca l'archivio dei logerr e dove si tenga traccia di tutte le situazioni di errore di cui si vuole avere memoria.

□ **Gestione stato delle risorse**

Servizio SGC da non utilizzare. La gestione di possibili situazioni di risorse CICS non disponibili va realizzata con altra modalità.

L'interrogazione sullo stato delle risorse CICS, siano archivi, transazioni, programmi o altro, viene effettuata attraverso istruzioni SGC. Tale interrogazione viene realizzata prima di richiamare le risorse stesse: ad esempio prima di leggere un archivio viene sempre controllato che sia disponibile. Presupponendo che la indisponibilità delle risorse definite al CICS sia un evento eccezionale, l'interrogazione preventiva appare spesso inutile.

Il controllo della disponibilità delle risorse del CICS si realizza con istruzioni SGC:
- EXEC SGC INQY: per fare interrogazione diretta di risorse;
- EXEC LINK PGM "CXPMCSER" e 'CXPSRSER': programmi che effettuano interrogazioni sul CICS e poi forniscono risposta.

Nessuna delle istruzioni sopra descritte va utilizzata in questo tipo di applicazioni.

Il controllo non si effettua preventivamente, ma solo in seguito all'attivazione di un comando CICS: si testa il buon fine dell'istruzione CICS richiamata.. Tale procedimento prevede l'inserimento della funzione CICS di 'Handle Condition', ed il test puntuale della condizione di 'INVREQ' sulla risorsa richiesta, condizione generata dal comando stesso.

□ **Exit routine di Input, Output e Program Error Program**

Servizio SGC non necessario a questo tipo di applicazioni.

Si sottolinea che:
- la exit di input del TCP/ZCP, permette la gestione centralizzata dei tasti funzionali; tale exit è relativa ad emulazione 3270;
- la exit di output del TCP/ZCP, consente di intercettare i messaggi di errore del CICS e tradurli, prima che essi siano inviati a terminale; tale exit è relativa ad emulazione 3270;

- la exit di program error program, disabilita le transazioni in caso di Abend e scrive un archivio SGC di checkpoint

5 HELP ON LINE CON CORPO PRODOTTO W.O.L.

5.1 GENERALITÀ

Obiettivo del presente documento è quello di descrivere le modalità per la redazione dell'help on line utilizzando il prodotto WOL (Writer On Line) prodotto dalla società TMM e distribuito dalla TMM stessa e dalla società Sistemi Informativi.

Writer-On-Line ricade nella classe degli "Help-Authoring Tools" ossia quei prodotti che aiutano lo sviluppatore nella formazione dell'help dell'applicazione.

Questi prodotti sono basati prevalentemente su piattaforme di word-processing e costituiscono una classe di prodotti di nicchia le cui funzionalità si trovano spesso integrate nei tool di sviluppo di più elevata complessità.

Sono quindi strumenti di sviluppo i cui prodotti finali vengono poi integrati (in modo più o meno automatico) nell'ambiente run-time dell'applicazione utente (ad esempio per applicazioni a tre livelli basate su web il prodotto viene inserito in una directory del server Web referenziata dalle pagine dell'applicazione).

Gli elementi caratterizzanti tali strumenti sono essenzialmente:
- facilità d'uso e flessibilità della componente di sviluppo:
- varietà dei prodotti che possono essere originati da uno stesso "source file".

Oggi le applicazioni utilizzano prevalentemente i seguenti cinque formati di help:

- **WebHelp** sviluppato dalla eHelp Corporation è il formato più comune per siti web, intranet e extranet.;
- **HTML Help** è la versione Microsoft dell'help HTML e comprende capacità di Html Dinamico;
- **WinHelp** è l'help standard delle applicazioni Windows e nella versione per Windows 2000 comprende anche funzioni che erano proprie di HTML Help;
- **Oracle Help per Java** è lo standard Oracle per la scrittura di help per applicazioni Java;
- **JavaHelp** è lo standard Sun Microsystem per applet e applicazioni Java.

5.2 IL PRODOTTO WOL WRITER ON LINE.

Il prodotto Writer on Line si discosta in parte da quanto descritto nelle generalità poiché produce un help che non ricade nei formati suindicati in quanto costituito da oggetti HTLM e Java.

Infatti, in fase di generazione, insieme alle pagine HTML, vengono generati anche un applet Java e dei Java Script.

Questo applet, che ha dimensioni fisse (90 KB), contiene l'albero di navigazione, che si apre sulla base del nodo selezionato, un indice generale e un motore di ricerca testuale che permette di cercare qualunque parola contenuta nei testi che fanno parte dell'help on line.

Tali caratteristiche fanno si che l' help sia completamente funzionante (anche la ricerca per parola) con qualsiasi server Web (Microsoft Internet Information Server (IIS), Apache etc) imponendo il solo vincolo che il browser supporti JavaScript e HTML 4.

L'ambiente di sviluppo WOL è stand alone su personal computer windows.

Per completezza di documentazione va notato che ulteriori moduli del prodotto WOL permettono anche la produzione di manuali on line (in formato HTML) e cartacei (.DOC.), e la produzione di corsi CBT.

5.3 AMBITO DI APPLICABILITÀ

Alla luce delle caratteristiche di flessibilità, della semplicità d'uso per gli sviluppatori e del positivo utilizzo nell'ambito di applicazioni del S.I.R.G.S., il prodotto W.O.L. è lo strumento attualmente adottato per la produzione di help on line.

L'utilizzo di WOL per la produzione di help on line va valutato nell'ambito delle funzionalità che si vogliono offrire all'utente, essendo un prodotto che offre funzionalità non sofisticate, bilanciate peraltro dalla reale indipendenza dal browser utilizzato dati i requisiti minimi richiesti.

L'utilizzo del prodotto WOL per la produzione di help on line è rivolta ad applicazioni in architettura a 3 livelli con interfaccia web, sia in ambiente Unix che NT, indipendentemente dal linguaggio o strumento di sviluppo utilizzato per creare l'applicazione.

Lo sviluppo di help on line con WOL non si applica alle applicazioni in cui l'interfaccia utente è fornita da un prodotto programma (ad esempio Business Objects), né a siti web, intendendo come tali le "applicazioni" di navigazione e non, ad esempio, una applicazione "gestionale" (ad esempio di acquisto on line) pur acceduta da un sito web.

Per quanto riguarda gli aspetti prestazionali, è consigliabile che il servente applicativo (web server) sia in rete locale.

Va posta attenzione inoltre a non generare testi particolarmente lunghi (oltre una pagina). Nel caso è necessario adottare l'accorgimento di spezzare il testo, inviando pagine successive a richiesta.

5.4 CONTENUTI DELL'HELP ON LINE.

5.4.1 GENERALITÀ

Questa sezione rappresenta l'help dell'help.
A titolo esemplificativo e non esaustivo, devono essere descritti:

- la struttura dell'help e le modalità di navigazione
- le funzioni di navigazione presenti (avanti, indietro, home, etc.)
- come sono evidenziati i riferimenti ipertestuali (se presenti)

In questa sezione verranno riportate anche le informazioni per descrivere sinteticamente l'area o il gruppo di applicazioni oggetto dell' help on line

5.4.2 L'APPLICAZIONE "X"

Per ogni applicazione devono essere presenti le seguenti informazioni :

- descrizione generale;
- le funzioni dell'applicazione

Descrizione generale
Descrizione generale dell'applicazione, contesto amministrativo, uffici/utenti interessati.

Funzioni dell'applicazione
Descrizione generale delle funzionalità dell'applicazione.

5.4.3 LE FUNZIONI DELL'APPLICAZIONE.

Per le funzioni dell'applicazione dovrà essere realizzata una sezione per ogni singola funzione utente .

Va posta attenzione a definire con chiarezza cosa s'intende per funzione utente.
Se per funzione si intende "operazioni effettuate dalla stessa interfaccia grafica con un'unica sessione di input ed un'unica sessione di output", dovrà essere presente una sezione per ogni funzione e non per ogni maschera (fatto salvo che più funzioni innescate dalla stessa maschera possono essere contenute nella stessa pagina di help). Quindi, ad esempio, saranno

raccolte insieme tutte le informazioni su come effettuare un Inserimento, separatamente da quelle per effettuare una Modifica , anche se entrambe partono dagli stessi campi d'input in maschera utilizzando due bottoni diversi.

Viceversa: se le funzioni sono strettamente quelle "funzioni utente" , va specificato per ognuna di essa il percorso operativo, cioè se devono essere utilizzate una o più maschere per operare.

Nel caso che, per ottenere un certo obiettivo, debbano essere effettuate più operazioni, ovvero utilizzate più funzioni (ciò equivale a definire la funzione complessa), deve essere prevista una sezione "processo", premessa alle altre sezioni, che spieghi in quale sequenza devono essere utilizzate le funzioni specifiche e con che vincoli eventuali.

In dipendenza della scelta fatta, per ogni maschera o funzione dovrà essere realizzata una sezione, i cui contenuti sono:

- Funzione
- Oggetti
- Stampa
- Glossario

5.4.3.1 Funzione

Descrizione, in forma testuale, della funzione utente corrispondente alla maschera in esame.

Vanno indicate le eventuali azioni da intraprendere in caso di segnalazione di errore, se non già inserite all'interno della segnalazione di errore stessa.

Nel caso di funzioni particolarmente complesse si può rappresentare, in questa sezione, uno schema grafico che illustri la sequenza delle singole funzioni elementari.

5.4.3.2 Oggetti

Elenco degli elementi presenti nella maschera, nelle tipologie di :

- bottoni/pulsanti
- campi della mappa.

Per ogni elemento, sia di input che di output, sarà riportata la descrizione del campo e/o le funzioni attivabili dai singoli bottoni/pulsanti.

La descrizione del singolo campo di input dovrà essere corredata da :

- valori ammessi e/o formato, a meno che i valori siano inseribili solamente da liste di scelta
- obbligatorietà

- lunghezza massima
- referenzialità logica tra campi (cioè se un campo è vincolato o utilizzabile solo in base all'immissione di dati in un altro campo), in funzione dell'utilità di tale indicazione per l'utente.

5.4.3.3 Stampe

Sono riportati esempi delle eventuali stampe prodotte localmente dalla funzione utente in esame, e, se non riportate tutte le stampe, l'elenco delle informazioni ottenibili dalle stampe.

5.4.3.4 Glossario

Definizione di tutti i termini e gli acronimi, di uso non corrente, necessari alla corretta e completa comprensione della parte testuale.

Il glossario, anche se richiamabile da ogni singola funzione, sarà definito come un elemento unico nei contenuti per tutta l'applicazione

6 EBCDIC – CHARACTER SET

Si richiede di utilizzare, nello sviluppo di applicazioni su OS/390, il set di 81 caratteri del code page 640 chiamato "syntactic character set" (vedi manuale: "ESA/390 Principles of Operation" appendice "I"). Solo tale set di caratteri fornisce garanzia sull'univocità di trascodifica dei caratteri, indipendentemente dal sistema di rappresentazione EBCDIC utilizzato.
Tale regola è necessaria in quanto sulle postazioni utente presenti in periferia, possono essere istallati programmi di emulazione diversi e utilizzati code page differenti. A titolo di esempio si è verificato che il carattere "\" su una mappa Cics veniva visualizzato in modo errato utilizzando emulatori 3270 EICON anziché Personal Communication.

7 VALORI ASSOLUTI NELLE APPLICAZIONI

Qualunque sia il tipo di applicazione sviluppata e indipendentemente dal linguaggio di programmazione utilizzato si richiede di non fare uso di variabili assolute, con le quali si inseriscano in modo chiaro ed esplicito i valori specifici, ad esempio, dell'ambiente in cui si sta codificando (indirizzi IP, porte, driver, path assoluti), oppure della userid e password di connessione. Si richiede, in altre parole, che il codice non abbia al suo interno valori assoluti e che, dove necessario, si utilizzino opportuni criteri di parametrizzazione che garantiscano la portabilità del software realizzato.