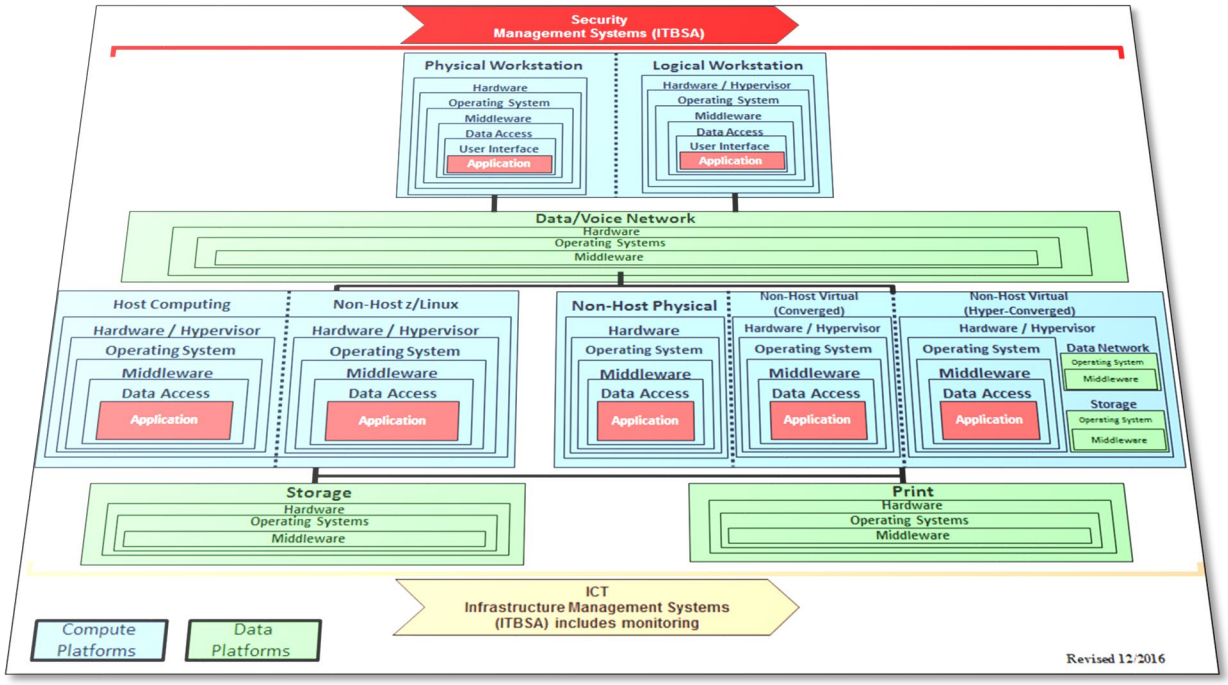
South Carolina

BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association

imagination to Implementation i2

INFORMATION TECHNOLOGY

# Technical Standards — Applications




# Information Systems Standards Manual

# Table of Contents

# Table of Figures

# List of Tables

# Chapter 1 Application Coding

This chapter contains standards that are applicable to the application systems coding at BlueCross BlueShield of South Carolina.

# 1.1 Online Design Considerations

Online program size should be kept to the minimum necessary to meet functional business requirements.

Modular programming techniques used in batch programming should not be used for online programs. Related program logic should be kept together, with the processing logic starting from the beginning of the program and moving down.

# 1.2 AMB/COBOL

## 1.2.1 Programming Guidelines

### 1.2.1.1 Batch Coding Guidelines

When designing and programming Procedure Division code to be used by both batch and online, the units of common code should be of paramount concern.

The Procedure Division common code should be organized in logical units, broken only by the need for the performance of I/O services. COMMAREAs should be developed in the work and file description areas so that the master module may pass processing data and/or return code indicators to the next logical unit of included code.

#### Maintenance

Once the included common code is in use in the online environment, maintenance procedures should be as follows:

##### Files

The Database Administration (DBA) group must be given advance notice of any changes in the logical record length or the block size of any file used online. Normally five workdays will be satisfactory if no program modifications are required. The manager of the DBA group will schedule changes involving programs as well as files.

##### Common Included Modules

Work Areas
Changes that affect the length of a work area must be brought to the attention of the application online group prior to the compilation of the modules for validating. Changes affecting existing data and/or data elements need not be enumerated unless the online portion of the executable module must be changed to accommodate the new information.

Procedure Division
Changes in these modules need not be specified unless the online portion of the executable module must also be changed.

#### Use of Display in Batch Programs

The "DISPLAY" verb will not be used in Production programs except as specified below.

For all new program development and for existing programs as they are modified; the "DISPLAY" verb may be used to provide information about the following:

1. Start/end of the program (standalone and composite drivers only).
2. ABEND conditions.
3. Significant summary statistics of data used by the program.

### Start/End Program Displays

Each stand-alone or driver program in a composite may use DISPLAY to indicate the program has started and/or is at the end of the program. An example follows:

```
* PROGRAM XXXXXNNN STARTS *

or

* PROGRAM XXXXXNNN ENDS *
```

These displays are most beneficial when separation of SYSOUT outputs from different job steps is desirable. Called programs may NOT use DISPLAY in this manner.

### Display ABEND Conditions

Refer to *Technical Standards — Applications > Application Coding > AMB/COBOL > Programming Guidelines* for additional information.

### Significant Summary Statistics

Each program with significant functions (e.g. read, write, add, delete, or update functions) may have code to accumulate record counts and to display those counts in summary form before ending the program.

Summary statistics might include, but not be limited to, the following:

- Number of records read from input file(s)
- Number of records written to output file(s)
- Number of records processed in significant functions (e.g., bypassed, invalid, created)
- Any other counts needed to establish functional relationships.
- Where equalities are desired, equations may be displayed to show the relationships.

## Special Considerations

In order to improve program maintainability and execution, whenever possible, list all called modules in the function box at the beginning of the program. In addition, indicate whether the modules are statically or dynamically linked and whether they are compiled as DATA 24 or DATA 31. Dynamic calls and DATA 31 are the standard.

The following applies to all new development and any system work effort that is undergoing a major revision.

For both "batch" and "online," do not include any hard coded logic within an application program that examines and makes decisions based upon the layout and content of a RACF UserID, OPID, and/or TermID. If there is a need to obtain the identity and security authorization of the user or the organization that the user belongs to, refer to *Security Management > Information Security Management > Windows Server Security > Technical Class* for the prescribed methods to systematically retrieve the identity and security authorization of the user represented by the ID.

In AMB, if "your own" error handling logic is to be used in lieu of the AMB Error Handling logic, the valid parameter must be used. The valid AMB Error Handling parameters are:

```
For DB2:      SYM1 % &D2-AUTO-ERROR-HANDLING = 'OFF'

For IMS:      SYM1 % &IM-AUTO-ERROR-HANDLING = 'OFF'

For VSAM:     SYM1 % &VS-AUTO-ERROR-HANDLING = 'OFF'
```

**NOTE** If the SYM1 is left off the parameter, it will be treated as comments by AMB and the AMB default error handling logic will still be used. The default error handling logic attempts to send a message to a terminal. Therefore, in the case of a transaction listener, MQ process, or any transaction not terminal invoked, an ABM3 ABEND will occur.

## 1.2.1.2 Standard ABEND Codes

The purpose of this section is to define standards for user ABEND codes used by in-house application programs.

### When to Employ User ABENDS

User ABENDS should be used to alert operations and support staff that an error has occurred that prevents the application program from being able to process. This would include situations such as being unable to access DB2.

User ABENDS should not be used for application data errors that can be reported, written to an error file, or otherwise bypassed.

To ensure the usefulness and validity of the user ABEND code (refer to the subsection *Valid User ABEND Codes* below), no new user ABENDS may be added to a new or existing in-house application program without approval by the appropriate Systems Support area.

### Valid User ABEND Codes

User defined ABEND codes must be within the range of U3500 through U4093 with the exception of number U3610 that is restricted for use by IBM. Vendor assigned ABEND codes are outside BlueCross control and are not affected by this standard.

### Batch Display of ABEND Conditions

For all new program development and for existing programs as they are modified, the DISPLAY verb will be used to provide the following types of information about any programmed ABEND:

- Program ID
- Paragraph Number
- Abend Code
- Short Description of Error Condition
- Key of Record Being Processed
- And Any Fields Relevant to the Abend

Additional data may be displayed if it is deemed necessary for the resolution of the ABEND.

## 1.2.1.3 Restart Considerations

All programs or sorts having an estimated or actual run time of greater than 35 minutes will have explicit restart/recovery procedures either invoked through standard software or through special programming. Restart procedures are to be documented in job documentation (Refer to *Technical Standards — Infrastructure > Operating Systems > Production Control Log* for additional information.).

## 1.2.1.4 DB2 Programming Considerations

This section will explain general considerations for using Database2 (DB2) and provide a checklist for DB2 programming.

For all AMB programs, Database Administration staff will require a listing that details which tables are to be accessed by each program in the system. It is usually most efficient for the Software Designer or Project Manager to supply this information for the entire project, rather than on a one-by-one basis.

From the above information, Database Administration will build a DB2 subschema. This is necessary before the Software Developer can compile, bind or execute the application.

When coding column comparisons, make sure columns have similar definitions (e.g., both are numeric). Preferably, both should share the same format.

CICS applications that access DB2 must supply Resource Control Table (RCT) information to Technical Support.

The SQL return code should be checked after each database operation (e.g., open cursor, fetch, update, select, etc.). If there is an abnormal condition, it should be displayed on the terminal or report.

Always check null indicator variables before accessing null able data elements.

Certain types of DB2 data (e.g., date columns) are stored in a different physical form than they are presented to the user. The designer should not be concerned about how DB2 stores this data, but it does require that the designer be aware of the existence of this data.

## DB2 Programming Checklist

- Test queries using QMF, DB2 Optime, or another approved tool that can access Host DB2 for zOS prior to embedding them in application code.
- Use EXPLAIN to identify and evaluate the access path chosen by DB2 for efficiency. Check for tablespace scans, non-matching index columns, etc.
- Qualify WHERE clauses using an index. This encourages index reads versus table space scans.
- Select only the columns and rows needed. Physical data retrieval is by column not row. Let the SQL eliminate the rows and columns rather than application code.
- Fully qualify update and delete statements. This prevents changing rows incorrectly.
- When updating, update the data portion of the row only. Do not attempt to update the primary key of a table.
- Avoid duplicate code. Place redundant code in a paragraph and perform it. For example, if the program inserts rows in the same table from five points in the program, code one insert statement and perform it from each insert point.
- Avoid ordering on columns selected with an equal predicate. The value will be constant and sorting is generally not necessary. However, it may be necessary to order with the constant value to influence the use of an index.
- Use Host variables from the DCLGEN where possible. If using Host variables not in the DCLGEN, define the Host variables like the DCLGEN description. Avoid forcing DB2 to convert a comp Host variable to a comp-3 value.
- Avoid scalar functions such as SUBSTR.
- Always check null indicator variables.
- Always check the SQLCODE after each SQL statement executed.
- Do not hardcode the Collection ID. Use one of the methods described in *Systems Architecture > Concepts & Techniques > Host Platform Techniques > DB2 PLANBINDs and PACKBINDs.*

## Call Attach

The usage of Call Attach is preferred over the use of TSO Attach (IKJEFT01). Call Attach does not incur TSO overhead, and SMF execution statistics will be recorded under the application program name, instead of TSO.

The purpose of Database Attach is to allow batch programs that access DB2 to execute natively, rather than under batch TSO. Database Attach utilizes the Call Attach facility of DB2 to communicate with DB2. No source program changes are required to use Database Attach. Link edits, changes, and minor JCL changes are all that are required to utilize database Call Attach.

One and only one attachment method can be used by a specific instance or execution of an application to establish a connection to DB2. In other words, a batch job step can only have one attachment method.

The attachment method is set by the combination of NDVR type and processor group. If a program is to be linked to from online and batch, making it XAPSDB2 creates two attachment methods by creating two executables, one with CICS Attach and another for batch. Traditionally, the programmer had to choose which batch attachment type to use. This led to multiple copies of programs because one version was called from a DB attach load module and the other from an IMS load module.

To eliminate the need for two versions of the program in this scenario, Generic Attach was developed by BlueCross BlueShield of South Carolina. With Generic Attach, the attachment method is determined at each execution, not at compile/link time. Instead, the JOBLIB or STEPLIB concatenation in the execution JCL controls the attachment method. In one job or job step, a Generic Attach program can be run under IMS; in another, under DB attach. All that Generic Attach requires is a different processor group and setting the JCL properly.

## When Not to use Generic Attach

Load modules that update DB2, IMS, and MQ need coordinated/two-phased commits. **If a program is under IMS and DB2 updates are made via DB Attach, IMS doesn't know about them — this must not happen.** To prevent this, the driver program for load modules that update DB2 and IMS must be BAPSIDB2 or XAPSIDB2 and run under IMS (not generic). If there is no DB2 in the driver program, a simple SQL set current date statement should be added to pull DB2 in. This establishes the attachment method so that no subsequently called DB2 module will be able to use a different attachment method.

## DB2 and DB2/IMS Processor Groups

Only these processor groups (Table 1-1) should be used for new programs. Existing programs should be changed over to these. It is possible to have a program run under IMS but attach to DB2 with DB Attach, but this is not recommended. Including DB2 in IMS's two-phase commit feature is less error prone and easier to use than handling DB Attach commitment and recovery separately.

**DB2 and DB2/IMS Processor Groups**

| Type | Processor Group | DB2 Attachment Method | Description |
|---|---|---|---|
| BAPSDB2 | COBNY202 | IMS | EXEC |
| BAPSDB2 | COBNY208 | DB ATTACH | EXEC |
| BAPSDB2 | COBNY404 | GENERIC ATTACH | EXEC |
| XAPSDB2 | COBNY242 | IMS | EXEC/CICS |
| XAPSDB2 | COBNY248 | DB ATTACH | EXEC/CICS |
| XAPSDB2 | COBNY444 | GENERIC ATTACH | EXEC/CICS |
| BAPSIDB2 | COBNY208 | IMS | EXEC |
| BAPSIDB2 | COBNY203 | IMS | LOAD (LINKDECK) |
| BAPSIDB2 | COBNY210 | DB ATTACH | EXEC |
| BAPSIDB2 | COBNY404 | GENERIC ATTACH | EXEC |

**DB2 and DB2/IMS Processor Groups**

| Type | Processor Group | DB2 Attachment Method | Description |
|---|---|---|---|
| XAPSIDB2 | COBNY248 | IMS | EXEC/CICS |
| XAPSIDB2 | COBNY444 | GENERIC ATTACH | EXEC/CICS |

**Table 1-1** DB2 and DB2/IMS Process Groups

## Execution JCL Requirements for Call Attach

The JOBLIB concatenation is dependent upon the Call Attach used.

### Concatenation for IMS Attach

```
//JOBLIB  DD  DSN=BC.NDVR.PROD1.EXECLIB
//        INCLUDE MEMBER=IMSPBLNK
//        INCLUDE MEMBER=DB2OILIB
//* MQ libraries always go last
```

### Concatenation for DB Attach

```
//JOBLIB  DD  DSN=BC.NDVR.PROD1.EXECLIB
//        INCLUDE MEMBER=DB2ODBAT
//        INCLUDE MEMBER=DB2OILIB
//        INCLUDE MEMBER=IMSPBLNK
//* MQ libraries always go last
```

### Non-IMS Example

```
//FIPSS010 EXEC PGM=FIPSD32A       ** PROGRAM NAME
//SYSTSIN  DD  *
   DSN SYSTEM(DB2T)                ** DB2 SUBSYSTEM
   RUN PLAN(FIPSD32A)              ** PLAN TO EXECUTE
   END
//SYSTSPRT  DD  SYSOUT=*           ** DB2 MESSAGES
```

**A STEPLIB must be used if different attachment methods are used in the same job.

**When trying to run DB Attach under Cope, a STEPLIB is required because Cope will generate a STEPLIB that will undo the concatenation you established in your JOBLIB.

There cannot be any spaces between SYSTEM and (DB2T). There cannot be any spaces between PLAN and (FIPSD32A).

Production SYSTSIN input must be in BC.NDVR.PROD1.CARDLIB.

Concatenating the CDMS DB2 region CARDLIB with the plan-specific CARDLIB eliminates the need for multiple plan-specific CARDLIBs for the various DB2 subsystems.

```
//SYSTSIN DD DSN=BC.NDVR.PROD1.CARDLIB(CDMSDB2T),DISP-SHR
//        DD DSN=BC.NDVR.PROD1.CARDLIB(FIPSD32A),DISP=SHR
```

The DB2 subsystem can be identified by BC.NDVR.PROD1.CARDLIB(CDMSxxxx) where xxxx = the DB2 subsystem.

For example, BC.NDVR.PROD1.CARDLIB(CDMSDB2T) contains DSN SYSTEM(DB2T).

The plan to be executed is identified by a CARDLIB member indicating the plan to be run.

For example, BC.NDVR.PROD1.CARDLIB(FIPSD32A) would contain

```
RUN PLAN(FIPSD32A)

END
```

## IMS in BMP Mode

```
//AMMSS087        EXEC PGM=DFSRRC00
//               COND=(087,LT,AMMSS000),
//               PARM=(BMP,MM27D006,MM27D006,,,N00000,,,,,,,,IMP1,
                     ___________|________|__________________IMS ID--|
                                |        |
                                |        |----PSB NAME
                                |----PROGRAM NAME
//               AMMSPROD,,,,'',)
                 ___|--Database
//*
//               INCLUDE MEMBER=IMSPBMP
//*
//***            INCLUDE FOR PROD BMP BCBSSC (IMSPBMP)
//*
//PROCLIB DD     DSN=SYS3.IMSP.PROCLIB,DISP=SHR
//*
//DFSESL  DD     DSN=SYS2.IMSVS.RESLIB,DISP=SHR
//        DD     DSN=SYS2.DB2.DB2O.EXCP.DSNLOAD,DISP=SHR
//        DD     DSN=SYS3.MQSERIES.SCSQAUTH,DISP=SHR
//        DD     DSN=SYS3.MQSERIES.SCSQANLE,DISP=SHR
//*
//IMS     DD     DSN=IMSP.CDMS.PSBLIB,DISP=SHR
//        DD     DSN=IMSP.CDMS.DBDLIB,DISP=SHR
//*
```

## Binds and Plan Execution Authorization

DB2 Packages are created and bound for each move or generate performed in Endevor for a DB2 element.

DB2 Plans group packages into executable units. PLANBIND elements only need to be modified when new packages are added to the PKLIST.

There are 2 types of DB2 Plans — Batch and Online. All Batch plans will use DATCNTL as the OWNER and all online plans will use CICSAPP.

When a DB2 PLANBIND element is moved or generated in Endevor, the plan is bound to the DB2 subsystem defined by the BINDCARD, with the following exception. Online plans that are moved to production using normal packages are bound at 6:00 a.m. Friday mornings. Online plans that move in # (pound sign) packages require DBA assistance to be bound.

All Online plans will be executable in any CICS Region. Batch plans are only executable by defined user IDs. RACF controls the user IDs that can execute a plan. RACF profiles must be created before a batch plan can be executed.

There are 2 types of Batch Plan RACF Profiles — Discrete and Generic. Generic profiles use the wildcard characters of * and % and the Discrete profiles do not.

Most of the RACF plan execution profiles have been created using a four (4) character generic scheme. Plans that begin with the same four (4) characters are executable by the IDs defined by the profile. Individual authorization for each plan is not required. When a new batch plan is introduced into an existing application, it will generally not require any change to the authorization scheme.

For example:

> A RACF Profile of DB0P.ABCD*.EXECUTE -- ABCDAPP – READ:
>
> Will allow all plans that begin with ABCD to be executed by any job submitted using the USER=ABCDAPP.
>
> Access to ABCDWXYZ does not have to be explicitly granted when the USER=ABCDAPP will be executing the plan.

When it is determined that an individual plan is to be executed by another USER, a new RACF profile must be established for that plan when the generic plan cannot be modified.

A RACF profile must be in place before it is executed. All updates to RACF require that a TSC Ticket be created. Managers will need to submit the "DB2 PLAN USER ACCESS" form. The form is located under "Security/Access Services" in TSC Self-Service.

# 1.2.1.5 AMODE, RMODE, DATA Compile/Link Parameter Considerations

## Definitions

AMODE specifies the addressing mode of the load module.

- AMODE(24) indicates that 24-bit addressing must be in effect.
- AMODE(31) indicates that 31-bit addressing must be in effect.

RMODE specifies where the program executes above or below the line.

- RMODE(24) indicates that the module executes below the 16 MB virtual storage line (within 24-bit addressable virtual storage).
- RMODE(ANY) indicates that the module may execute anywhere in virtual storage either above or below the 16 MB virtual storage line. The module will execute above the 16MB virtual storage line when sufficient contiguous storage is available or below the 16MB virtual storage line when not.

DATA option specifies the location of dynamically allocated storage.

- DATA(24) allocates storage below the 16 MB virtual storage line.
- DATA(31) allocates storage above the 16 MB virtual storage line when sufficient contiguous storage is available or below the 16MB virtual storage line when not.

### Selection of AMODE, RMODE, and DATA Option

The Endevor Processor Groups determine the AMODE, RMODE, and DATA option combinations. All new programs should use an appropriate Endevor Processor Group for AMODE(31), RMODE(ANY), and DATA(31). There is no requirement to change all programs to an above-the-line processor group. When a program is modified, AMODE (31), RMODE(ANY), and DATA(31) must be used. The conversion of a program to an above-the-line processor group should be made when maintenance is applied to an existing program.

There are several exceptions that differ between new code and existing code:

- New code should always be AMODE(31), RMODE(ANY), and DATA(31), except when interfacing with vendor software that requires another combination.
- Existing code should always use an AMODE(31) processor group.
- All calling and called program AMODE and DATA combinations are valid except an AMODE(31) or DATA(31) program passing data to an AMODE(24) program.
- An IT Business Systems Help form should be filed if problems are encountered with processor group changes.

## 1.2.1.6 AppMaster Builder Application Standards

The purpose of this section is to define the standards for maintenance and the support of AppMaster Builder (AMB) applications.

The AMB application defines the environment used to create one or more AMB programs. The application lists programs, the targeted processing environment, and all their related AMB components and properties. The following defines the proper usage of an AMB application.

### Each Application should specify the correct properties

- If the application is MVS, the COBOL project name property must by MVSProj. If the application is Customer Information Control System (CICS), the COBOL project name property must be CICSProj.
- Valid User Interface properties are MVS and CICS.
- MVS (batch) and CICS (online) programs must be kept in separate applications.
- The following properties should all be selected for every application:

    - IMS
    - SQL Option for DB2
    - VSA
    - Floating point
    - Indexed files

- MOD size should be MOD2 unless explicitly specified otherwise in the program design specifications.

## Each AMB program must be listed in an Application

Every AMB program must be included on at least one AMB application prior to checking it into Endevor.

Any AMB program generated for more than one user interface (MVS & CICS) must be listed in a separate application, one for each user interface type.

An AMB program should only be listed in as many applications as there are unique target environments/execution libraries for program execution.

- A program must be listed only once within an application.
- MVS (batch) programs must have the Batch property selected for that program.
- AMB Stubs must have the Stub property selected for that member.
- Online Express programs must have the Online Express property selected for that member.

## All AMB components used by an AMB program must be listed in the program's Application

AMB components include the following entities:

- Stubs
- Screens
- Reports
- Data Structures
- Data Views (DDISYMB)
- User Rules (USERMACS)

### All Components

Any entity listed before the first program name is considered global and is inserted in all programs in the application. Valid global entities are screens, Data Structures, macro library members, and subschemas. Different types of global entities can be entered on the same line, each within their own column.

- To associate an entity with a few, but not all, programs, specify the entity as local to each desired program.
- List entities local to a single program on the same line as the program name, and on any line following the program name but preceding the next program name.
- Do not add unnecessary components to an application.

## Stubs

A program stub is a segment of Procedure Division code created in the Program Painter and is capable of being shared by different programs in your application.

- If a stub is used by any program defined in an application, the stub must be listed in the application. Use the Add function to add a Stub to the application.
- Stubs cannot have entities associated with them. For example screens, reports, etc., cannot be defined on the same line as a stub.
- Stubs are not included automatically by inclusion in the application. To place a program stub in a program, use the STUB keyword within the program.
- Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > AMB Naming Conventions* for AMB Stub naming standards.

## Screens

AMB screens contain the information necessary to create Basic Mapping Support (BMS) Maps and the associated program logic.

- For online programs, add each screen component to the program that uses it in the Application View.
- Only edit an AMB screen with the AMB Screen Painter. Never directly edit the associated SCRNSYMB member.
- If an AMB screen is to be replicated, use the COPY + ADD operations within AMB Application View to clone a screen under a new name. Do not use TSO functions to replicate or rename.
- CICS screen names cannot exceed seven characters in length. Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > AMB Naming Conventions* for screen/map naming standards.
- Specify whether the screen is input-only (I), output-only (O), or input/output (IO) in the screen member properties.
- To redefine a screen, create a user rule to redefine your screen record description in the macro library. The macro must be named $screenname-RECORD-RDF and the USERMACS member that contains it must be listed in the application. The macro doing the screen redefine should be added as a component to the program that will use it.

## Reports

Report layouts associated with AMB Report Writer programs are called Report Mock-ups.

- All report mock-ups used by a program should be listed within the application. Specify a program's report mock-up by adding it to the program within the AMB Application View. Multiple report mock-ups are permitted.
- Report Mock-ups are not included automatically by indicating the Report Mock-up name in the application. To place a Report Mock-up in a program, use the MOCK keyword within the program.

- Only edit an AMB report with the AMB Report Painter.
- Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > AMB Naming Conventions* for Mockup naming standards.

## Data Structures

AMB allows you to code reusable Data Structures for programs and copy libraries, using a shorthand format in the Data Structure Painter. The shorthand format substitutes indentation for level numbers and allows shorthand picture formats.

- Add a data structure (or multiples) as a component to a specific program in the AMB Application View.
- Data Structures may be added as components to the Globals folder in the AMB Application View.
- Specify where the Data Structure is to be placed within the program in the properties for the Data Structure. Valid types are Working-Storage Section , Linkage Section , or COMMAREA.
- Allow addressability to a Data Structure by including it within the application. Do not use the % INCLUDE to include Data Structures within the program.
- Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > AMB Naming Conventions* for additional information.

## Data Views (DDISYMB)

AMB stores information required for program access to databases and VSAM files in data views, also called DDISYMBs.

- Add a data view to a program within the AMB Application View. Do not use the DB-SUBSCHEMA command to bring a data view into a program.
- Data views may be added to the Globals folder of the AMB application.
- A program cannot include more than one subschema.
- Never directly edit a subschema/DDISYMB member. They should only be created through the AMB Data Definition Interface (DDI) utilities.
- It is the responsibility of the application programming areas to define and import VSAM file information into the subschema. DB2 & IMS database information will be defined and imported by Database Administration.

## User Rules (USERMACS)

AMB allows the construction of common user rules for use in AMB programs. These user rules are stored in a user rule library (USERMACS).

- User rules can be added to the Global folder of an AMB application. These rules are accessible to all programs within the application.
- User rules can be added to individual programs within an AMB application. These rules are accessible only to that program.
- Do not use the % INCLUDE to include USERMACS within the program.
- Update the properties of the user rule to specify the location it should be added to in the program. **Top of Program** is the location most used.
- Redefinition of the TP-USER-AREA is done by specifying CA (Top of COMMAREA) for the location. The macro member should generate the following line as its first statement, starting in column 12:

- 05 'dataname' REDEFINES TP-USER-AREA

### It is the Responsibility of the Project Team Archiving an AMB Program to Remove It from the all AMB Application(s)

Related components that are not used by any other program should also be removed from the application(s) and archived. If the last program in an application is archived, the application should be archived.

## 1.2.1.7 AMB ABEND Routine Standards

The purpose of this section is to define the proper code to be used to ABEND any AMB module.

The use of the AMB MACRO **BC-UNV-ABEND** is the required method of abending an AMB program:

```
$BC-UNV-ABEND WS-ABEND-CD

The BC-UNV-ABEND MACRO Generates the following AMB code:

% IF &AP-GEN-DC-TARGET = 'MVS'

   WS-ABEND-CODE = &ABCODE

   CALL WS-TS00L100 USING WS-ABEND-CODE

% IF &AP-GEN-DC-TARGET = 'CICS'

   WS-ABEND-CODE-X = <% &Q&ABCODE&Q>

   $TP-BACKOUT( "ABORT(WS-ABEND-CODE-X)" )
```

## 1.2.1.8 VSAM Coding

**NOTE** COBOL ONLY

Example of file status:

```
004100                                  INPUT-OUTPUT SECTION.

004110                                  FILE-CONTROL.

004100              SELECT              RAN-CLAIM-FILE ASSIGN
                                        TO DD1
```

```
004130                                          ORGANIZATION INDEXED

004140                                          ACCESS IS RANDOM

004150                                          FILE STATUS IS STATUS-
                                                BYTES

004160                                          RECORD KEY IS RAN-
                                                CLAIM-KEY.

                                                etc...

006980...                                       WORKING-STORAGE
                                                SECTION. etc

07553**                                         *****

007554****          VASM  STATUS BYTES         *****

07555**                                         ****

007556      01      STATUS-BYTES  PIC 99      VALUE  ZERO.

007556      88      VSAM-SB-NORMAL            VALUE  00

007557      88      VSAM-SB-AT-END            VALUE  10

007558      88      VSAM-SB-INVALID-KEY       VALUE  20

007559      88      VSAM-SB-DUPLICATE-KEY     VALUE  22

            88      -------ETC------

            etc..
```

```
011360                PROCEDURE DIVISION

            etc..



095671                WRITE RAN-CLAIM-RECORD FROM XXXXX.

                      IF VSAM-SB-DUPLICATE-KEY.
                                          MOVE XXXX ...ETC...
                                          GO TO LABEL 1.
                                     IF NOT VSAM-SB-NORMAL
                                          GO TO ABEND.
                                     ...Etc...
```

# 1.2.2 Other Considerations or User Rules Guidelines

## 1.2.2.1 AMB User Rules Usage

All AMB user rules must be called with a dollar "$" sign. This enables Software Developers to quickly and easily identify user macros invoked within a program. For example, to call "UVCHKPTERM," code the program with '$UV-CHKPTERM.'

AMB applications should only list the user rule members that are actually accessed by programs within that application.

### Macro Development

While development and maintenance of AMB user rules are the responsibility of the appropriate application development areas, IT Business Systems (ITBS) Host retains final approval of all new or changed user rules.

Changes that could impact more than one application area will be coordinated through the Endevor/Application Productivity System Sub Committee. ITBS Host must attend any Design or Code Reviews that could involve new or changed user rules. Developers will furnish at the reviews scans of existing AMB application and program libraries to document any impact to other areas. Sign off by any

other affected application areas must be coordinated through the Endevor/Application Productivity System Sub Committee and be completed prior to Implementation.

User rules must follow these coding standards:

- The names of user rules must be unique.
- The names of all variables defined by a user rule must be prefixed to indicate that they were inserted by a user rule. The user rule library or name is preferred.

**Example:** A user rule in a member named BCEXMPL might create DATA DIVISION code:

```
01 BCEXMP-CONTROL-AREA.

05 BCEXMP-FIELD1  PIC X  VALUE SPACE.

05 BCEXMP-FIELD2  PIC X  VALUE SPACE.
```

Code placed into a program by a user rule should include comments displaying the user rule name both prior to and after the actual code.

The following guidelines should be used to determine how new user rules will be placed into user rule library members (USERMACS).

- Always-related functions should be grouped together.
  For example, when user rules are written for an OPEN, READ, and CLOSE of a special database, they may be grouped within a single user rule library member.
- User rules called by other user rules should be grouped into the same user rule library member as the calling user rules.
- User rules widely called by other user rules (by design) may be placed into a common user rule library member, such as BCUNVMAC.
- All other new user rules must be individually contained within their own new user rule library member.

# 1.3 Application Development Standards

## 1.3.1 General Discussion

This section of the Information Systems Standards Manual is to document the standards required in all I/S areas for developing software on alternate platforms (Non-Enterprise Server).

All software developed on Non-Enterprise Server platforms will be developed following the standard development methodology. The I/S Governance Governor must approve any deviation from this development methodology.

All software developed on Non-Host platforms will be developed and maintained using one of the standard programming languages, and the development must be under a change control management tool.

All source code must be managed in either Endevor or GitHub Enterprise (GHE). These are the approved source control management tools for Information Systems.

CIAO (Team Studio) may continue to be used for software changes that are developed in Lotus Notes until Lotus Notes is no longer used.

## 1.3.2 Internal vs Vendor Applications

There are two types of applications built to execute on the Non-Host platform.

1. Internal Applications — This is where the bulk of the logic for the system is developed and managed by BlueCross BlueShield of South Carolina software engineers. In this case, some vendor supplied code can be used as appropriate, but the bulk of the core system code is built by our team.
2. Vendor Managed Applications — This is where the bulk of the logic for the system is developed by a vendor. In some cases, scripts, interface programs or other tools can be written by internal resources using tools authorized by the vendor, but the bulk of the system code is written by a vendor.

All new internal applications will be developed using an approved language.

## 1.3.3 General Design Considerations

### 1.3.3.1 Citrix Compatibility

All BlueCross BlueShield of South Carolina developed and/or purchased Non-Host software is required to have the ability to be used within CITRIX.

### 1.3.3.2 Service Oriented Architecture

Software applications will be written in such a manner as to maximize the use of Web services when available. These applications will offload as much processing requirements to the Web service as possible.

# 1.3.4 Programming Languages

## 1.3.4.1 Application/Product Development Languages

The following are the currently approved set of languages for internal application development:

- Java
- HTML
- CSS
- JavaScript
- .NET VBScript / VBA
- XML

## 1.3.4.2 Scripting Languages

Scripting languages can be extremely helpful productivity enhancement and automation tools and can be used in a supporting role within an application system. They are not approved programming languages for end-user processes.

- Perl
- Python/jython
- Windows Powershell
- UNIX/LINUX shell (Bourne, BASH, Korn, C Shell, etc.)
- C (for HP VuGen-based scripting)

## 1.3.4.3 Sunset/Legacy Languages

There are many legacy applications which are written in other languages not in the currently approved list. These applications will continue to be supported, but the application team should work to bring these applications up to date with an approved language.

## 1.3.4.4 OpenMake

Effective 1/1/2014:

- All OpenMake TGT files must be generated by RAD plugin
- TGTs must not be hand edited without waiver from IT Business Systems (ITBS) Non-Host

# 1.4 Enterprise Server - JCL

## 1.4.1 Programming Guidelines

### 1.4.1.1 System Library Descriptions

The purpose of this section is to describe the uses of the various libraries used by all of Systems and Programming for source management and load module storage.

#### BC.NDVR.*.LOADLIB

LOADLIBS should never be referenced in a JCL that executes an application program.

The "LOADLIB" system libraries are used to store non-executable modules, also called sub programs. Programs are placed in the appropriate "LOADLIB" by the linkage editor step of the compile and link PROCs; the Endevor processors do this automatically. The Endevor processors also include “LOADLIB” automatically when the link deck for the main or calling program is generated.

#### BC.NDVR.*.EXECLIB

EXECLIBS must be referenced in a JCL via use of a JOBLIB or STEPLIB statement.

These program libraries are used for executable programs. The PROD1 EXECLIB is where production executables reside. EXECLIB members are created by ENDEVOR MOVE and GENERATE processors when a link deck is generated. The PROD1 EXECLIB is moved from QUAL1; it is not generated again.

#### BC.NDVR.*.JOBDECK / BC.NDVR.*.JOBDECKT

JOBDECK is the library that contains the job decks for all production jobs.

JOBDECKT is the library that contains the job decks for all test cycle jobs using ZEKET.

#### BC.NDVR.*.PROCLIB / BC.NDVR.*.PROCLIBT

"PROCLIB" is a cataloged procedure library. All production cataloged procedures are kept in this library.

PROCLIBT is used for special PROCs used exclusively in test cycle jobs (ZEKET).

The PROCLIBTs are for the use of the Systems and Programming staff during validation.

In order to pick up a JCL from a PROCLIBT, a statement such as the one below must be used in the JCL.

```
//   JCLLIB ORDER=dsn
```

#### BC.NDVR.*.CARDLIB / BC.NDVR.*.CARDLIBT

"CARDLIB" is used to contain members of card images. The card images are data cards that usually follow a "SYSIN DD" statement and don't change from run to run. Examples of "CARDLIB" members are sort control cards, program control cards, and utility control cards. New CARDLIB members must be moved into Production before any JCL that references them may be added to Endevor.

CARDLIBT is used exclusively for test cycle jobs (ZEKET).

## BC.NDVR.*.COPYLIB /BC.NDVR.*.COPYLIB

"COPYLIB" is a system library used for COBOL source statements. Source statement members on "COPYLIB" are included in COBOL and AMB programs with the "COPY" statement.

## BC.NDVR.*.DOCULIB / BC.NDVR.*.DOCULIBT

"DOCULIB" is used to contain members of card images. The card images are documentation text to be input into the documenter procedure (OP01C020). Further explanation of the documenter text can be found in the documenter standards section of the Endevor documentation. DOCULIBT is used for test cycle jobs under ZEKET only.

## BC.NDVR.*.VSAMLIB / BC.NDVR.*.VSAMLIBT

This library is used to contain card images used to define VSAM datasets. Members from this library are used exclusively as input to IDCAMS under the SYSIN DD card. This is a RACF-protected PDS and may be updated only by the VSAM coordinator or Tech Support. No migration procedures exist for this library.

VSAMLIBT is used exclusively for test cycle VSAM dataset definitions (ZEKET).

## Other Endevor Libraries

Endevor libraries have been set up to support a four-stage development process.

1. Libraries with "UNIT1" or "UTDV*" in their names are designed for Unit validation (e.g., small amounts of data through a single program).

2. Libraries with "SYST1" or "STDV*" in their names are designed for System validation (e.g., large amounts of data through a series of programs).
3. Libraries with "QUAL1" in their names are for final validation prior to production moves.
4. Libraries with "PROD1" in their names are designed for production execution.

## ENDEVOR Libraries

```
BC.NDVR.*.APRAPPL
BC.NDVR.*.APRPROG
BC.NDVR.*.APSAPPL
BC.NDVR.*APSCNTL
BC.NDVR.*.APSDATA
BC.NDVR.*.APSEXPS
BC.NDVR.*.APSPROG
BC.NDVR.*.APSREPT
BC.NDVR.*.APSSCRN
```

```
BC.NDVR.*.APSXPROG
BC.NDVR.*.ASSEMBLR
BC.NDVR.*.CARDLIB
BC.NDVR.*.CARDLIBT
BC.NDVR.*.CICSLIB
BC.NDVR.*.CICS.DBRMLIB
BC.NDVR.PROD1.CLIST
BC.NDVR.*.COBOL
BC.NDVR.*.COPYBMS
BC.NDVR.*.COPYCICS
BC.NDVR.*.COPYDB2
BC.NDVR.*.COPYLIB
BC.NDVR.*.DBRMLIB
BC.NDVR.*.DB2FORMS
BC.NDVR.*.DB2PROCS
BC.NDVR.*.DB2QUERY
BC.NDVR.*.DCLGEN
BC.NDVR.*.DDISRC
BC.NDVR.*.DDISYMB
BC.NDVR.PROD1.DISASTER
BC.NDVR.*.DOCULIB
BC.NDVR.*.DOCULIBT
BC.NDVR.*.EASYTREV
BC.NDVR.*.EXECLIB
BC.NDVR.*.JOBDECK
BC.NDVR.*.JOBDECKT
BC.NDVR.*.LINKDECK
BC.NDVR.*.LOADLIB
BC.NDVR.*.OAPSSCRN
BC.NDVR.*.OBJLIB
BC.NDVR.*.OMAP
BC.NDVR.*.OTABLE
BC.NDVR.*.PACKBIND
BC.NDVR.*.PLANBIND
BC.NDVR.*.PROCLIB
BC.NDVR.*.REXXDB2
BC.NDVR.*.SAS
BC.NDVR.*.SCRSYMB
BC.NDVR.*.STORAGE
BC.NDVR.*.USERMACS
BC.NDVR.*.USERMAC2
BC.NDVR.PROD1.VSAMLIB
BC.NDVR.PROD1.VSAMLIBT
```

**BC.NDVR.*.CICSLIB**

This program library is used for online programs.

## 1.4.2 Design Considerations

Job Names, Job Step Names and Cataloged Procedure Names are covered in *Technical Standards — Applications > Application Development Support Tools > Naming Conventions*.

DASD File Allocation Information, VSAM Standards and Practices, Tape Management System, Tape Management Standards - External Tape Labels, Tape Drive Allocation Standards and Multiple Datasets on Tape are covered in *Technical Standards — Applications > File Management Standards and Guidelines > Tape Standards*.

# 1.4.2.1 JCL Standards and Formats

The purpose of this standard is to describe specific requirements and restrictions concerning test JCL. These standards are used to supplement the information available in an IBM OS JCL manual.

## Test Job Card Standards

### Job Name

The job name will be structured in accordance with *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Job Names*.

### Job Accounting Data

Job accounting data is enclosed in parentheses with its sub parameters separated by commas. The sub parameters are:

1. **Job account number**
   Valid five-digit account numbers may be obtained from the Accounting and System Codes found in this chapter (above).
2. **Employee ID**
   This code is used to identify to the system the person responsible for a given job. The format of this field is defined in I/S Employee ID Codes found in this chapter (above).
3. **Substitute date**
   No longer used. A comma should be used to indicate omission of this sub parameter.
4. **Comment**
   The Information Systems engineer not to exceed nine characters may use this field in any way deemed necessary.

### Software Developer's Name

Place the Software Developer's name or other identifying information within quotes. For example:

'John Doe'

'DEV1 SC TEST 1'

There can only be a maximum of 20 characters between quotes. This information is visible when viewing held output in SDSF.

### Class

Refer to *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Other Considerations*.

### MSGLevel

(0,0) or (2,0) is used for compileswhile (1,1) is used for tests.

### Time

For time restrictions on execution classes, refer to *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Other Considerations*.

### MSGClass

Always equals 'X.' This is the Hold queue.

### PRTY

The priority parameter normally defaults to seven (7) and increases by +1 every hour until the job's PRTY reaches 10 or the job begins execution. A PRTY less than seven (7) both can and may be utilized in any class without approval. PRTY zero (0) is highly recommended in those cases where a large number of jobs are submitted and the turnaround window needed is not critical. PRTY zero (0) remains constant. A PRTY of greater than seven (7) and less than 11 requires the approval of your manager or director. A PRTY greater than 10 must have the approval of your senior director.

## Job Card

```
//CAMPJO6T JOB (11300,10373,,'BB10P400'),'CAMPP100',
//  CLASS=L,MSGLEVEL=(1,1),TIME=(1,59),PRTY=8,MSGCLASS=X,
//  NOTIFY=&SYSUID
```

## JOBPARM Card

The JOBPARM card is used to specify the destination of hard copy printed output. It can also be used to specify the LPAR on which the job is to be executed. The format of the JOBPARM card is as follows:

```
/*JOBPARM   R=XXXX              to route printed output to the
                                specified location

/*JOBPARM   S=SYSx              to run on specified LPAR SYSx

/*JOBPARM   R=XXXX,S=SYSx       a comma separates combined parms
```

If a hardcopy print destination is not specified in this manner, printed output will be put on the table outside the computer room and must be retrieved.

## Exec Card

The PARM parameter of the execute card will not be used by applications programs, except in IMS/DB applications.

## DD Card

On the "SYSOUT DD" statement for test jobs, the FCB=parameter should only be used when there are special print requirements.

Dataset names will be structured in accordance with *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > File Names*.

For test executing via a link and go, the target dataset of the link step must be a temporary dataset (i.e., &&Syslmod). If the standard link Prod (linkedit) is being used, the dataset name of the SYSLMOD DD statement must be overridden to be a temporary dataset.

For space allocation, refer to *Technical Standards — Applications > File Management Standards and Guidelines > DASD File Allocation Information* for further information.

For unit parameter information, refer to *Technical Standards — Applications > File Management Standards and Guidelines > DASD File Allocation Information* for DOSL and for tape information, refer to *Technical Standards — Applications > File Management Standards and Guidelines > Tape Standards*. For SYSOUT class, refer to *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Other Considerations*.

For label parameter information, refer to *Technical Standards — Applications > File Management Standards and Guidelines > Tape Standards*.

## Running P Jobs

To execute "P" jobs the following actions must be taken:

1. A RUNSHEET must be submitted through the Technology Support Center Self-Service Portal to Operations.

---

**NOTE** A person submitting a “P” job request is responsible for updating the TSO Toolbox Production support On-call Information for their job name.

---

2. Any "P" job that takes production files off-line requires:

    a. Sign off from the line of business director or above.
    b. Sign off by either a Tech Support Manager, Director of Operations or V.P. of Operations.
    c. Corporate HELP Desk must be notified by phone.

3. In case of ABENDs, the operations escalation process will be to contact the person(s) on the RUNSHEET. If the person(s) responsible cannot be contacted, the manager on the RUNSHEET will be contacted to resolve. If the manager cannot be contacted, the area's systems support or a system support designee will be contacted to escalate.

4. To restart an abended "P" job, you must use the batch response code on the ABEND ticket that was created on INFORM using your jobname. In the response field, describe how to handle the ABEND (e.g., restart from staging, fdone, etc.).

# 1.4.2.2 Production JCL

The purpose of this section is to explain requirements and restrictions concerning production JCL. All production JCL updates must be submitted via the System Support email Account.

These standards are used to supplement the information available in an IBM OS JCL manual.

## Production Job Card Standards

### Job Name

The Job Name will be structured in accordance with *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Job Names*.

### Job Accounting Data

Job accounting data is enclosed in parentheses with its sub parameters separated by commas. The sub parameters will be:

1. **Job Account Number**
   Valid five digit account. Numbers may be obtained from the Accounting and System Codes found in this chapter (above).
2. **Employee ID**
   The same as used on the n5500 report.
3. **Substitute Date**
   No longer used. A comma should be used to indicate omission of this sub parameter.

### Software Developer's Name Parameter

Must contain a brief description of a given job's major function, for example:

- 'Pricing update' 20 characters between quotes maximum.
- 'Suspense maintenance' 20 characters between quotes maximum.

### Class

Refer to *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Other Considerations*.

### MSGLevel

```
(1,1)
```

### MSGClass

Always equals '0' (zero)

### USER=USERID

The USERID must be a valid RACF ID. It is usually the four character LOB code suffixed by the character string "APP" or “AP2” (e.g., USER=CHAPAPP).

### A Job Card Normal Run

```
//CHAPDS0D JOB (35000,,,PROD),'STRIP TRICARE
TRANS',CLASS=7,

//       USER=CHAPAPP,

//       MSGLEVEL=(1,1),MSGCLASS=0,REGION=6M
```

### Password

The password=XXXXXXXX parm on the jobcard should be removed from all production NON RJE jobs.

## Exec Card

The "PARM" parameter of the execute card will not be used by applications programs, except for IMS/DB applications.

Each Software Developer is responsible for abnormal termination of a PROC step based on return codes. When a job must be terminated because a return code greater than zero (0) or some other allowable value was set by a given program, the "code-setting" step should be followed by a PROC step, which checks the return code set in the previous step and executes PGM= ABEND806) when the return code is greater than an acceptable value.

## DD Card

1. In production jobs, "DISP=PASS" will not be used.
2. Dataset names will be structured in accordance with *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > File Names*.
3. For space allocation, refer to *Technical Standards — Applications > File Management Standards and Guidelines > DASD File Allocation Information*.
4. For unit parameter information, refer to *Technical Standards — Applications > File Management Standards and Guidelines > DASD File Allocation Information*.
5. For SYSOUT class, refer to *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Other Considerations*.
6. DD* and/or DD data JCL cards in the run deck will be reflected in a catalogued procedure by the use of the "ddname=xxxxxxx" parameter.

## General Standards

The use of temporary overrides in production JCL decks will be permitted only with the use of a production control log (refer to *Technical Standards — Infrastructure > Operating Systems > Production Control Log*). Permanent overrides in production JCL are allowed only when necessary to adapt common PROCs for use with multiple clients and do not require PCLs once implemented in the production schedule. If overrides are necessary for a special run, I/S systems should supply their own deck to be used in lieu of the regular production deck.

All jobs containing a "restart" parameter will cause the operator to reply before allowing the job to start. Procedures for submitting "P" jobs are explained in *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Design Considerations*.

For IMS programs accessing non-shared databases, the PROC DLIBATDB should be used as a guide in coding production JCL. For shared databases, use PROC DLIDRPDB as a guide.

Use of a STEPLIB DD within a production job or PROC is prohibited.

## Production Remote Job Entry (RJE) Jobs

The purpose of production jobs submitted via RJE will be to place incoming data into production datasets that can then be used for application processing. RJE jobs will not perform any application processing.

The only RJE protocols supported by BlueCross are 2780 3780.

Transmissions require a 2400 or 4800 baud dial RJE and must be bi-synchronous.

The standard BlueCross setup is as follows:

- 3780 protocol
- No compression
- Fixed length records
- No console
- Printer=1
- Punch=1
- Reader=1
- Blocked
- No transparency

RJE jobs will execute PROC 'RJETC100.' This PROC is set up with the following symbolic parameters and defaults:

```
System='BC.PCIS'   starting DSN nodes Device-SYSDG

DSCB='BC.MODLDSCB'

NBLKS01=0007  Primary space
```

```
NBLDS02=0007  Secondary space

OUTCLAS=A,  SYSOUT/SYSPRINT PRINT

RPTCLAS=A,  Report output

OUTSYS='*',  Report print

OUTS=',DST=LOCAL'  Print on local printer
```

The output dataset is a disk GDG. The PROC will produce a report that contains the total records received in a given transmission. One copy of the report will be routed to data control for balancing purposes, and one copy will be routed back to the RJE job submitter.

Only RJE jobs using PROC 'RJETC100' will be allowed to be executed by systems exits. The execution statement must be coded as follows:

```
//EXEC PROC=RJETC100
```

The job name for a given RJE job will be constructed as defined in *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Job Names*. The system code to be used will be that of the application system receiving the incoming data.

The frequency code will be 'U' (for on-request).

RJE jobs will use 'class=8' for execution.

RJE jobs will not use tapes.

RJE jobs must be placed into Production as any other production job. An RJE job will be scheduled each day by the automatic scheduler for execution in a "hold" status. Thus when a given RJE job executes, it will be tracked by the scheduler. The USER=RJECAPP and the PASSWORD=XXXXXXXX must be placed on the jobcard. Contact the EMC Administrator for the appropriate password to code.

An example of execute JCL follows:

```
//SSSSJOOU JOB        (99999,G9999,,'PROD'),'RJE INPUT',CLASS=8,

//                    MSGLEVEL=(1,1), USER=RJECAPP,PASSWORD=XXXXXXXX
```

```
//                    MSGCLASS=0

                      DSN=BC.NDVR.PROD1.EXECLIB,DISP=SHR
//JOBLIB  DD

//PCISSO1O EXEC       PROC=RJETC100,

//                    SYSTEM=SSSS

//RJETNO1O DD  *
```

```
RJE input data stream
```

```
/*
```

```
//
```

RJE jobs must also contain the following record as the last record in their data:

| Field Name | Description | Usage | From | To |
|---|---|---|---|---|
| Record Type | Constant- 'transtrl' | A | 1 | 8 |
| Submitter ID | Unique identifier for the submitter | A | 9 | 20 |
| Record Count | Total of all 80 byte records being transmitted, including this one. | N | 21 | 15 |
| Filler | Spaces | A | 26 | 80 |

This record will be used by the RJETP010 program to balance the number of records in the record to the number of records computed by the RJETP010 program. The 'transtrl' record will be dropped by RJETP010 prior to the file being catalogued.

## Retention of Production JCL SYSOUT

All production "JCL SYSOUT" is retained online available for look-up via TSO and CICS using the procedures outlined below.

All production "SYSOUT messages" such as sort parameters, displays, and Easytrieve parameters should be written to SYSOUT=C (refer to *Technical Standards — Applications > Application Coding > Enterprise Server – JCL > Other Considerations*) and are retained online available through the SDSF for viewing only (it cannot be printed or purged) until archived to tape. Once SYSOUT C files are archived, output may be retrieved using PRTCLC.

Production jobs must have "MSGCLASS=0" in the JCL and must be submitted via the scheduling system. All JCL with or without error will be put to a master file for online access via PRDSPOOL and CICS.

If an error other than condition codes is detected in the JCL by the system, the JCL will be printed and also put to a master file.

All of these errors will flash on the operator's console, and all are not able to be deleted.

Each morning a report will be printed with all errors and condition codes reflected for use by programming. This O488 report will be distributed to the responsible manager of each I/S area. It is the responsibility of the manager to identify his jobs with errors and to notify his personnel. This is a double check for the daily problem logs produced by Operations.

The following steps should be followed to view production JCL via TSO:

1. The menu item PRDSPOOL is available under the Application Development Dialog.
2. Initial selection screen:
   This screen prompts you for criteria that the interface will use to find jobs on the archival database. The jobname field is optional and may contain a partial name. This allows generic searches from the database. Start date and JES2 job number are optional also, but if either is entered, the entire field must be filled in. If any jobs on the database match your criteria, a selection list will be displayed next. Otherwise, a message indicating a no-hit situation will be displayed.
3. Selection list:
   This list shows all jobs on the database that match your selection criteria. To view the job's output you must enter an "S" to the left of the desired member. You may view the job's profile record by entering a "P" to the left of the desired member.
4. Select for viewing:
   This option will place you in an ISPF browse mode.
5. Profile record:
   The profile record displays any error message trapped by JHS along with various statistical information. If you are using the DCMS option, certain fields will be filled in with DCMS information.

The JCL retention available online will be as follows (Table 1-2):

**JCL Retention Time Lines**

| Job # Suffix | Frequency | Online Retention |
|---|---|---|
| D | Daily | Eight (8) days |
| B | Twice weekly | 30 days |
| W | Weekly | 15 days |
| B | Twice monthly | 30 days |
| M | Monthly | 45 days |
| Q | Quarterly | 45 days |
| S | Semi annual | 45 days |
| A | Annual | 45 days |
| U | User request | 15 days |

**Table 1-2** JCL Retention Time Lines

## Viewing JCL via CMOD

To view JCL via CMOD, follow these steps:

1. Sign into the Enhanced Seamless Search (ESS) viewer http://essearchc-prod.bcbssc.com.
2. Enter PRDSPL in the folder name and open the folder.
3. Select the date range that you wish to search and the job name and press Enter.
4. Click on the correct line of the hit list to view.

**NOTE** Production job log information is kept permanently in CMOD.

**NOTE** If you do not see the PRDSPL folder in your original screen, you will need to request RACF access via TSC Self Service.

# 1.4.3 Other Considerations

DASD File Allocation Information, VSAM Standards and Practices, Tape Management System, Tape Management Standards - External Tape Labels, Tape Drive Allocation Standards and Multiple Datasets on Tape are *Technical Standards — Applications > File Management Standards and Guidelines*.

## 1.4.3.1 Job Classes

The purpose of this section is to define Job classes used within the corporation and restrictions on each class. For all job classes, there are job name restrictions (refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Job Names*). Restrictions are specified under each item.

### Customer Easytrieve Jobs

If an Easytrieve is to be executed, you may run in the following classes: M, N, O, P, or Q. (Jobs that print over 20,000 lines, use any dataset with over five volumes, or exceed two (2) minutes of CPU time must be submitted as an "N" job. Distribution of these jobs is controlled by the JOBPARM route code. If no route code is listed, the job output will be found on the output table located outside of the computer room by the Xerox machine.

Job Name Restriction: The job name must start with 'EASY' and end with 'T' or 'N' to run in these job classes.

```
For class = M   No (0) tape drives allowed; 15 seconds or less;
                outlim=10,000

For class = N   One (1) tape drive allowed; eight (8) seconds or
                less;  outlim=5,000

For class = O   One (1) tape drive allowed; 15 seconds or less;
                outlim=10,000

For class = P   Two (2) tape drives allowed; 15 seconds or less;
                outlim=10,000

For class = Q   Three (3) tape drives allowed; two (2) min or less;
                outlim=10,000
```

### Information Systems Test Jobs

Job Name Restriction: "T" is the allowable trailing job name character.

```
Execution Job Classes Follow:    (JOBCARD="CLASS").
```

A TIME parameter must be coded on the job card for each class listed below (Table 1-3) except for class "C," class "H," and the maximum time specified may not exceed the maximum listed under restrictions. For jobs that require two minutes or more of CPU time, refer to the subsection *Production Test Jobs* below.

**Time Parameters**

| Class | Restrictions |
|---|---|
| "C" | COMPILES, LINKS AND Endevor Only 60 CPU seconds or less (see below in this subsection) |
| "E" | 02 SECONDS, NO TAPES |
| "F" | 08 SECONDS, NO TAPES |
| "H" | High Volume COMPILES, LINKS AND Endevor only (see below in this subsection) |
| "V" | 15 SECONDS, NO TAPES |
| "L" | 1 minute 59 SECONDS, NO TAPES |
| "R" | 04 SECONDS, 1 TAPE |
| "S" | 04 SECONDS, 2 TAPES |
| "X" | 1 minute 59 SECONDS, 01 TAPE (ONE) |
| "Y" | 1 Minute 59 Seconds, 02 Tapes (Two) |
| "Z" | 1 Minute 59 Seconds, 03 Tapes (Three) |
| "6" | None, TRICARE Systems Managed |
| "J" | None, Private Systems Managed |
| "1" | None, Medicare Systems Managed |
| "W" | None, Corporate Systems Managed |
| "8" | Valid for ZEKE Test Jobs Only |

**Table 1-3** Time Parameters

The following system prefixes are excluded from this procedure:

```
TSOO, DASD, CDMS, & APSP.

/*JOBPARM ROOM=XXXX     Follows jobcard and routes output for deliveries.
Compiles, link edits or Endevor jobs

Job Name Restriction: Allowable trailing job name characters: "C" or "H".
```

| | |
|---|---|
| For class = C | Only used for compiles, Endevor/MVS compiles updates, or link edits that take 60 CPU seconds or less. Do not use for link and go jobs. |

| | |
|---|---|
| For class = H | High Volume Only used for compiles, Endevor/MVS compiles, updates, or link edits. Do not use for link and go jobs. |

## Production Special Run Jobs and Production Reruns

"P" is the allowable trailing job name character. This naming convention is particularly important in order not to cause problems with production job scheduling.

| | |
|---|---|
| For class = A | Used for production special run jobs that create/modify production files or reruns of individual production jobs. |

Complete "reruns" jobs or cycles must be submitted as "P" jobs and must follow production JCL execution class to seven (7) or eight (8).

## Production Test Jobs

Job Name Restriction: "N" is the allowable trailing job name character.

| | |
|---|---|
| For class = A | Only used for long running test jobs (two minutes or more of CPU time), which do not create production files. These jobs follow all test job restrictions except:<br>• No CPU time limit,<br>• No output line limit,<br>• No tape limit, will access tape drives after Production |

| | |
|---|---|
| For class = 6 | No tape limit; test cycles only are to be submitted in class=6. |

## Production Jobs

Job Name Restriction: Allowable trailing job name characters: "A, B, D, M, Q, R, S, U, W"

| | |
|---|---|
| For class = A | Used for all production jobs. Only constant checks are made against this class. Scheduling will change class to seven (7) or eight (8) if scheduled. |

## JCL Scans

JCL scans - use "TYPRUN=SCAN" in job card.

TYPRUN=SCAN will identify JCL errors relating to mismatches between job class and the TIME parameter or the number of tape drives allowed. It will not identify mismatches of class and valid job name. Mismatching of job class and name is identified at execution time.

## 1.4.3.2 SYSOUT Classes

**SYSOUT Classes**

| Class | Used for: |
|---|---|
| A | Printer Output, 150 char max |
| B | SMTP |
| C | PRINTCLC - Special (dumps), JHS overflow > 25000 lines |
| D | TBINFOPK - Trailblazer INFOPACK |
| E | Printer Output - >3000 lines, printed at Willoughby |
| F | |
| G | Production 2 UP 18 x 11 DUPLEX no hole cut sheet - IP4000 |
| H | Special Payroll - 3900 or 4245 printer - HR rep must be present |
| I | Special forms, 11 x 8 1/2 - 3900 printer, 132 char max |
| J | Test Jobs, INFOPAC standards |
| K | Personnel Test Jobs - HR rep must be present |
| L | PRODSPOOL - Job Information copied to JHS - Production Only |
| M | |
| N | Special forms, 9 1/2 x 11 - 3900 printer |
| O | Special - Trailblazer |
| P | Payroll - Printed at Willoughby - HR rep must be present |
| Q | Online viewing of Production (INFOPACK?) |
| R | |
| S | IBM 4245 impact printer. |
| T | Special - Trailblazer |
| U | IP4000 Printer - req. PRMODE (PRIV, CHAP, COMP, MEDA, MEDB, TEXS) |
| V | |
| W | IBM 3130 printer |
| X | SYSOUT HOLD queue. Delete after 5 days. |
| Y | |
| Z | Special Payroll - 4245 printer - HR rep must be present |
| 0 | JHS - Production Only |
| 1 | Production 2 UP 18 x 11 DUPLEX - IP4000 |
| 2 | Production 2 UP 18 x 11 SIMPLEX - IP4000 |
| 3 | Production 2 UP 18 x 11 SIMPLEX holed - IP4000 |

**SYSOUT Classes**

| Class | Used for: |
|---|---|
| 4 | SYSDUMP for disk storage and online viewing |
| 5 | Tech Support reserved |
| 6 | Production 2 UP 18 x 11 DUPLEX no hole - IP4000 |
| 7 | Production 2 UP 18 x 11 SIMPLEX holed cut sheet - IP4000 |
| 8 | CA-View (BCBSA, Hosted Customers) |

**Table 1-4** SYSOUT Classes

## Test SYSOUT Class Requirements

All test jobs should use "SYSOUT" class X unless one of the exceptions listed below is applicable. All "SYSOUT" class X output is first directed to the output processor. "SYSOUT" class X output needs to be released from the output processor to print in the hold queue. All "SYSOUT" class X on hold in the output processor will stay in the computer until released or deleted by programming. After remaining in "SYSOUT" class X for two days (excluding weekends and holidays), the computer will delete the output from "SYSOUT" class X.

Test jobs with output greater than 3000 lines may be re-queued to "SYSOUT" class E and released to print. SYSOUT class E output will be printed at "I/S Print Operations" and routed to your mailing address based on your /*JOBPARM ROOM= number. Test jobs with 3000 lines or less can be released to "SYSOUT" class '[A' and the remote printer name nearest your location. See your management for the remote printer name nearest your location.

All Payroll test jobs to be printed are to be directed to "SYSOUT" class P and never printed, unless an HR representative or I/S HR programming representative is present.

All Personnel test jobs to be printed are to be directed to "SYSOUT" class K and NEVER printed, unless an HR representative or I/S HR programming representative is present.

"SYSOUT" class J can be used for test jobs. However, if used, it must conform to INFOPAC standards in Automated Report Distribution (Refer to *Procedures & Tools > Host Tools > Developer Tools > Host Development Environment* for additional information.).

Remember, output will not have "TEST" printed on the document because the output will have a production name.

All test job printed output from I/S Print Operations will have the word "TEST" shaded on the printed output. This includes special forms. All test jobs printed on the IP4000 using standard default FCB and FormDef will print 2 UP duplex, hole paper, and cut sheet. A cut sheet is four (4) logical pages per one (1) physical cut sheet page. Direct output to SYSOUT class E.

## Production SYSOUT Class Requirements

All special forms that are printed on hardcopy and are not printed on the IBM 3900, IP4000, or IBM 3130 printers should be printed on the IBM 4245 impact printer. Impact printer SYSOUT should go to

SYSOUT=S. SYSOUT=W goes to the IBM 3130 printer. Notify ICT Print Operations via the Outlook public mail list of PRINT.AFP in order to have an FCB created. Once the FCB has been created and validated, you will be notified. Listed below are standard FCBs which may fill your print needs.

```
BC01 - landscape,   6 lpi

BC02 - landscape,   8 lpi

BC03 - landscape,  12 lpi

BC06 - portrait,    6 lpi

BC07 - portrait,    8 lpi

BC08 - portrait,   12 lpi
```

All production, one part, printing 150 or fewer print positions, are required to be in SYSOUT=A . Print lines in excess of 150 print positions are not currently supported. See an I/S Print Coordinator if you have questions.

All special forms (pre-printed) or blank forms to be printed on 9-1/2 x 11 form are required in SYSOUT=N and must be directed to the 3900 printer.

All special forms to be printed on 11 x 8-1/2 form and one part requiring more than 132 print positions are required to be in SYSOUT=I and must be directed to the 3900 printer.

Trail Blazer SYSOUT information:

- SYSOUT class O is a special SYSOUT class for Trailblazer.
- SYSOUT=T is a special SYSOUT class for Trailblazer.
- SYSOUT=0 (zero) is a special SYSOUT class for Trailblazer.
- SYSOUT=D is for TBINFOPK, Trail Blazer's Infopk utility for online reporting.

All production checks to be printed on the IP4000 printer must be in SYSOUT=U with proper PRMODE. The PRMODEs are as follows:

| | |
|---|---|
| Private Business | PRIV |
| TRICARE | CHAP |
| Companion Life | COMP |
| Medicare A | MEDA |
| Medicare B and DMER | MEDB |

Trailblazer TEXS

All Payroll reports to be printed on 11 x 8-1/2 stock form are to be in SYSOUT=P and never to be printed unless an authorized HR representative is present.

All Personnel reports to be printed on 11 x 8-1/2 stock form are to be in SYSOUT=K and never to be printed unless an authorized HR representative is present.

Coded data replaces fiche that is no longer used. Call the Print Coordinator in I/S PRINT OPERATIONS if you have questions.

All special forms (pre-printed) to be printed on the IBM 3130 should be in SYSOUT=W.

Special SYSOUT in excess of 100 lines, such as sort parameters, displays, and Easytrieve parameters that are stored by data control should go in SYSOUT=C. This class is available through the SDSF for viewing only (it cannot be printed or purged) until stored to tape backup. To retrieve the SYSOUT once it has been put to tape, refer to the Production Support Menu on the Tool Box and choose the PRINTCLC item.

The first 25,000 lines of any print with SYSOUT= 0 will be stored by JHS.

Lines, in excess of 25,000, will be sent to PRINTCLC.

SYSOUT under 25,000 lines can go to SYSOUT=0 and will be stored on JHS. Any SYSOUT can be directed to JHS.

All production SYSDUMP SYSOUT should be directed to SYSOUT=A for paper copy or SYSOUT=4 for disk storage and online viewing. SYSOUT=4 outputs go to a dataset with the following name format: `BC.DMPX.Jobname.DYYYYDDD.THHMMSS.Jobnumber.`

All special payroll forms to be printed on the 3900 printer are required to be in SYSOUT=H and NEVER to be printed unless an authorized HR representative is present.

All special payroll forms to be printed on the 4245 printer are required to be SYSOUT=H or Z and NEVER to be printed unless an authorized HR representative is present.

All production that is to go through the report distribution system must be in SYSOUT=J. This must be coordinated with the Report Distribution Coordinator (who is part of Data Distribution).

Online viewing of Production using the report distribution system will use SYSOUT= Q.

All production output printed on 2 UP format, duplex, cut sheet, or hole paper, and uses 18 x 11 paper size should be printed on IP4000 printer with SYSOUT=1.

All production output printed on 2 UP format, simplex, fold output, and uses 18 x 11 paper size should be printed on IP4000 printer with SYSOUT=2.

All production output printed on 2 UP format, simplex, hole paper, and cut sheet should be printed on IP4000 printer with SYSOUT=3.

SYSOUT=5 is a special SYSOUT that is used solely by TECH SUPPORT. DO NOT USE THIS SYSOUT AT ANY TIME!

All production output printed on 2 UP format, duplex, no hole, fold output and uses 18 x 11 paper size should be printed on IP4000 printer with SYSOUT=6.

All production output printed on 2 UP format, simplex, no hole paper, and cut sheet should be printed on IP4000 printer with SYSOUT=7.

All production output printed on 2 UP format, duplex, no hole, cut sheet and uses 18 x 11 paper size should be printed on IP4000 printer with SYSOUT=G.

To produce a production report from a test job without "TEST" shaded on the output, the job must be run as an "N" job (e.g., EASYJO1N). "N" jobs can be submitted from your account code and do not require run sheets. SYSOUT class must be a valid production output class.

# 1.5 Language Specific Standards

## 1.5.1 Java

All Java code is stored in the BlueCross BlueShield of South Carolina (BlueCross) Enterprise GitHub. Java code standards are published and managed in the BlueCross Enterprise GitHub on premises and are available for enforcement in SonarQube. All Java code is written to these published standards. BlueCross has adopted a "Clean as You Code" approach to making sure our code meets the approved standards and practices.

### 1.5.1.1 Java Code Conventions

#### File Organization

For an example of a Java program best practices related to file organization, refer to the Java ISSM repository (java-sonar-config-bcbssc-issm) in GitHub.

##### Java Source Files

Each Java source file contains a single public class, interface or enum. When an inner-class, interface or enum is declared private, it must be stored within its associated public class. The public "top" class being the first class or interface in the file in which the class is named.

Java source files have the following ordering:

- Package and Import statements
- Java Doc comments
- Fields
- Class and interface declarations (Refer to the Best Practices guide in the ISSM Java SonarQube Standards section of the GitHub Repository for more information.)

##### Beginning Copyright Comments

This copyright statement must be at the head of each Java file.

```
/*
 * This Material Is The Confidential, Proprietary And Trade Secret Product Of
 * BlueCross BlueShield of South Carolina And Its Subsidiaries.  Any
 * Unauthorized Use, Reproduction Or Transfer Of These Materials Is Strictly
 * Prohibited.
 * Copyright 2023 BlueCross BlueShield of South Carolina
 * All Rights Reserved.
```

```
*/
```

### Package and Import Statements

The first non-comment line of most Java source files is a package statement. The package statement must include the line of business or product. After that, other import statements can follow.

## Naming Conventions

Naming conventions (**Error! Reference source not found.**) make programs more understandable by making them easier to read. They can also give information about the function of the identifier (e.g., constant, package, class), which can be helpful in understanding the code.

See the Best Practices guide in the ISSM Java SonarQube Standards section of the GitHub Repository for naming conventions.

# 1.5.1.2 SonarQube

All new and updated code must be checked against the "BCBSSC_ISSM" SonarQube quality profile, which houses the current ISSM Java standards for code quality.

There are two methods for applying a quality profile:

1. Using the SonarLint Plugin in the IDE.
2. Using a pipeline in GitHub to run the SonarQube method to apply a Quality Gate.

# 1.5.2 Hypertext Markup Language

Hypertext Markup Language (HTML) is a tag-based markup language that describes the structure and layout of web page content. The World Wide Web Consortium (W3C) maintains the standards that define HTML. HTML5 is the latest major version of HTML recommended by the W3C. All new applications need to be written in HTML5, and legacy applications should consider a migration plan for moving to HTML5.

HTML elements, which include tags and attributes, semantically describe the content they contain. Attributes modify the default action associated with an HTML tag and can be required or optional.

# 1.5.2.1 General Guidelines

## Tags

HTML tags that have an end tag must use an end tag: `<a> </a>`, `<p> </p>`. This does not apply to elements that do not contain content, and only have attributes that affect or relate to elements or information in the HTML document, such as:

- `<br>` — line break
- `<hr>` — horizontal rule (also called a *thematic break* in HTML5)
- `<link>` — link
- `<img>` — image
- `<input>` — defines an input control

HTML tags and attributes must always be lowercase.

As shown below, attributes only appear in the start tag, and their names must always be lowercase characters with their values contained in double quotes.

```
<a href="http://google.com">this is the link text</a>
```

## Basic HTML Element Hierarchy

The following example shows the structural hierarchy used for HTML document elements and tags.

```
<!DOCTYPE html>

<html>

    <head>

        <title> </title>

        <meta />

        <link />

        <script> </script>

    </head>

    <body>

        <!-- Begin Section 1 -->

        <div>

            <h1></h1>

            <img />

            <p></p>

        </div>

        <!-- End Section 1 -->

        <!-- Begin Section 2 -->

        <div>

            <h2> </h2>

            <p> </p>
```

```
            <ul>
                <li> </li>
            </ul>
        </div>
        <!-- End Section 2 -->
    </body>
</html>
```

## Avoidance of Table-Based Page Design

Do not use the table element to define any kind of page structure for a website. Page design that does not depend on tables lets you create pages that are quick to load and exhibit faster performance.

In the early days of web design, tables were used to define page structure. By using images and tables within other tables, a web developer could achieve complex designs. Cascading Style Sheets (CSS), which is discussed later in this chapter, eliminated the need for this approach, but lack of support in major browsers, lack of familiarity with CSS, and tools that continued to encourage table-based page designs perpetuated this undesirable practice.

Table-based designs also increased bandwidth usage and rendering costs due to all of the empty `<table>`, `<tr>`, `<td>` tags that make up a large portion of the HTML document. Even though the tags are empty, the browser must traverse the table and render it, which leads to longer load times.

**NOTE** Web page design that does not use tables does not mean that you must never use tables. Tables are meant for tabular data presentation and should be used as such.

## Avoidance of Inline Styles

Avoid inline styles. Inline styles are styles that only appear inside an HTML tag in a style attribute. All styles must be stored in an external style sheet. Refer to *Technical Standards — Applications > Application Coding > Language Specific Standards > Hypertext Markup Language* for more information.

**Example:**

```
      <a href="http://www.google.com" style="color: green">
```

The preceding example does not meet the standard of semantic HTML because the presentation is not separated from markup. There is also no way to reuse the style elsewhere. Inline styles will only change the color of one link to green. To make it consistent, you would have to apply the inline style to every single `<a>` to change the color of every link to green. For an explanation of semantic HTML, refer to the subsection *Semantic HTML* below.

The solution is to write the following in a style sheet for all instances of the `<a>` tag:

```
a {

    color: green;

}
```

During early validating and development, style blocks within pages are acceptable, but they must be moved to the external style sheet for production deployment. Refer to *Technical Standards — Applications > Application Coding > Language Specific Standards > Hypertext Markup Language* for more information.

### Comments

Use Comments to add notes about how the code has been constructed and for marking the beginnings and endings of content sections. The browser ignores the comment tag content and it is visible only when viewing the HTML source code. While comments are not visible within the rendered page, they are visible within any browser's development tools and *view page source* functions. As a security precaution, never use HTML comments to *debug* sensitive information such as credentials, system errors or statuses, or any values in general that are not appropriate to share with business end users.

**Example:** `<!-- This is a comment -->`

### Semantic HTML

Practice semantic HTML. Semantic HTML refers to the use of HTML markup to support the meaning of information content in web pages rather than defining its appearance. The practice of semantic HTML should be used so that content markup can be easily interpreted by screen readers for accessibility. This also ensures markup that is clean, concise and easier to update. The introduction of CSS let web developers move away from defining content presentation using HTML tag attributes and use CSS to define the presentation instead.

### Accessibility

Web accessibility means that a website is designed and developed so that people with disabilities can use them. According to Section 508 of the Rehabilitation Act, all public facing web applications must adhere to the Web Content Accessibility Guidelines (WCAG) 2.0. Following these guidelines will ensure that your web page is perceivable and operable by a wider variety of users.

An accessibility scanning tool is recommended to assist with finding any guideline violations. Current suggestions are HTML_CodeSniffer, Compliance Sheriff or an integration validation library such as axe-core.

## 1.5.2.2 HTML Structure

HTML5 introduced a few new element tags for structuring content using different layout types. CSS must be used to define the classes in the examples — the example HTML alone will not produce the desired layout.

### One Column with Header, Navigation, Content, Footer

```
<body>
    <div class="wrapper">
        <header class="site-header">
            <!-- Header Content Here -->
        </header>
        <nav class="site-nav">
            <!-- Navigation Content Here -->
        </nav>
        <main class="main-content">
            <!-- Content Here -->
        </main>
        <footer class="site-footer">
            <!-- Footer Content Here -->
        </footer>
    </div>
</body>
```

## 1.5.2.3 Cascading Style Sheets

Cascading Style Sheets (CSS) provides a way to associate presentation information with the HTML elements of a page without distorting the underlying content structure. From a browser perspective, the style sheet is a set of guidelines that indicate how the elements of HTML-tagged content should be presented. The CSS Working Group of the W3C maintains the standards for CSS.

The visual format of commonly used HTML elements can be specified in an external CSS file, and this single style sheet can be referenced by multiple web pages. Using CSS separates presentation from content markup, so that changing the appearance of a whole website can be done by simply changing the related style elements in the CSS.

By targeting the *class* attributes of the HTML elements on those pages, styles can be specified within the style sheet and applied to any element on any page referencing that style sheet.

By using multiple style sheets, and different combinations of those style sheets, many varying designs can be applied to the same core HTML code.

CSS consists of rules that have three components:

- Selectors
- Properties

## Selectors

Selectors are identifiers within a style sheet that are used to associate the style with the HTML element where you want to apply the style. In a single style sheet, selectors define a group of styles to be applied to a specific HTML subject or to a group of elements that specify a style is to be applied only in a particular context. See Table 1-5 below for examples of selectors.

**CSS Selectors**

| Selector | Description | Example |
|---|---|---|
| Type | Affects all instances of an HTML tag | `body {`<br>`  background-color: #000;`<br>`}` |
| Descendant | Affects all descendants of an HTML tag | `h1 em {`<br>`  color: red;`<br>`}` |
| Child | Affects all children of an HTML tag | `body>p {`<br>`  line-height: 1.3;`<br>`}` |
| Attribute | Affects all elements with the corresponding attribute value (`rel="copyright"`) | `a[rel="copyright"] {`<br>`  text-decoration: none;`<br>`}` |

### CSS Selectors

| Selector | Description | Example |
|---|---|---|
| Class | Affects all HTML elements that reference the same class (`class="big-border"`) | `.big-border { border: 20px solid #CCC; }` |
| ID | Affects an HTML element of the same ID (`id="column1"`) (IDs should only be assigned to a single element, whereas classes are used for multiple elements.) | `#column1 { padding: 4px; }` |

**Table 1-5** CSS Selectors

Selectors can be chained together using four different CSS Combinators (Table 1-6).

### CSS Combinators

| Combinator | Selector | Example |
|---|---|---|
| Blank space | Descendant | `div p { background-color: yellow; }` |
| > | Child | `div > p { background-color: yellow; }` |
| + | Adjacent Sibling | `div + p { background-color: yellow; }` |
| ~ | General Sibling | `div ~ p {` |

**CSS Combinators**

| Combinator | Selector | Example |
|---|---|---|

```
    background-color: yellow;

}
```

**Table 1-6** CSS Combinators

One pseudo-element may be appended to the last simple selector in a chain, applying the styling to a subpart of each HTML subject.

### Attribute Selectors

Use attribute selectors to create CSS classes that match a particular attribute of an HTML object. These are often used in combination with inputs such as text boxes or radio buttons. For instance, an attribute selector such as `input[type="textarea"]` refers only to inputs that are text boxes.

## Properties

CSS properties define what aspect of the selector will be changed or styled. They are the specific stylistic characteristics that are applied to the referenced HTML selector. The property name is written first, followed by the property value.

**Example:**

```
color: #222;

font-size: 14px

font-weight: bold;
```

## CSS Guidelines

When developing CSS, use the following guidelines:

- Avoid inline styles wherever possible.
- Use properly formed style sheets applied to semantic and valid HTML to achieve the goal of maintaining a separation between the structure and the presentation of a web page.
- Use a top-down organizational structure to make finding styles easier. For ease of use, you should also list properties alphabetically.
- Combine CSS elements instead of duplicating code.
- Design for cross-platform compatibility. Minimize the use of styles that are browser dependent.
- Refer to HTML elements (`div`), classes (`.class`), pseudo-elements (`:before`), and pseudo-classes (`:first-child`) with CSS selectors if desired.
- Do not use CSS selectors that reference element IDs because that impairs modularity. Use common classes instead.

- Follow CSS selector names by brackets `{ }` with the attributes for each selector contained within the brackets.
- Ensure that CSS properties are in the format: *`property: value;`* The property name is written first, followed by a colon, followed by a space and the property value, followed by a semicolon.
- Separate multiple words in property names by hyphens.
- Ensure that CSS properties and selectors are always lowercase. Use hyphens (–) to separate words or improve legibility. Underscores ( `_` ) are not permitted.
- Write each property on a separate line to maintain legibility, unless there is only one property for the selector.
- Ensure that color values are in hexadecimal format.
- Use descriptive naming conventions to make it easier for others to understand the code.
- Build on and reuse a common CSS framework to help speed up the design and prototyping process. In other words, don't create a new Cascading Style Sheet for every page regardless of incremental design.
- Create responsive layouts using CSS media queries.
- Use generic classes for commonly used styles (such as hidden, left, right).
- Use the current recommended tool Visual Studio Code for writing CSS. It offers features such as syntax highlighting, auto-complete style definitions, and color value previews.

## Use of Multiple Cascading Style Sheets

Because browsers read the style sheets in a cascading sequence, multiple style sheets can be applied to a single HTML page. Styles are applied with precedence given to style sheets in the order in which they are referenced by the HTML page.

For example, if *stylesheet1.css* is referenced in the HTML before *stylesheet2.css*, then the styles in stylesheet2.css will override the corresponding styles in stylesheet1.css.

### Cascading Examples (with the Same Degree of Specificity)

In the example below (from *example1.css*) a black background color is applied to a `<div>` tag.

```
div { background-color: #000; }
```

If the HTML references a second style sheet (") after example1.css but uses the same degree of specificity to apply a white background color, the second style sheet overrides the previous style:

```
div { background-color: #fff; }
```

This same concept applies to styles with the same degree of specificity within the same style sheet, with the preference given to the last selector listed in the CSS document.

### Cascading Examples (with Differing Degrees of Specificity)

In the example below, a black background color is applied to a `<div>` tag.

```
div { background-color: #000; }
```

If a higher degree of specificity (in this case a class called “sidebar”) is used to apply a white background color, it will override the previous style. The order in which these styles are specified within a single style sheet, or across multiple style sheets does not matter.

```
div.sidebar { background-color: #fff; }
```

This same concept applies to styles with the same degree of specificity within the same style sheet, with the preference given to the last selector listed in the CSS document.

# 1.5.3 IBM Forms

## 1.5.3.1 IBM Forms Definition

IBM Forms provides for creation, deployment, and streamlining of eXtensible Markup Language (XML) forms-based processes. The application developer uses the IBM Forms Designer (Designer) for code generation. The eXtensible Forms Description Language (XFDL) Specification describes a class of XML documents called XFDL forms and partially describes the behavior of programs that process them. An XFDL processor is software that reads, processes, and writes XFDL forms. Graphical User Interface (GUI) rendering, data extraction, or modification are tasks which may be included in the processing.

## 1.5.3.2 When to Use IBM Forms

IBM Forms should be considered whenever there is a simple field conditional logic based on data entry that can be handled by the form itself. IBM Forms may also accommodate complex, cross-field calculations. However, business logic should utilize servlet calls to the server. This provides the ability for the servlet/portlet to run the business logic and return the response to the form. IBM Forms should also be avoided when there is a need to accommodate numerous entry fields or for placing business logic on the client.

Database queries are another factor to consider when determining whether to use IBM Forms. Delays could result if numerous queries must be done.

Decisions on whether to use IBM Forms should be made based on the requirements of each work effort taking into consideration application complexities, the number of entry fields, and database queries.

## 1.5.3.3 Source Files

### File Names

Always use meaningful file names. As an example, an employee transfer form would be EMPLOYEETRANSFER.xfdl rather than EMPTRA.xfdl. Always use upper case for the file names.

### Comments

All XFDL files should begin with a detailed comment describing the business purpose. Any technical information which makes the processing of this file unique should be included here as well.

All comments should use the standard HTML comment tags (i.e., <!- - and - - >).

# Structural Standard for Every Source File

Each source file should contain the following four sections:

1. The XML and XFDL tags should always be located at the top of the form.

2. Copyright comments should be placed after the standard XML and XFDL tags.
3. Xformsmodels should occur next. The instances used in the form should be declared here, following the compute statements and form submissions.
4. The Unique Value Finder (UVF)/GUI Design should be the final section.

## Xformsmodel Section Standards

### Declarations

- Usage and attribute specifics should be commented for each Xformsmodels instance.
- Declare a variable's instance and all variables under that instance.
- Declare one variable per line; see below.
  <totalnumberOfRows></ totalnumberOfRows>
  <maximumNumberOfRows></ maximumNumberOfRows>
- Variables may be initialized by using the properties dialogue. Initialization at Declaration may be done as shown in the examples below.
  <totalnumberOfRows>25</ totalnumberOfRows>
  <maximumNumberOfRows>16</ maximumNumberOfRows>
- Use camel case when naming fields and variables of the form. For example, if the name of the field is first name, use firstName as the sid of the field.
- Use a name that identifies the field specifically. For example, if there is an entry field for first name, then the data tag for that should be <firstName></firstName>.
- If there is more than one occurrence of the first name, create a data binding specific to the first name and to the context. For example:
  Applicant first name : <applicantFirstName></applicantFirstName>
  Dependent first name : <dependentFirstName></dependentFirstName>
- Always specify the length of fields. Otherwise IBM Forms defaults to the content length enhancing the possibility of data truncation.
- If the label needs to dynamically wrap text, do not specify the label height or text may be truncated.
- Do not break the compute statements; Designer aligns them into single lines with white spaces.

### File Sizes

File sizes are reduced by setting the font type and size as a global setting based on the requirements of each customer. The following lines for each label or field should be removed and set as a global setting:

```
<fontinfo>

          <fontname>Arial</fontname>

          <size>9</size>
```

```
    </fontinfo>


<labelfontinfo>
          <fontname>Arial</fontname>
          <size>9</size>
    </labelfontinfo>
```

**File Submission**

Two types of form submission are available. They are:

1. X-Forms Submission
2. XFDL Submission

Use X-Forms Submission when there is only one instance from the form and not the entire XFDL and its items. Use the XFDL submission when the entire form and all contents are necessary.

Use the X-Forms Submission to call a servlet from the form. A simple instance should be passed as an XML format to the servlet. The response from the servlet should be an XML instance.

## UVF/GUI Design Section Standards

**Form Size**

When a new form is created, form globals and page globals are to be set so that the form size is decreased; the form globals will be applied to the whole form, whereas page globals will be applied to that particular page.

Set the pagesize property of the globalpage in the Designer when creating forms to ensure they are rendered properly. When creating a form in the Designer, ensure you set the pagesize property of the globalpage, located in the Properties view under Appearance to the size of your finished form. Setting the page size ensures that when forms are opened, the user will see the entire form and that all form pages are the same size. This is set as per the customer requirement.

Set the saveformat in the global page as either an XFDL format or a Zip format. For large files select the saveformat as an **application/vnd.xfdl;content-encoding="base64-gzip"**. This ensures the file size is less on the local storage and network.

Delete the auto generated default spacer as shown below and set the form size to what we need.

```
<spacer sid="vfd_spacer">
```

```
<itemlocation>
    <x>960</x>
    <y>1260</y>
    <width>1</width>
    <height>1</height>
</itemlocation>


</spacer>
```

Use compressed image formats such as PNG or JPG. This minimizes the form size thus reducing load times. The following compressed image formats are supported:

GIF — GIF image support is limited to images that do not contain animation, transparency, or interlacing.

JPG — some types of progressive JPG images are not supported.

PNG

**NOTE** The BMP image format is not to be used.

### Performance

By default the **<pageloading>** setting is enabled. This determines which pages are loaded into memory when a form is opened. Subsequent pages are loaded into memory as required. For example, when dealing with large forms with multiple pages, this functionality only loads one page at a time instead rendering the entire form. This will significantly reduce the amount of time to present a large form initially. However, it may slightly increase the amount of time required to switch to a new page. Do not disable this setting.

### Presentation

Show only tool bar items that are required. If the tool bar is not required, then the tool bar should be disabled using the following code:

```
<ufv_settings>
     <menu>
        <visible>off</visible>
```

```
    </menu>

</ufv_settings>
```

To display a label with different formats and limited small text labels, use Text Label rather than a Rich Text Label. This will create a simple label item with no data binding.

Always use relative alignment between two labels on the form. Labels are stored with X, Y coordinate properties. These coordinates must be eliminated and the label must be aligned relative to the previous label as required. For example:

| Code Before Alignment | Code after Alignment |
|---|---|
| `<itemlocation>`<br>`<x>44</x>`<br>`<y>44</y>`<br>`</itemlocation>` | `<itemlocation>`<br>`<below>firstName</below>`<br>`</itemlocation>` |
| Where firstName is the item **sid** | |

**NOTE**

1. Delete the duplicate alignments from the items.
2. Using the mouse to drag and place items will generate XY coordinates. Use the arrow keys to position items.

**Panes**

It is not possible to access fields inside a pane from a JSP to update the fields. So always create a hidden field outside the pane and bind the nodeset to the same node as that of the field inside the pane and update this field instead.

Avoid using inner panes (i.e., a pane within a pane).

**Attachments**

There are two ways an attachment may be added to the form. They are:

Xforms — upload

XFDL Attachment Button

Always use the XFDL Attachment Button. Properties such as size, type, and number of attachments are controlled better with this.

# 1.5.3.4 API Standards

Two varieties of APIs are available to handle the submissions from the form. They are:

- Classic API (Native API)
- Streaming API

Depending on the requirement of the consuming Application, one should plan to select the API. Most applications use Streaming API when no relevance is set up on the form and X-Forms Submission is being used. Use Classic API (Native API) when XFDL Submission is being utilized and relevance is subject to change based on user selection on the form.

When declaring the formNodeP, always use a singleton of the formnodeP so no other thread will update the formnode simultaneously. See below:

```
import com.PureEdge.IFSSingleton;

XFDL theXFDL = IFSSingleton.getXFDL();
```

Regardless of which API is being used, always destroy the formNodeP that is being generated to modify the contents.

Use updateXFormsInstance only once. Complete all modifications to the instance then update the instance once.

Always close a fileoutputstream when it is used. See example below:

```
FileOutputStream fos = new FileOutputStream(attachFile);
                       formDataBean.setIsAttachment(true);
                       formDataBean.setFileName(attachFile);
                       fos.write(attachData);
                       fos.close();
```

# 1.5.3.5 Programming Guidelines

## Application Guidelines

1. Configuration must never be contained within the design of an application. Applications must be configurable with data within the application. Configuration data must always be part of ini files, properties files, or configuration data within a database.
2. Separation of data from design. The structure of the data must be separate from the presentation of the data.
3. Data flow must not be automated between environments.

4. Design moves between environments via template refreshes. If a design element needs to be added or removed to/from the next environment, a new design template must be created on the lower environment. Example: Do not request that the form 'ABC' be removed on a move request to system.
5. Replication formulas are not to be used.
6. Group names only in a database's ACL. Individual users must never be listed in ACL's
7. Any document must be able to be modified from a browser. Client access must not be required for any document changes.
8. Any data entry forms must have merge conflicts enabled on the form properties.
9. Computed text must be used versus computed for display fields whenever possible for HTML and CSS.
10. User name and passwords must be stored in secure configuration documents.
11. All site and content management must be able to be performed over a browser.
12. Revision history entries, as well as other items that will be used for tracking purposes, must be contained in separate documents if items are not removed after 10 entries.
13. If at all possible, the content manager interface must be based on the standard look and feel. This must include following LCAS standard user interface error handling.
14. Master/Working logic (versioning) must be able to be easily implemented in an application.
15. Application design must consist of two databases if possible – one main database, and one archive database.
16. Only active design elements must exist in an application. Do not use the "Prohibit design refresh or replace to modify" setting on the form design tab.
17. JavaScript functions and variables must be placed within JavaScript libraries or pages (if computed text is needed). Functions can be called from either pass-through HTML or Domino form events.
18. Option Declare or Option Explicit must always be used in agents.

## Form and Page Formatting

Web formatting of forms and pages must always be done with pass-through HTML via CSS styles. Any style used more than once must be in a class in a CSS page. Style definitions must be semantic in nature, and not presentational. (e.g.: "Navigation" or "Content", not "LeftNav" or "RedText"). Names must not include actual attributes in the name (i.e., style100px). Also note that styles must be presented in pixels, not points. Developers must ensure that there is not an existing style that already meets their needs. Domino formatting (i.e. fonts, tables, etc.) must be done explicitly with HTML. Use of Domino tables and other Domino specific Web objects must not be used, with the exception of the file upload control.

Form and page content sections must be easily identifiable. The different sections include default HTML, computed text, hidden fields, and comments.

### Default HTML

8pt. or 10pt., Sans Serif, Black

### Computed Text

8pt. or 10pt., Sans Serif, Blue

### Hidden Fields

These fields must be grouped at the top or bottom of a page.

8pt. or 10pt., Sans Serif, Red

### Comments

8pt. or 10pt., Sans Serif, Dark Green

## Subforms

Subforms must always be commented. For computed subforms, comment blocks must be placed around the 'Computed Subform' area. Inline subforms must have comments surrounding the actual subform content. Subforms must be used where you can use the same code in multiple places. They must not be used where they will not be used only in one location, and there is no plan to use this code in future functionality.

Field Formatting

Field titles must have a specific syntax. Field titles must be fully spelled out. No abbreviations must be used. In addition, each word in the field name must be proper-cased.

## LotusScript Variables

### Variable Name

Variable names will use the Hungarian Notation standard. This denotes that a variable name will begin with an abbreviation of its type.

For example, a String that denotes a color would be titled:

**Correct:**

strColor

**Incorrect:**

color

### Variable Length

Variable names must be no longer than 32 characters in length. Variable names must contain no abbreviations. Abbreviations must only be used if the fully spelled-out version would exceed 32 characters. Then only the final word(s) must be abbreviated. For example if you wanted to describes a dog's color you might name your variable as follows:

**Correct:**

strDogColor

**Incorrect:**

strDgClr

## Global Variables

Global variable usage (Table 1-7 and Table 1-8) must be minimized. If a global value is judged technically needed, the variable must be prefixed with "g_". In addition, Hungarian notation will still apply.

For example: g_strColor

### LotusScript Global Variables

| LotusScript Variable | Prefix |
|---|---|
| String | str |
| Variant | var |
| Double | dbl |
| Long | lng |
| Integer | int or just i, k, j, l |
| Boolean | bool |
| Single | sng |
| Byte | byte |
| Currency | cur |
| **Domino Object** | **Prefix** |
| Form | frm |
| View | vw |
| Page | pg |

**Table 1-7** LotusScript Global Variables

### Domino Object Global Variables

| Domino Object | Prefix |
|---|---|
| NotesSession | sess |
| NotesDatabase | db |
| NotesDocument | doc |
| NotesDocumentCollection | dc |
| NotesItem | itm |

**Domino Object Global Variables**

| Domino Object | Prefix |
|---|---|
| NotesRichTextItem | rti |
| NotesView | view |
| NotesViewColumn | vcol |
| NotesViewEntry | vent |
| NotesViewEntryCollection | ventc |
| NotesACL | acl |
| NotesACLEntry | aclent |
| NotesAgent | agt |
| NotesDateTime | date |
| NotesEmbeddedObject | emb |
| NotesLog | log |
| NotesName | name |

**Table 1-8** Domino Object Global Variables

# 1.5.4 JavaScript

## 1.5.4.1 General Discussion

### Unobtrusive JavaScript

Whenever possible, JavaScript in web documents should be written in an unobtrusive fashion. *Unobtrusive JavaScript* refers to the programming practice of separating JavaScript functionality from the markup and presentation of a web page. This programming practice:

- Separates the script from the HTML allowing the web page to degrade gracefully.
- Reduces repetition in code.
- Adheres to web and accessibility standards.
- Takes different browsers, devices, and compatibility issues into account.
- Allows you to work on the JavaScript without having to touch the HTML or CSS.

Similar to CSS, externalizing the JavaScript from the HTML is achieved by referencing the script via the <script> tag.

**Example:** `<script type="text/javascript" src="my-script.js"></script>`

Unobtrusive JavaScript is obtained through using an HTML element's tag, class, ID, or a combination of CSS selectors to attach events to it.

**Traditional JavaScript Example:**

HTML

```
<a href="http://www.google.com/" class="google-link">Go to Google!</a>
```

JavaScript

```
window.onload = function() {
      var myLink = document.getElementsByClassName('google-link');
      myLink[0].onclick = function() {
            window.location = 'http://www.bing.com ?q=bing+is+evil';
            return false;
      }
}
```

**Dojo 1.10+ Example:**

HTML

```
<a href="http://www.google.com" class="google-link">Go to Google!</a>
```

JavaScript

```
require(
    [‘dojo/query’, ‘dojo/on’, ‘dojo/_base/event’, ‘dojo/domReady!’],
    function(query, on, event){
        var myLink = query(‘.google-link’);
        on(myLink, ‘click’, function(evt) {
            event.stop(evt);
            window.location = ‘https://www.google.com’;
```

```
        });

    }

);
```

## 1.5.4.2 File Names

This section lists commonly used file suffixes and names.

### File Suffixes

JavaScript programs generally will be stored in and delivered as .js files, and stand-alone JavaScript Object Notation (JSON) will be stored and delivered as .json files. Depending on the JavaScript framework, however, other file extensions may be relevant. For example, when using the Vue framework, single-file components are delivered as .vue files. In these cases, framework specific conventions are acceptable.

## 1.5.4.3 File Organization

For traditional JavaScript development, do not embed JavaScript code in HTML files unless the code is specific to a single session. Code in HTML adds significantly to page weight with no opportunity for mitigation by caching and compression. Place `<script src=filename.js>` tags as late in the body as possible. This reduces the effects of delays imposed by script loading on other page components. There is no need to use the language or type attributes. It is the server, not the script tag, which determines the MIME type.

With modern JavaScript development, especially with Single-Page Applications (SPA), JavaScript and template HTML may be intermingled depending on the framework's conventions. Module bundlers and JavaScript build optimizers are responsible for reducing the load associated with this approach, and should be used for production level code.

## 1.5.4.4 Naming

Names must be formed from the 26 upper and lower case letters (`A` .. `Z`, `a` .. `z`), the 10 digits (`0` .. `9`), and `_` (underbar). Avoid use of international characters because they may not read well or be understood everywhere. Do not use `$` (dollar sign) or `\` (backslash) in names. Do not use `_` (underbar) as the first character of a name. It is sometimes used to indicate privacy, but it does not actually provide privacy. If privacy is important, use the forms that provide private members. Avoid conventions that demonstrate a lack of competence. Start variables and functions with a lower case letter. Constructor functions, which must be used with the new prefix, must start with a capital letter. JavaScript issues neither a compile-time warning nor a run-time warning if a required new is omitted. Bad things can happen if new is not used, so the capitalization convention is the only defense we have. Global variables must be in all caps. (JavaScript does not have macros or constants, so there isn't much point in using all caps to signify features that JavaScript doesn't have.)

### General Naming Conventions

- When constructing string IDs or ID prefixes in the code, do not use *dojo*, *dijit* or *dojox* in the names. Because we now allow multiple versions of dojo in a page, it is important to use _scopeName instead (dojo._scopeName, dijit._scopeName, dojox._scopeName).
- Write names representing modules all lower case. (MANDATORY)
- Make names representing types (classes) nouns and written using CamelCase capitalization (e.g., `Account`, `EventHandler`). (MANDATORY)
- Place constants within a single object created as a holder for constants, emulating an Enum. Name the Enum appropriately, and name the members using either CamelCase or UPPER_CASE capitalization.

```
var NodeTypes = {

    Element: 1,

    DOCUMENT: 2

};
```

- Do not use UPPERCASE for abbreviations and acronyms that are used as a name: `getInnerHTML()`, `getXML()`, `XmlDocument`.
- Make names representing methods verbs or verb phrases: `obj.getSomeValue()`. (MANDATORY)
- Write public class variables using mixedCase capitalization. (MANDATORY)
- Ensure that CSS variable names follow the same conventions as public class variables. (MANDATORY)
- Write private class variables using mixedCase.

```
var MyClass = function(){

    var _buffer;

    this.doSomething = function(){}

}
```

- Give generic variables the same name as their type: `setTopic(topic) //` where topic is of type Topic. (MANDATORY)
- Write all the names in English.
- Use globally unambiguous names for variables with a large scope; ambiguity MAY be distinguished by module membership. Variables with small or private scope MAY have terse names.
- As the name of the return object is implicit, avoid using it in a method name. (MANDATORY)

```
getHandler(); // NOT getEventHandler()``
```

- Make public names as clear as necessary and avoid unclear shortenings and contractions. (MANDATORY)

```
MouseEventHandler // not MseEvtHdlr
```

- Note that, again, any context that can be determined by module membership MUST be used when determining if a variable name is clear. For example, a class that represents a mouse event handler.

```
dojo.event.mouse.Handler; // NOT dojo.events.mouse.MouseEventhandler
```

- Name classes/constructors based on their inheritance pattern, with the base class to the right of the name.

```
UiEventHandler

MouseEventHandler
```

- Drop the base class from a name if it is obviously implicit in the name.

```
MouseEventHandler; // as opposed to MouseUIEventHandler
```

- Name after nouns those functions that act as both getters and setters depending on the number of arguments. The '*get*' and '*set*' are implied. For example:

```
dojo.attr(node, "tabIndex"); // getter

dojo.attr(node, "tabIndex", -1); // setter
```

## Specific Naming Conventions

- Do not use the terms get/set where a field is accessed, unless the variable being accessed is lexically private.
- Use the `is` prefix for boolean variables and methods. Alternatives include `has`, `can` and `should`.
- Use the term `compute` in methods where something is computed.
- Use the term `find` in methods where something is looked up.
- Use the terms `initialize` or `init` where an object or a concept is established.
- Use `control type` as a suffix for UI Control variables.
    - Examples: `leftComboBox, topScrollPane`
- Use the plural form to name collections. (MANDATORY)
- Use a `num` prefix or `count` postfix for variables representing a number of objects.
- Call iterator variables `i`, `j`, `k`, etc. (MANDATORY)
- Use complement names for complement entities. Examples: get/set, add/remove, create/destroy, start/stop, insert/delete, begin/end, etc. (MANDATORY)
- Avoid abbreviations in names.
- Avoid negated Boolean variable names, `isNotError` and `isNotFound` are unacceptable.
- Name methods that return an object similarly to what they return. For example, a method returning a patient object may be named `getPatient()`.
- Name methods that return void similarly to what they do. For example, a method that changes the status of a flag not returning anything may be named `toggleFlag`.

## 1.5.4.5 Linting Standards

Linting is a form of static code analysis that alerts the user to potential errors or bugs as well as enforcing stylistic preferences. These tools help ensure consistency within the codebase and organization.

There are multiple tools that are available for linting. Each one is aimed at a specific technology or framework. The linting rulesets are maintained in GitHub and are approved by the Application Technical Standards Subcommittee. The current linting standards for JavaScript can be found here:

JavaScript Linting Standards

## 1.5.4.6 Miscellaneous

Do not use magic numbers in the code; declare them using named constants instead.

Always write floating point constants with a decimal point, at least one digit before the decimal point, and at least one decimal value.

## 1.5.4.7 Avoid Synchronous Ajax Calls

Using synchronous requests will cause the browser to lock up. If it takes some time for the server to respond, the user's browser will not allow anything else to be done until it receives a response. If a response is never received, the browser may continue to block until a time-out occurs. Use asynchronous mode.

## 1.5.4.8 Keep Document Object Model Access to a Minimum

Accessing the Document Object Model (DOM) is an expensive operation. Rendering in browsers can take much time. This can result in changes taking longer than usual or rendering halfway through. To create fast code, keep DOM access to a minimum. Instead of creating and applying elements, have a function that turns a string into DOM elements and execute it at the end of the generation process to disturb the browser rendering only once. If a large section of DOM elements do need to be generated, use a DocumentFragment object as a temporary container for the generated elements, and then insert the entire fragment into the DOM.

These guidelines are not the case for widgets that encapsulate the state of the DOM and manipulate small sections of DOM nodes as a core competency.

## 1.5.4.9 Browser Compatibility

Code written to process on a specific browser is a certain way to keep the code difficult to maintain and dated. Write code according to agreed standards. If there has to be a segment of code written for a specific browser, place it in its own script document and name it with the browser and version that the code is expected to run on. This will allow the script to be removed if the browser version becomes obsolete. Use feature detection whenever possible instead of browser detection when forking logic for functionality on browser requirements. This will keep the code forward compatible. Polyfills are also a great option for using newer JavaScript features while having to maintain legacy browser support.

## 1.5.4.10 Robustness

Check that all data being used is what it is expected to be. Use the `typeof` keyword to check all data. Handle unexpected data by notifying the user that an error has occurred or handle it using the code logic.

For example, to validate an Array called members, the following code might be used:

```
if(typeof members === 'object' && typeof members.slice === 'function'){...}
```

Arrays returen a type of `'object'`. Slice is a method that is specific to arrays, so it helps ensure that the variable is not just a different type of object.

When manipulating DOM elements, check that the element is what you expect it to be.

## 1.5.4.11 Maintainability over Optimization

Do not comment complicated code, rewrite it.

There are many things that can be done to increase the performance of JavaScript code, but there usually is a trade-off with readability. Don't write for performance, write for readability. A build script or tool can optimize the code. Focus on maintainability and writing code for the next developer.

An example of a supported code minifier is Dojo Layers. However, minification is not required for JavaScript development.

## 1.5.4.12 API Documentation

JavaScript code should be documented inline the same way that Java code is documented with JavaDocs. For JavaScript, there are several frameworks that support inline documentation. A specific documentation library has not been mandated yet, but the three most typically used are:

- Dojo API (generally only used to document Dojo code)
- JSDoc
- YUIDoc

## 1.5.4.13 Frameworks

### Approved Frameworks and Toolkits

There are currently no JavaScript frameworks or toolkits that are *approved* from a third-party support contract perspective. The following frameworks are considered the current development choices for new work. If one of these does not fit the needs of a particular effort, a technical advisor should be consulted on options.

- Dojo Toolkit (AMD)
- Vue.js
- Angular.js
- TypeScript

### Legacy Frameworks

There are many legacy applications that are written using frameworks that are not considered modern development frameworks (examples may include applications using legacy frameworks). These applications will continue to be supported, but the application team must work to bring these applications up to date with a modern framework.

Some of these legacy frameworks may include:

- Dojo Toolkit (pre-AMD)
- Yahoo User Interface Library (YUI)
- JQuery
- PrototypeJS

## 1.5.4.14 Validation

Validate all JavaScript. JavaScript code should be written in a manner that promotes Unit validation. The frameworks listed below are available for validating and are currently in use today. It is recommended that you use the one that best suits the framework that you are developing against.

- Karma/Jasmine
- Mocha
- Intern

The Dojo Objective Harness framework (DOH) has been a popular choice for JavaScript validation as it was bundled with the Dojo Toolkit. DOH has been deprecated by SitePen in favor of their new validation framework, Intern. Furthermore, it is not easily automatable in a build process. For these reasons, DOH is no longer recommended as a validation framework for new development. It is also recommended that existing DOH validation be migrated to one of the newer validation frameworks listed above.

# 1.5.5 Visual Basic

Effective September 2007, the following standards apply to all new development and will be retrofitted into any program that is undergoing a major revision. The standards adaptation applies to the entire source file, not just the change. (A source file is considered to include any valid Visual Basic (VB) 6.0 file extension capable of containing source, designer, or resource definitions.).

An exception to these standards is granted in situations where VB code has been obtained from a vendor at a source level and no significant modifications have been made to it.

This section specifies the standards and guidelines for the development of applications using the following VB products:

- VB 6
- Rumba VB Script
- VBS Script (.vbs files)
- VB.net (Object Oriented)
- VBA 32-bit
- VBA 64-bit

The following information applies to all the VB versions listed above unless otherwise specified. Specific sections will detail specific exceptions.

## 1.5.5.1 Option Explicit

All modules must have `Option Explicit` to enforce explicit variable declarations (e.g., `DIM`, `Private`, `Public`, and `Local`).

The `Option Explicit` statement must be used at the ehad of all source files (e.g., UserForms, Modules, Class Modules, etc.).

Use of the `Option Explicit` statement helps to avoid bugs caused by misspelling of variable names. The absence of this statement allows variables within the source file to be used without being declared, which may lead to unexpected bugs that are difficult to identify. It also causes unnecessary conversations between variant and other types that could lead to unexpected results involving logical operations.

**NOTE** The VB Integrated Development Environment (IDE) can be configured to automatically generate the `Option Explicit` command at the head of each source file by checking the *Require Variable Declarations* checkbox on the *Editor* tab of the Tools Menu Option.

## 1.5.5.2 Declare Statements

The standards below are related to the use of `Declare` statements to call functions in external components.

`Declare` statements used to call functions in external components must ensure the correct use of `ByVal` as well as the correct choice of data types. If necessary, also ensure that string parameters are pre-populated to the correct lengths and structure elements are correctly aligned on memory boundaries.

Return codes must also be checked and handled appropriately.

### Declare Statement Must Have Private Scope

`Declare` statements must always be declared with `Private` scope and the functionality exposed through a wrapper function, which may have `Public` scope if necessary.

### Declare Statements Must Not Use Ordinal Numbers

`Declare` statements must always bind to function names. The `Alias` keyword must not be used to bind to the ordinal number of a function's entry point in an external component.

### Declare Arguments Must Not Use Data Type As Any

The data type `As Any` must not be used when declaring arguments within `Declare` statements as this can lead to spurious errors when incorrectly used. Always take time to determine and understand the exact stack signature of the function that you wish to invoke in the external component. If an external

function supports use of different parameter types, provide `Aliased`-type, safe versions of the function, each using the correct parameter types.

### Declare Statements Must Not Contain Hard-Coded Paths to Components

The `Lib` parameter of declare statements must define only the component file name excluding any directory path.

## 1.5.5.3 Constant Declarations

### Public Constant Names Must Be Correctly Formed

All public constant names must take the form `g<data type prefix><body>`. The data type prefix must be the appropriate Hungarian notation for the data type being declared. The body of public constant names must always be upper case and underscore characters may be included to aid readability. Public constant names must not use VB data type suffix characters but must instead use the `As` keyword to specifically state the constant data type. The name chosen must clearly make obvious the intended use of the constant.

### Local Constant Names Must Be Correctly Formed

All local constant names must take the form `l<data type prefix><body>`. The data type prefix must be the appropriate Hungarian notation for the data type being declared. The body of local constant names must always be upper case and underscore characters may be included to aid readability. Local constant names must not use VB data type suffix characters but must instead use the `As` keyword to specifically state the constant data type. The name chosen must clearly make obvious the intended use of the constant.

### Private Module-Level Constant Names Must Be Correctly Formed

All private module-level constant names must take the form `m<data type prefix><body>`. The data type prefix must be the appropriate Hungarian notation for the data type being declared. Private module-level constant names must always be upper case and underscore characters may be included to aid readability. Private module-level constant names must not use VB data type suffix characters but must instead use the `As` keyword to specifically state the constant data type. The name chosen must clearly make obvious the intended use of the constant.

### Hungarian Notation Variable Type Prefixes

**Hungarian Notation Variable Type Prefixes**

| Prefix | Data Type |
|---|---|
| bool | Boolean |
| byt | Byte |
| cur | Currency |

**Hungarian Notation Variable Type Prefixes**

| Prefix | Data Type |
|---|---|
| int | Integer |
| dbl | Double |
| dt | Date and Time |
| lng | Long |
| sgl | Single |
| str | String |
| var | Variant |
| arr | Array |
| ut | User-Defined Type |
| obj | Object |

**Table 1-9** Hungarian Notation Variable Type Prefixes

### Constant Data Type Must Be Explicitly Specified Using As Keyword

The `As` keyword must be used within a constant declaration to explicitly state the data type of the declared constant.

### Constant Scope Must Always Be Specified

The scope for a module-level `Const` declaration must always be explicitly declared `Public` or `Private` as appropriate.

### Multiple Item Const Declarations Must Not Be Used

To avoid confusion and aid readability, individual constants must be declared using separate `Const` statements.

## 1.5.5.4 Enum Declarations

The following standards are related to the declaration of `Enum` (including its elements).

### Enum Scope Must Always Be Specified

The scope for an `Enum` declaration must always be explicitly declared `Public` or `Private` as appropriate.

### Enum Elements Must Always Specify a Value

To aid clarity and debugging, `Enum` elements must always specify a value.

An `Enum` statement can appear only at the module level. Once the `Enum` type is defined, it can be used to declare variables, parameters, or procedures returning its type. An `Enum`-type name can't be qualified with a module name. `Public Enum` types in a class module are not members of the class; however, they are written to the type library. `Enum` types defined in standard modules aren't written to type libraries. `Public Enum` types of the same name can't be defined in both standard modules and class modules since they share the same name space. When two `Enum` types in different type libraries have the same name, but different elements, a reference to a variable of the type depends on which type library has higher priority in the References. You can't use an `Enum` type as the target in a `With` block.

### Enum Element Names Must All Share the Same Prefix

To aid clarity of usage, the names of `Enum` elements for an individual `Enum` definition must all begin with the same prefix. The prefix chosen should be at least two (2) characters.

This is because `Enum` are typically Public structures, which means the enumerated constants within the block must have unique names in their scope. An example of the required prefix is shown below.

```
Public Enum OrderStatusConstants

      osInProcess = 1

      osShipped = 2

      osBackOrdered = 3

End Enum
```

### Enum Element Values Must Be Numeric

Despite the VB compiler allowing string literals to be defined for `Enum` element values, `Enum` element values must always be numeric. For example: eMyValue = "55" is not allowed and should instead be defined as eMyValue = 55.

## 1.5.5.5 Variable Declarations

### ReDim Usage

`ReDim` must not be used for variable declaration.

The `ReDim` statement must not be used to initially dimension (declare) an array, instead it should only be used to re-dimension an existing array.

## Keyword Dim Usage

`Dim` must not be used to define module-level variables.

Module-level variables must always be defined using the `Private` keyword.

## Keyword As Usage

Variable data types must be explicitly defined by using the `As` keyword.

All variables defined in a VB 6.0 project must explicitly be defined using the `As` keyword to prevent a default type of `Variant` from being created.

## Multiple Item Variable Declarations

Multiple Item Variable declarations must not be used.

To aid readability, each variable defined within a VB 6.0 project must be defined on a separate line.

## Public Variables in Class Modules

`Public` variables must not be used in class modules.

The `Public` keyword must not be used in classes but instead the `Private` keyword should be used. `Private` variables should be accessed externally via property procedures.

## Keyword New Usage

Variables must not be declared as `New`.

Auto instantiating variable declarations using the `New` keyword in the declaration must not be used. Instead, all object variables must be instantiated using the `= New` or `=CreateObject` syntax

## Keyword To Usage

The `To` keyword must be used for array variable declarations.

When declaring an array variable, both the lower and upper bounds must be specified using the `To` keyword. Also, the lower bound must always be zero.

## Member Object Variable Life Span

Member object variables must be explicitly destroyed.

Object variables created in a procedure must be explicitly set to `Nothing` at the end of the procedure (e.g., any object that was created with the `Set` keyword).

## Variable Names

All variable names must take the form `<scope prefix><array prefix><data type prefix><body>`. The scope prefix must be blank in the case of local variables, `g_` for global, `st_` for local static, or `m_` for module level. The array prefix must be `arr` when an array is being declared. The data type prefix must be the appropriate Hungarian notation for the data type being declared. The body of a variable name must employ mixed case, exclude underscore characters and be at least one character long. Variable names must not use VB data type suffixes but must instead use the `As` keyword to specifically state the data type. The name used must make it as clear as possible the intended use of the variable.

# 1.5.5.6 Source Files

## Class File Name Prefix

Class file names must be prefix.

VB 6.0 class file names must be prefixed `cls`.

## Form File Extension

Form File names must use an extension `frm`.

VB form files must have an extension `frm`.

## BAS Module Extension

Module files must use an extension `bas`.

VB module files must have an extension `bas`.

# 1.5.5.7 Error Handling

## Err Object

Use `Err` object to determine the error that was received during processing. For example, use the `Err.Description` to obtain the description of the error.

## Use of Stop Statements

The `Stop` statement should never be used as it may be accidentally left within source code and cause the compiled application to terminate abruptly. The `End` statement should be used instead if the application needs to terminate. In place of the `Stop` statement, use `Debug.Assert False` to highlight a problem during debugging.

An exception for this standard exists when a third-party vendor has given specific instruction that this statement be left in their code.

## Use of On Local Error

Use of `On Local Error` must not be used. Use of this syntax for error handling declarations is provided in VB for backward compatibility with previous versions. To keep code consistent, omit the `Local` keyword.

### Use of Error Handlers

All Subs and Functions must contain an error handler. Simple property procedures used to access native member variables and empty procedures within interface class definitions need not contain an error handler.

### Use On Error Resume Next

Limit the use of `On Error Resume Next` for validation purposes, during the development phase, or where it will not cause any errors. Simply ignoring errors and not handling them properly is bad coding practice and could result in additional errors being ignored that could be damaging to the code. That said, all `Class_Terminate` events must contain the `On Error Resume Next` as the first statement to ensure that errors are correctly handled.

### Error Handlers Must Be Isolated

Error handling code blocks must be isolated from the rest of the code by ensuring that they are preceded with an `Exit` statement so that normal code execution does not accidentally fall through into the error handler code. Error handlers must be located at the end of a procedure.

### Error Handlers Must Be Valid

It is not enough to simply add an `On Error GoTo <LineLabel>` statement to a procedure with an empty error handling block. Error handlers must do something meaningful such as report or log the error, raise the error, or try to recover from it.

### Use of Resume

Ensure the proper use of `Resume` statements are in place as appropriate. `Resume` statements should be used after error handling instead of `Goto`. `Resume` will clear the `Err` object and switches the error handler back on to ensure that the previous error handler statement will be enforced in the event of another error.

## 1.5.5.8 Code Layout

To reduce maintenance and future development costs, it is essential that source code layout and formatting is consistent, making code easy to read and understand.

### Declarations

All declarations must be made at the start of all Classes, Modules, Forms, Subs and Functions. Please refer to the section *Procedure Declarations* below.

### Empty Procedures Must Be Removed

Empty procedure declarations must not be left in source files. First, remove references to the obsolete procedure and then remove the procedure itself. Only classes used to define COM interfaces may contain empty procedures. (Interface classes are denoted by names beginning with `I`.)

### For...Next Counter Variable Must Not Be Omitted

To aid source code readability, all `For...Next` loops must include the name of the loop counter variable after the `Next` keyword.

### Individual Property Get/Let/Set Procedures Must Be Grouped Together

The property procedures for an individual property with more than one access method (`Get`, `Let` or `Set`) must not be separated by other procedures but should always be grouped together. The grouping can appear above or below other source code.

### Use a Single Statement Per Line

All VB statements should be clearly arranged on a separate line. Concatenation of multiple statements per line using the `:` separator must not be used.

### Use of Error Handlers

All Classes, Modules, Forms, Subs and Functions must contain an error handler. Please refer to the section *Error Handling* above.

### Use of Comments

All Classes, Modules, Forms, Subs and Functions must contain comments. Please refer to the section *Use of Comments* below.

## 1.5.5.9 Procedure Declarations

### Procedures Must Not Return Array Data Type

Procedures must not return arrays directly as return data types. If a procedure must return an array, it should do so via a `ByRef` parameter. Procedures returning arrays directly can fall victim to memory leaks.

### Function Procedure Return Type Must Be Explicitly Declared

Always specify the actual data type returned by function procedures. The exception to this rule applies when calling a function through an Access data macro.

The return data type of a function procedure must be explicitly defined using the `As` keyword and must not use VB data type suffix characters.

Not doing so leads to confusion later as it is not immediately obvious whether the author intended that the function procedure should return a variant data type.

### All Function Procedures Must Assign a Return Value

All function procedures must explicitly assign a return value. The exception to this rule applies when calling a function through an Access data macro.

### All Procedures Must Specify Procedure Scope

To avoid any confusion, all procedures must be explicitly scoped using the `Public`, `Private` or `Friend` keyword as appropriate.

### Calling Convention for Argument Must Be Explicitly Defined

The calling convention for all procedure arguments must be explicitly specified using the appropriate `ByVal` or `ByRef` keyword for the argument declaration.

### Argument Data Type Must Be Specified

The data type for all arguments must be explicitly specified using the `As` keyword and the appropriate data type. The exception to this rule applies to VB Script as it does not require data types.

### Procedure Declarations Must Not Include the Static Keyword

Specifying the `Static` keyword on a procedure declaration means that all local variables are treated as static variables whether or not they are explicitly declared as such.

If all local variables in a procedure need to be declared `Static`, then the `Static` scope should be changed to `Public` or `Private` as appropriate, and those variables that should be static should be explicitly declared so.

### Procedure Must Not Use ParamArray Argument

The `ParamArray` keyword allows you to provide an arbitrary number of variant arguments in an argument list without having to explicitly list the number or type of each. Since variant types are not allowed and each argument must be listed, usage of the `ParamArray` argument is prohibited.

## 1.5.5.10 Performance

### + Operator Must Not Be Used for String Concatenation

The `+` operator must not be used for string concatenation. The `+` operator should be reserved for arithmetic operations. Use of this operator for concatenation may lead to performance degradation or unexpected results if a variant variable is involved. The ampersand `&` should be used for concatenation operations.

### GoSub Must Not Be Used for Subroutine Calls

A project compiled in native code shows poor performance if the project contains calls to subroutines using the `GoSub` statement.

Convert the internal subroutine to a regular `Sub` and replace the `GoSub` call with a call to the `Sub` instead.

## Variant Trim Function Must Not Be Used

Variant `LTrim()`, `RTrim()` or `Trim()` functions must not be used due to slower performance to equivalent string functions. `LTrim$()`, `RTrim$()` or `Trim$()` functions must always be used.

## Variant String Manipulation Function Must Not Be Used

Variant `Left()`,`Right()`or `Mid()`function must not be used due to slower performance to equivalent string functions. `Left$()`, `Right$()` or `Mid$()` must always be used.

## Variant Character Function Must Not Be Used

Variant `Chr()`function must not be used due to slower performance to equivalent string function. `Chr$()` must always be used.

## Chr$() Expression Must Not Be Used To Convert VB Intrinsic Constant

VB intrinsic constants must be used instead of the `Chr$()` expression.

Intrinsic constants can be found by opening the VB Object Browser and selecting the *VBA* library and clicking on the *Constants* class to obtain a list of the intrinsic constants that may be used instead of `Chr$()` expressions to improve performance.

## Variant Hex() Function Must Not Be Used

Variant `Hex()` function must not be used due to slower performance to the equivalent string function. `Hex$()` must always be used.

## Variant Space() Function Must Not Be Used

Variant `Space()` function must not be used due to slower performance to the equivalent string function. `Space$()` should always be used.

## Variant Directory Function Must Not Be Used

Variant `CurDir()` or `Dir()` function must not be used due to slower performance to the equivalent string function. `CurDir$()` or `Dir$()` should always be used.

## Optional Argument Must Not Be Declared As Variant

Optional arguments must not be declared `As Variant`. Doing so will reduce type safety and add additional processing overhead, which can degrade operational performance.

Change the argument declaration to the correct data type for its intended use.

### For Loop Statement Must Not Contain UBound/LBound Function

The `For` statement must not use `LBound()` and/or `UBound()`. Doing so will add additional overhead to the loop operation, which may degrade software operation.

Assign the result of the `LBound()` and/or `UBound()` functions to a temporary variable prior to the `For...Next` loop and replace the `LBound()` and/or `UBound()` function with the appropriate temporary variable.

### Null String Comparison Must Not Be Used

`If...Then` statements must not perform a direct null string comparison (`= "" or = vbNullString`) due to degraded operational performance.

Use the VB `Len()` function to determine that the string length is zero instead of comparing the string with "" or `vbNullString`.

### For Loop Statement Must Not Reference Object Count Property

The `For` statement must not access an object's `Count` property. Doing so will add additional overhead to the loop operation, which may degrade software operation.

Assign the value of the object `Count` property to a temporary variable prior to the `For...Next` loop and access the temporary variable instead.

## 1.5.5.11 Operator Usage

### ! Operator Must Not Be Used to Access Collections

For consistency, the NOT operator (`!`) must not be used to access collection or control members, and the dot (`.`) operator must always be used instead.

### InStr() Function Must Not Be Embedded Within a Mid() Operation

Use of the `InStr()` function must be performed as a separate statement and must not be embedded within a `Mid()` function call as a zero return value from the `InStr()` could lead to a Run-time error '5': Invalid procedure call or argument.

## 1.5.5.12 User-Defined Types

### All User-Defined Type Names Must End With Suffix Type

To differentiate a user-defined type definition from an instance of a user-defined type, all user-defined type names must end with the suffix `Type`.

### User-Defined Type Naming

All elements of a user-defined type must take the form `udt<body>`. The body of an element name must employ mixed case, exclude underscore characters and be at least one character long. Element names must not use VB data type suffix characters but must instead use the `As` keyword to specifically state the element data type.

## 1.5.5.13 Code Complexity

To reduce maintenance and future development costs, it is essential that source code complexity is kept to a minimum, thereby ensuring code is easy to understand, modify and validate. Add comments around unavoidably complex code to facilitate further maintenance.

### For...Next Counter Variable Must Not Be Omitted

To aid source code readability, all `For...Next` loops must include the name of the loop counter variable after the `Next` keyword.

### Application Startup Object Must Be Sub Main

The application startup object must always be `Sub Main`.

## 1.5.5.14 Use of Comments

### Procedures Must Include a Comment Header

All procedures must include a standard comment header describing what the procedure does as well as details of any arguments used and return values. A flower box is not necessary, but if there are several points to make, it is acceptable to use a line of (hyphens, equal signs, and asterisks) to break up the points.

### Source/Module/Class/Object Files Must Include Header Comments

All source/module/class/object files must begin with a standard header comment block describing the role of the file. Refer to the section *Procedures Must Include a Comment Header* above about options for readability. The header comments should include the author of the code, what it does, any limitations, and amendment history. This would be at the top of the file.

### Use Comments For All Non-Obvious Actions

All code where it is not clear what action is happening should be explained in detail. Use the comment to describe what is happening, what will happen next, why this option was chosen and any external factors that need to be known.

### In-Line Comments

In-line comments should be directly above the line of code/block of code the comment is explaining. The comment should line up with the code that it is describing. There should be a space between the tick and the first letter of the comment. In-line comments are allowed on the same line as the code when used to label the end of a block of code (useful when there are multiple nested flow-control statements).

# 1.5.5.15 File Access

## File System Paths Must Not Be Hard Coded

Do not hard-code the directory path to a file when constructing the file name. You must instead obtain the directory path from a system function or a configuration parameter.

## File Numbers Must Not Be Hard Coded

When using file access commands such as `Open` and `Close` or others that utilize file numbers, you must not hard code the file number. You should instead obtain the next available file number using the `FreeFile` and save it to a variable. Use the same variable to refer to the file using any other command such as `Close`, `Print#`, `Input#`, `Get#`, `Put#`, `Write#`, `LOF` and `EOF`.

# 1.5.5.16 Dead Code

## Multiple Item Const Declarations Must Not Be Used

To avoid confusion and aid readability, individual constants must be declared using separate `Const` statements.

## Multiple Item Variable Declarations Must Not Be Used

To aid readability, individual variables must be declared on separate source lines.

## Unused Local Constant Declarations Must Be Removed

Do not leave local constant declarations that are not actually used within a procedure. Such declarations must be removed.

## Unused Local Variable Declarations Must Be Removed

Do not leave local variable declarations that are not actually used within a procedure. Such declarations must be removed.

## Unused Module-Level Constant Declarations Must Be Removed

Do not leave module-level constant declarations that are not actually used. Such declarations must be removed.

## Unused Module-Level Variable Declarations Must Be Removed

Do not leave module-level variable declarations that are not actually used. Such declarations must be removed.

## Unused Parameters Must Be Removed

Do not leave procedure declarations that include parameters that are not actually used. Such parameters must be removed.

### Unused Subroutines or Modules Must Be Commented out or Removed

Do not leave subroutines or modules that are not actually used. Such code must be commented out or removed.

## 1.5.5.17 Indentation

Indent four spaces at a time. This should be the default Tab value for the VBA editor. Every line of code beneath the sub title should be indented four spaces. Each nested flow-control statement (the `If`, `For` and `Do` statements, etc.) should have their contents indented four spaces. The Procedure Header comment does not need to be indented; however, all other comments regarding the code should match the code it is describing.

### If Then Else Statements

The `If Then Else…` statement should not be all on one line. The statement should be broken out to multiple lines for readability, and the proper indention should be applied.

### Select Case Statements

For `Case` statements, the `Case` should be indented under the `Select Case`. The action being performed should be on the line following the `Case` value and should be indented four spaces. Do not use the colon to put the code on the same line as the `Case` value.

## 1.5.5.18 Rumba VB Script

### Rumba Script Engine and Rumba scripts

Standards for code generated by the Rumba Script Engine are based on the standards established by MicroFocus. New code that is used by the Rumba Script Engine or any edits to the aforementioned generated code should follow VB standards.

## 1.5.5.19 VB.Net

### VB.Net

VB.Net projects/solutions should follow all the common VB standards.

### Projects Must Be Compiled to Native Code

To aid faster performance, all components and applications must be compiled to native code. (See *Compiled vs. Interpreted Applications* in the VB Documentation.)

Projects that are not compiled to native code degrade operational performance.

## Auto Increment Version Number Must Not Be Set

In order to control the use of version numbers for correct configuration management, the auto increment feature should be switched off. Version numbers must not be incremented accidentally.

## Source Files Must Reside In the Project File Folder

To ensure that all source files that pertain to a VB project are correctly accounted for, all project source files must reside in the same folder as the VB Project (VBP) file.

An exception to this standard is allowed when a source file is considered shared code used across multiple projects. This standard is to prevent non-shared source files from occurring outside of the VBP folder.

## Standard EXE Components

### Application Startup Object Must Be Sub Main

The application startup object must always be `Sub Main` for standard EXE components.

### Project Icons Must Be Set

An Icon name must be specified on the *Make* tab of the project properties dialog as this becomes the application Icon when the project is compiled.

### Option Base Statement Must Not Be Used

The `Option Base` statement should not be used, and the exact dimensions of arrays must be clearly stated by dimensioning with both lower and upper bounds. (A lower bound of zero is required.)

### Let Statement Must Not Be Used for Variable Assignment

The `Let` keyword must not be used for variable assignment.

### Gosub Statement Must Not Be Used

The `GoSub` statement must not be used. Instead, subroutines must always be defined as `Sub` or `Function` procedures.

### Return Statement Must Not Be Used

The `Return` statement must not be used. Instead, subroutines must always be defined as `Sub` or `Function` procedures, thereby negating the need for `Return`.

### Pointer Functions Must Not Be Used

VB pointer functions must not be used.

Use of the hidden VB pointer functions `ObjPtr()`, `StrPtr()` and `VarpPtr()` are not allowed.

### Global Keyword Must Not Be Used for Variable Declaration

The `Global` keyword must not be used to declare global variables. Instead they must be declared using the `Public` keyword.

### Global Keyword Must Not Be Used for Constants

The `Global` keyword must not be used to declare global constants. Instead they must be declared using the `Public` keyword.

## Compilation Options

### Advanced Compile Optimizations Should Not Be Set for VB Projects

The Advanced compiler optimizations located on the *Compile* tab of the project properties dialog should not be set.

### Create Symbolic Debug Info Option Must Not Be Used

The *Create Symbolic Debug Info* option located on the *Compile* tab of the project properties dialog must not be selected.

This should only be used for debug builds for use with debugging tools that can use the information as it exposes the internals of the application/component and significantly increases the size of the compiled application/component.

### Option Explicit Must Appear in All Source Files

The `Option Explicit` statement must be used at the head of all source files to ensure that variables can only be used if declared prior to use. Use of the `Option Explicit` statement helps to avoid bugs caused by the incorrect spelling of variable names.

### Non-MDI Forms

Non-multiple-document interface (non-MDI) form names must be correctly prefixed with `frm`.

VB projects that make use of one or more independent forms must ensure that each form file name is prefixed `frm`.

---

**NOTE** In VB 6.0, non-MDI applications are created by adding a form file to the project and leaving the default setting of `False` for the `MDIChild` property.

---

#### Error Handling

Always include some type of Error Handling in your code. Whenever possible, use structured exception handling (`Try-Catch`) rather than unstructured exception handling (`On Error …`).

### Try-Catch Use

Always use (at a minimum) a `Try-Catch` block to determine if any errors were received during processing.

## Control Names Must Begin with the Correct Prefix

Dependent on the type, all controls must be named with the correct prefix using the Object Hungarian notation naming conventions for VB as shown in the table below (Table 1-10).

**Control Names and Correct Prefixes**

| Object | Prefix |
|---|---|
| Animation button | ani |
| Checkbox | chk |
| Combo box | cbo |
| Data-bound combo box | dbc |
| Picture clip | clp |
| Command button | cmd |
| Communications | com |
| Control | ctl |
| Data | dat |
| Data Control | dtl |
| Directory list box | dir |
| Common dialog ctrl | dlg |
| Drive list box | drv |
| File list box | fil |
| Form | frm |
| Frame | fra |
| Gauge | gau |
| Group push button | gpb |
| Graph | gra |

## Control Names and Correct Prefixes

| Object | Prefix |
|---|---|
| Grid | grd |
| Data-bound grid | dbg |
| Pen Hedit | hed |
| Horizontal scrollbar | hsb |
| Image | img |
| Pen Ink | ink |
| Keyboard key status | key |
| Label | lbl |
| Line | lin |
| List box | lst |
| Data-bound list box | dbl |
| MDI child form | mdi |
| MAPI message | mpm |
| MAPI session | mps |
| MCI | mci |
| Menu | mnu |
| Object | obj |
| OLE container | ole |
| Option Button (3d) | opt |
| Outline control | out |
| 3d Panel (3d) | pnl |
| Picture box | pic |
| Report control | rpt |

**Control Names and Correct Prefixes**

| Object | Prefix |
|---|---|
| Shape controls | shp |
| Spin control | spn |
| Text box | txt |
| Timer | tmr |
| Vertical scroll bar | vsb |

**Table 1-10** Object Hungarian Notation Naming Conventions

### Project Properties

#### VB.net Projects Must Be Compiled

All VB projects must be unit tested using the compiled component.

#### Project File Name Must Match Component File Name

The project file name and its compiled component file name must be the same name. Use of any default project name assigned automatically by the Integrated Development Environment (IDE) such as project1, project2 … is not allowed.

#### Project Title Must Be Defined

Every VB project must have a meaningful title defined for the project.

## 1.5.6 Blueworx Voice Response Standards

Blueworx Voice Response is a scalable interactive voice response (IVR) system used to process incoming phone calls for routine requests for information that is available on Host 3270 screens. Blueworx Voice Response supports simple information retrieval using the telephone keypad along with advanced applications using VoiceXML (VXML).

## 1.5.6.1 Programming Guidelines

### Programming Considerations

The purpose of this section is to provide standards and guidelines for the development of Blueworx Voice Response applications.

Blueworx Voice Response is a high-level software product used to develop Voice Response applications.

## Program Design Considerations

Blueworx Voice Response applications contain the following basic components:

- Application definition
- 3270 scripts and screen definitions
- State tables
- Prompts (stored in prompt directories)
- Voice Segments (stored in voice directories)

The application definition contains the application name, initial 3270 script name, default screen timeout and refresh parameters. All 3270 scripts and screen definitions are defined within an application. Each application must have its own scripts and screens. These may be copied from another application.

The 3270 scripts are used to access the Host. The scripts mimic keystrokes and pass information between the state table and the Host. The screen definitions are captured from the Host, and the fields are defined in Blueworx Voice Response.

State tables control the progress of the telephone call and all processing of data. Tables may call other tables, 3270 scripts and prompts. They also answer, hang up, and transfer calls.

Prompts consist of segments that are voiced to the caller. They may also include logic.

Voice Segments are the actual verbiage that is spoken to the caller. They may be pieced together to form sentences.

Blueworx Voice Response has an inherent program design method. The bottom-up method must be used to program Blueworx Voice Response. The programming flow must go as follows:

1. The Voice Segments must be created.
2. Prompts must be coded.
3. 3270 scripts must be created and the screen captured.
4. State tables must be coded.

## System Design Considerations

Custom servers can be used to do unique operations that are not available in Blueworx Voice Response. The primary language of the Custom server is C, but other languages can be used as long as a C interface is developed for the call to that language. For example, a Custom server is used to send INFOrm data to MQ for logging entries to the Host.

Blueworx Voice Response can handle multiple calls of the same type, at the same time. Each call has its own copy of the state table, prompts, and so on. Data that is voiced to the caller should be logged, either in a log file or in INFOrm.

# 1.5.6.2 Design Considerations

## Blueworx Voice Response Naming Conventions

The purpose of this section of the Information Systems Standards Manual is to provide naming conventions for entities associated with a Blueworx Voice Response application.

All production state table names must be descriptive and fifteen characters or less.

All prompts and 3270 scripts must have descriptive names.

All voice segments names should be assigned a numeric value less than 65,536. Blueworx Voice Response sets this limitation.

# 1.5.7 Non-Host Databases

## 1.5.7.1 Approved Non-Host Database Management Systems

I /S currently supports the following Non-Host database management systems. The database management system software version must be a version currently supported by the database management system vendor.

- Oracle Enterprise Edition
- Microsoft SQL Server
- DB2 Enterprise Server
- DB2 Workgroup Server

Each application database supported by Database Administration must be implemented using a full version of the database management system. The Database Management System should not be a "**free**" version or scaled down version of the vendor's database management system engine (e.g., desktop editions, personal editions, express editions).

The Database Management Systems listed below do not currently provide the facilities to properly secure, backup, recover, diagnose and/or tune their databases and are considered non-enterprise level database systems. Database Administration does not support application databases implemented with these database management systems. Although these database management systems may be deployed, they are not preferred. Enterprise Architecture approval is required when a non-enterprise level database system is included in an enterprise-wide, customer facing or externally facing solution.

- Microsoft SQL Server Express Edition
- Microsoft Access
- Oracle Express and Standard Editions
- MySQL

## 1.5.7.2 Database Authentication and Security

Database Administration and Database Security are separate roles and have separate responsibilities. Database security is administered by the Data Security Group. Refer to *Security Management* > *Information Security Management* > *Application and Data Security Control Standards*.

Host Database Administrators (DBAs) perform limited database security functions that include the following:

- Send requests to add new resources to RACF

- Grant access to DB2 resources (limited test environments if they are not currently under RACF)

Alternative Platform Database Administrators (ALTDBAs) perform limited database security functions that include the following:

- Remove public access after new database setup
- Drop any accounts created by the database installation that are not required
- Grant SQL Server access to Active Directory Security Groups and Service Accounts
- Create Database Roles in SQL Server and Oracle
- Create Oracle Profiles
- Create ALTDBA and limited Data Security accounts on new Oracle databases

The data within the customer databases are owned by the I/S Application Teams (usually the I/S Application Support group). The Application Owner is responsible for identifying the users, groups, and roles requiring access to these databases (tables, columns, and rows).

Applications connecting to SQL Server databases must be domain aware. The accounts must be Active Directory accounts and not local database accounts.

If an application has no way of working properly without a local account, a Business Request Justification (BRJ) must be submitted by the I/S Application Team.

## 1.5.7.3 DBA Data Manipulation

Application databases are created in support of business processes implemented as application software. The data stored in application databases in Production must be maintained by application software or application support resources or both. Application databases include both operational databases and informational databases (e.g., Data Warehouses and Data Marts).

There is a distinction between maintaining database structures and database data. Database data is the responsibility of the application area; therefore, Database Administrators must not manipulate data. Database Administrators use Data Definition Language (DDL) commands to maintain and manage database structures. Data Manipulation Language (DML) commands include delete, insert, and update statements performed on rows in application database tables, and DML is performed by the application areas.

The DBAs will review database scripts provided by the application custodians for DML type operations. Since DBAs do not perform operations that delete, insert and/or update business data in a database, any scripts containing DML type operations will be returned to the requester for separation of DML from DDL.

DBAs may make infrequent exceptions to this policy to resolve critical database issues (e.g., Severity 1 Break/Fix issues) and in unusual circumstances when approved by both the DBA Manager and the I/S Application Manager

## 1.5.7.4 Data Archival and Purging

Application software or the application's support team is responsible for database data archival and data archival deletion. Archived data should be separate and independent from the database from which it is moved. Archived data should be written to flat files or stored to a hardware and software independent

format that is readable long after the operating system, hardware and/or database management software are no longer supported.

## 1.5.7.5 Contingency Planning and Disaster Recovery Planning

### Database Backups

DBAs have the responsibility to set up database backup processes based on the application specifications.

### Database Disaster Recovery (Remote) Plans

The application's support team is responsible for creating and maintaining the application's DR Plan, which identifies any databases required for the recovery of the application. Database Administration maintains DR Plans for each database platform.

### Local Database Restore

A request for local database restore is initiated through a Service Request or Incident Ticket submitted by an I/S Application area. The DBA taking ownership of the database restore must get both the DBA Manager and the I/S Application Manager approvals prior to executing the recovery activity. The Manager approvals must be documented in the Service Request or Incident Ticket.

## 1.5.7.6 Data Encryption

The Database Administrator does not make a determination as to whether the data or a subset of the data needs to be encrypted. The decision to encrypt data is the responsibility of the I/S Application Manager, and they are responsible for communicating that decision to the DBA. Data encryption, if required, must be implemented using a Database Management System (DBMS) and approved BlueCross BlueShield of South Carolina software, appliances, and methods.

## 1.5.7.7 Database Links

DBMS links provide a communication and data transfer path between any two connected databases. DBMS links are not allowed between production and test databases because such links may be used by malicious users to discover and obtain unauthorized access to remote systems. Database links between production and development DBMSs may provide a means for developers or users or both to access production data not authorized for their access, or to introduce untested or unauthorized application code to the production database.

When data in one database environment is needed in another database environment (e.g., production data is needed in the Qual environment), the application's system support team must develop a method to extract and load the required data.

# 1.6 Platform Development Standards

## 1.6.1 IBM Web Content Manager Standards

### 1.6.1.1 Requiring Name Changes

Names for existing ***IBM Web Content Manager*** (***WCM***) components will be brought into alignment with this specification when the component is altered for any project, change sheet or Break/Fix.

### 1.6.1.2 Component Naming Conventions

The component name and display name will be identical to facilitate identification of component references within the SystemOut.log. All design components created for ***WCM*** will follow the naming conventions as stated herein (Table 1-11) and will have one of the following formats for the first few characters.

| | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* |
|---|---|---|---|---|---|---|---|---|---|
| **Component Format** | | | | _ | | | | _ | |

**Legend**

**Component Type** *See Table 1-11 below.*

**Application** *See Table 1-11 below.*

**Function** *No limit – See Table 1-11 below.*

## Component Naming Conventions

| Pos | Field Description |
|---|---|
| 1-3 | **Component Type** – a capitalized three-letter code indicating what type of component is being used. |

| Component Type | Code |
|---|---|
| Authoring Tools | AUT |
| Component Reference | CMR |
| Date and Time | DAT |
| Federated Content | FDC |
| File Resource | FIR |
| HTML | HTM |
| Image | IMG |
| JSP | JSP |
| Link | LNK |
| Menu | MNU |
| Navigator | NAV |
| Number | NUM |
| **Component Type** | **Code** |
| Page Navigation | PGN |
| Personalization | PZN |
| Rich Text | RTX |
| Search | SCH |
| Short Text | SHT |
| Style-sheet | CSS |
| Taxonomy | TAX |
| Text | TXT |
| User Name | USR |
| User Selection | USS |

**Component Naming Conventions**

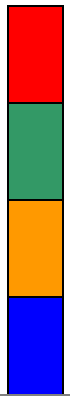
| Pos | Field Description |
|---|---|
| 4 | **Underscore "_"** |
| 5-7 | **Application** – a three-letter capitalized abbreviation or acronym identifying the application where this component is used.<br><br>Examples:<br><br>**Application** / **Code**<br>My e-Work / MEW<br>Communiqué / CMM |
| 8 | **Underscore "_"** |
| 9-on | **Function** – A description of the function of this particular component.<br><br>Examples: Jobs; Manager_Area, News_Events |

**Table 1-11** Component Naming Conventions

## 1.6.1.3 Template/Workflow Naming Conventions

The template/workflow name and display name, since they often are displayed to the content manager, may be different to facilitate readability and reduce confusion. All template and workflow names will follow the naming conventions as stated herein (Table 1-12), and will have the following format for the first 14 characters.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Template/Workflow Format** | | | – | | | | | | | – | | | | – | |



**Legend:**

**Template/Workflow Type** *See Table 1-12 below.*

**Library** *See Table 1-12 below.*

**Type** *See Table 1-12 below.*

**Constituent** *No limit – See Table 1-12 below.*

## Template/Workflow Naming Conventions

| Pos | Field Description |
|---|---|
| 1-2 | **Template/Workflow Type** – a capitalized two-letter code indicating what type of template is being used. |

| Component Type | Code |
|---|---|
| Authoring Template | AT |
| Presentation Template | PT |
| Workflow | WF |

## Template/Workflow Naming Conventions

<table>
<tr><th>Pos</th><th>Field Description</th></tr>
<tr><td>3</td><td>Underscore "_"</td></tr>
<tr><td>4-9</td><td>Library – a six-letter abbreviation or acronym identifying the WCM library where this template or workflow is stored.<br><br>Examples:<br><br>
<table>
<tr><th>Library</th><th>Code</th></tr>
<tr><td>My e-Work</td><td>myewrk</td></tr>
<tr><td>BlueCross BlueShield of SC</td><td>bcbssc</td></tr>
</table>
</td></tr>
<tr><td>10</td><td>Underscore "_"</td></tr>
<tr><td>11-13</td><td>Type – a three-letter capitalized abbreviation or acronym identifying the type of workflow.<br><br>Examples:<br><br>
<table>
<tr><th>Type</th><th>Code</th></tr>
<tr><td>Standard</td><td>STD</td></tr>
<tr><td>Expedited</td><td>EXP</td></tr>
<tr><td>Manager</td><td>MGR</td></tr>
</table>
</td></tr>
<tr><td>14</td><td>Underscore "_"</td></tr>
<tr><td>15-on</td><td>Constituent – A description of the users of this particular template or workflow.<br><br>Examples: Provider, Change Management</td></tr>
</table>

**Table 1-12** Template/Workflow Naming Conventions

## 1.6.1.4 Syndication Naming Conventions

Syndicator pairs created between ***WCM*** servers will follow the convention as stated herein. To identify what is being syndicated and where it is being syndicated, syndicator and subscriber pairs will follow the naming convention below (Table 1-13).

**Syndication Naming Conventions**

| Pos | Field Description |
|---|---|
| 1-3 | <table><tr><th>Function</th><th>Code</th></tr><tr><td>Syndicator</td><td>SYN</td></tr><tr><td>Subscriber</td><td>SUB</td></tr></table> |
| 4 | **Underscore "_"** |
| Var | **Content Syndicating** – A code describing what is being transferred in this syndicator pair.<br><table><tr><th>Function</th><th>Code</th></tr><tr><td>Content</td><td>Content</td></tr><tr><td>Design</td><td>Design</td></tr><tr><td>Both Content and Design</td><td>All</td></tr></table> |
| Var | **Underscore "_"** |
| Var | **From** – Usually a server designation, this is where the syndicator of the pair is configured. It can either be a server name or a server designation (e.g., "a70zpcmmwpt1" or "authoring"). |
| Var | **Underscore "_"** |
| Var | **To** – Usually a server designation, this is where the subscriber of the pair is configured. It can either be a server name or a server designation (e.g., "a70zpcmmwpt1" or "authoring"). |

**Table 1-13** Syndication Naming Conventions

**Examples:**

**SYN_content_authoring_prod2** (syndicator of content from authoring server to production server 2);

**SUB_design_a70zqcmmwpt1_a70zpcmmwpt1** (subscriber of design from server a70zqcmmwpt1 to server a70zpcmmwpt1)

# 1.6.2 IBM Websphere MQ Standards for Non-Host

## 1.6.2.1 Websphere MQ for Java

The WebSphere MQ classes for Java allow a program written in the Java programming language to connect to WebSphere MQ as a WebSphere MQ client using TCP/IP, or directly to an MQ Queue Manager using the Java Native Interface (JNI). They allow Java applets, applications, Java Server Pages and servlets access to the messaging and queuing services of WebSphere MQ. If the client-style connection is used, no additional WebSphere MQ code is required on the client machine. The WebSphere MQ classes for Java enable a message-based approach to application integration using Java.

Java programs should use well-established APIs to send and receive WebSphere MQ messages. In order to support guaranteed messaging, the following is required:

Archive all messages sent.

Store messages that failed in separate files.

Include batch retry logic to resend failed messages. This does not apply when the user has the option to retry.

# 1.7 Tool Specific Standards

## 1.7.1 Micro Focus Virtual User Generator

Micro Focus Virtual User Generator (VuGen) is a scripting tool used to emulate user actions. VuGen is used to create test scripts of user actions that are then used to load test applications and monitor end-user response time and availability. VuGen is part of the Hewlett Packard Enterprise LoadRunner software suite.

### 1.7.1.1 VuGen Scripting Files

This section lists the allowable filenames, suffixes and standards.

The developer is responsible for ensuring that functions, modules, components, etc., that generate files also follow these same naming conventions.

#### Filename Creation

##### Reserved Filenames

No reserved filenames are referenced in the VuGen documentation.

##### Filename Formats

**Naming Convention**

Always use the lower camel case naming convention. It is referred to as lower camel case due to the up and down nature of the characters in a name as shown in the example below.

```
getDataFromHost.c
```

**Abbreviations in Filenames**

Capitalize the first letter of abbreviations and use lower case for the rest, except when the abbreviation begins the name. In that instance, the lower camel case naming convention takes priority. For example, the filename beginning with an abbreviation would look like the filename below.

```
lcasnhGuidelines.doc
```

A filename that contains an abbreviation would look like this example.

```
guidelinesForLcasnh.doc
```

**Spaces**

Do not use embedded spaces in the filename.

**Underscores**

Do not use underscores unless it is justifiable, such as when you need to follow a developed and approved schema.

## File Suffixes

VuGen scripts use the following file suffixes (Table 1-14):

**File Suffixes**

| File Extension | Description |
|---|---|
| *.bak | Backup files automatically generated by VuGen. They can be deleted when no longer needed. |
| *.c | Standard C language file extension. All files in the script folder with the .c extension correspond to an Action in the VuGen script. Each action has its own .c file, including vuser_init, vuser_end. |
| *.cfg | Contains configuration data |
| *.ci | Binary file created during script compilation, <scriptname>.ci. They can be deleted when no longer needed. |
| *.dat | Data files |
| *.gif | Bitmap image format using lossless data compression |
| *.htm | Hypertext Markup Language File |
| *.html | Hypertext Markup Language File |
| *.idx | Parameter index files. The .idx files are automatically created index files. These files are created for faster access of the parameterization files. They are generated when the script is compiled and can be deleted when no longer needed without affecting future replay of the script. |
| *.inf | Setup information file |
| *.log | Log files |
| *.png | Portable Network Graphic file. A raster image format using lossless compression. |
| *.prm | Parameter file |
| *.txt | Text files |
| *.usp | Contains run login |
| *.usr | This is the main file from which VuGen opens the script, < script_name>.usr. |

**Table 1-14** File Suffixes

## File Organization

### Working Directory Structure

As shown below, VuGen generates a directory with the same name as the script name in the directory that the user specifies. VuGen creates all files necessary to run the script in this directory and also usually creates two subdirectories within the script directory: a Data directory and a Results directory. There are often subdirectories under the Data directory as well, such as Events, Snapshots, etc.

**The bold text part of the path is required, all else is for example purposes only.**
DriveName\ProjectDB \ **scriptName**
DriveName\ProjectDB \ **scriptName\Data**
DriveName\ProjectDB \ **scriptName\Results**

The directory structure on the scripting machine can be the developer's preference. However, the script directory format must be patterned after the above example.

The Results directory is optional during the development of the script because the directory will not be saved. The Results directory will not be part of the .zip file with the script, so it is not required for VuGen script development because it will not be included in the package that is uploaded to the client monitor.

However, when the VuGen script is executed in Load Runner, the Results directory is saved.

# 1.7.1.2 VuGen Code Conventions

## Indentation

Four spaces must be used as the unit of indentation. Do not use tab characters unless your editor is set to replace tabs with spaces.

## Comments

VuGen Script programs are generated in C and have comments that are block delimited by `/*...*/` or line delimited by `//`.

Use comments to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program. For example, information about how the corresponding script is built or in what directory the script resides should not be included as a comment.

Discussions of nontrivial or non-obvious design decisions are appropriate, but avoid duplicating information that is present in, and clear from, the code. It is too easy for redundant comments to get out of date. In general, avoid any comments that are likely to get out of date as the code evolves.

Do not enclose comments in large boxes drawn with asterisks or other characters.

**NOTE** The frequency of comments sometimes reflects poor quality of code. When you feel compelled to add frequent comments, consider rewriting the code to make it clearer.

## File Comments

File comments must appear at the beginning of all files and include a brief description of the functions and purpose of the code. All source files must begin with a business copyright statement followed by a standard purpose comment. Use the approved business copyright statement as shown below:

```
/*
 * THIS MATERIAL IS THE CONFIDENTIAL, PROPRIETARY AND TRADE SECRET PRODUCT OF
 * BLUECROSS BLUESHIELD OF SOUTH CAROLINA AND ITS SUBSIDIARIES.  ANY
 * UNAUTHORIZED USE, REPRODUCTION OR TRANSFER OF THESE MATERIALS IS STRICTLY
 * PROHIBITED.
 * COPYRIGHT {current year} BLUECROSS BLUESHIELD OF SOUTH CAROLINA   ALL RIGHTS
 * RESERVED.
 */
```

Programs can have four styles of implementation comments:

- Block
- Single-line
- Trailing
- End-of-line.

## Block Comments

Block comments are comments outside of a function and are used to provide descriptions of functions, Data Structures and algorithms. Block comments must be used at the beginning of each file and before each function. They can also be used in other places, such as within functions. Block comments inside a function must be indented to the same level as the code they describe. A block comment must be preceded by a blank line to set it apart from the rest of the code as shown in the following example.

```
/*
* Here is a block comment.
*/
```

Function comments, if present, must be in the form of a block comment. They must contain a brief description of what the function does. If a function change warrants a change in the function comments, change the existing comment.

Function comments must be structured according to the example below:

```
/*
* Description : Descriptive text here
* Parameters: A description of all parameters passed to the function
* Return Values: A description of the return value.
*/
```

Omit the parts that do not apply. For example, a function that does not throw any exceptions or accept parameters would omit the Parameters portion from the function comment.

## Single-Line Comments

Short comments can appear on a single line indented to the level of the code that follows. If a comment can't be written in a single line, it must follow the block comment format. Comments within a method declaration must use two slashes `//`, making it easier to comment out large blocks of code when needed. A blank line must precede a single-line comment. The following example is a single-line comment.

```
if (condition) {


    // Handle the condition.
    ...
}
```

## Trailing Comments

Very short comments can appear on the same line as the code they describe. They must be shifted far enough to the right to separate them from the code statements. If more than one short comment appears in a block of code, they must all be indented to the same tab setting. Be sure to use the double slash `//` comment delimiter when writing trailing comments to avoid interfering with `/*`. The following example shows a trailing comment in C code.

```
if (a == 2) {
    return true;        // special case
} else {
    return isPrime(a);  // works only for odd
```

```
}
```

### End-of-Line Comments

The double slash `//` comment delimiter can comment out a complete line or only a partial line. Do not use it on consecutive multiple lines for text comments. However, it can be used in multiple consecutive lines to comment out sections of code.

## White Space

### Blank Lines

Blank lines improve readability by setting off sections of code that are logically related.

One blank line must always be used between logical sections of a source file.

Two blank lines must always be used between sections of a source file.

One blank line must always be used in the following circumstances:

- Between functions
- Between the local variables in a function and its first statement
- Before a block or single-line comment
- Between logical sections inside a function to improve readability

### Blank Spaces

Blank spaces must be used in the following circumstances.

A keyword followed by a parenthesis must be separated by a space as shown in the example below.

```
while (true) {

    ...

}
```

To distinguish keywords from method calls, do not use a blank space between a method name and its opening parenthesis.

A blank space must appear after commas in argument lists.

All binary operators except a period `.` must be separated from their operands by spaces. Blank spaces will never separate unary operators such as unary minus, increment `++`, and decrement `--`from their operands. Refer to the following example.

```
a += c + d;

a = (a + b) / (c * d);
```

```
while (d++ == s++) {

    n++;

}

printSize("size is " + foo + "\n");
```

The expressions in a `For` statement must be separated by blank spaces as shown in the following example.

```
for (expr1; expr2; expr3)
```

## Line Length

To ensure readability in user-added code, line length must not exceed 80 characters. Code automatically generated by the tool does not have to conform to this requirement and will remain as generated.

## Wrapping Lines

When an expression will not fit on a single line, break it according to these general principles:

- Break after a comma.
- Break before an operator.
- Higher-level breaks are preferred over lower-level breaks.
- Align the new line with the beginning of the expression at the same level on the previous line, or the nearest tab after that point.

If the preceding rules lead to confusing code or to code that's up against the right margin, just indent four spaces instead.

Here are some examples of breaking method calls.

```
someMethod(inLongExpression1, inLongExpression2, inLongExpression3,

inLongExpression4, inLongExpression5);
```

```
var = someMethod1(inLongExpression1,

someMethod2(inLongExpression2,inLongExpression3));
```

Two examples of breaking an arithmetic expression are shown below. The first example is preferred because the break occurs outside the parenthesized expression, which is at a higher level.

Preferred

```
longName1 = longName2 * (longName3 + longName4 - longName5)
            + (4 * longname6);
```

Avoid

```
longName1 = longName2 * (longName3 + longName4
            - longName5) + (4 * longname6);
```

Two examples of indenting function declarations are shown below. The first is the conventional case. The second would shift the second and third lines to the far right if it used conventional indentation, so instead it indents only four spaces.

```
//CONVENTIONAL INDENTATION

someMethod(int inParameter, Object inObject, String inYetAnotherArg,
Object
            inStillAnother) {

...

}
```

Indent four spaces to avoid very deep indents.

```
private static synchronized horkingLongMethodName(

    int inAanArg, Object inAnotherArg, String YetAnotherArg,

    Object inStillAnother) {

    ...

}
```

Avoid this indentation; it is visually hard to read.

```
if ((condition1 && condition2)

   || (condition3 && condition4)

   ||!(condition5 && condition6)) {

doSomethingAboutIt();

}
```

Use this indentation instead.

```
if ((condition1 && condition2)

|| (condition3 && condition4)
```

```
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Here are two acceptable ways to format ternary expressions.

```
alpha = (aLongBooleanExpression) ? beta : gamma;
```

```
alpha = (aLongBooleanExpression)
    ? beta: gamma;
```

## Declarations

### Providing Access to Variables

Use global variables only when absolutely necessary and not for convenience or ease of programming.

### Number Per Line

Declare user-defined variables in a section at the top of each script.

One declaration per line is required because it encourages commenting.

```
int level;  // indentation level
int size;    // size of table
```

Avoid this type of declaration.

```
int level, size;
```

**NOTE** The examples above use one space between the type and the identifier. Another acceptable alternative is to use tabs like the example below.

```
int   level;              // indentation level
int   size;               // size of table
Object currentEntry;    // currently selected table entry
```

### Initialization

Always initialize local variables where they're declared. If a variable depends on some computation occurring before it is first used, override with a calculated value after initialization.

### Placement

Put declarations only at the beginning of blocks as shown in the following example. (A block is any code surrounded by curly braces `{` and `}`.) Don't wait to declare variables until their first use; it can be confusing and impede code portability.

```
void myMethod() {
    int int1 = 0;         // beginning of method block


    if (condition) {
        int int2 = 0;     // beginning of "if" block
        ...
    }
}
for (int i = 0; i < maxLoops; i++) { ... }
```

## Programming Practices

### Simple Statements

Each line can contain at most one statement. Examples of correctly formatted simple statements are shown below.

```
argv++;
argc--;
```

Avoid this style.

```
argv++; argc--;
```

### Compound Statements

Compound statements are statements that contain lists of statements enclosed in braces `{statements }`. See the following sections for examples.

The enclosed statements must be indented one more level than the compound statement.

The opening brace must be at the end of the line that begins the compound statement; the closing brace begins a line and must be indented to the beginning of the compound statement.

Use braces around all statements, even single statements, when they are part of a control structure, such as an `if-else` or a `for` statement. This makes it easier to add statements without accidentally introducing bugs due to missing braces.

## Return Statements

A return statement with a value should use parentheses when that makes the return value more obvious in some way. See the example below.

```
return myDisk.size();

return (size ? size : defaultSize);
```

## If, If-Else, If Else-If Else Statements

In the `if-else` class of statements, the else condition starts after a closing brace and must use the following format.

```
if (condition) {
    statements;
}
```

```
if (condition) {
    statements;
} else {
    statements;
}
```

```
if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
```

```
    statements;

}
```

**NOTE** The `if` statements always use braces `{ }`. Refer to the subsection *Compound Statements* above. Avoid the following error-prone form because it omits the braces `{ }`.

```
if (condition)

    statement;
```

### For Statements

A `For` statement must use the following format.

```
for (initialization; condition; update) {

    statements;

}
```

Avoid empty `for` statements where all the work is done in the initialization, condition, and update clauses.

An empty `for` statement has the following format.

```
for (initialization; condition; update);
```

Use separate initialization statements before the `for` loop to avoid the complexity of using more than one variable.

### While Statements

A `while` statement must use the following format.

```
while (condition) {

    statements;

}
```

Avoid using the following empty `while` statement.

```
while (condition);
```

## Do-While Statements

In a `do-while` statement, the while clause starts on a new line and must have the following format.

```
do {
    statements;
} while (condition);
```

## Switch Statements

A `switch` statement must use the format as described below.

Every time a case falls through (doesn't include a `break` statement), add a comment where the `break` statement would normally be as shown in the following code example that uses a `//  falls through` comment.

```
switch (condition) {
    case ABC:
        statements;
        // falls through
```

Every `switch` statement must include a default case. The `break` in the default case is redundant, but it prevents a fall-through error if another `case` is added later. Place the default case after all other case statements as shown below.

```
    case DEF:
        statements;
        break;
        case XYZ:
        statements;
        break;
    default:
        statements;
        break;
}
```

## Constants

Do not code numerical constants (literals) directly, except for -1, 0, and 1, which can appear in a `for` loop as counter values.

## Variable Assignments

As shown in the example below, avoid assigning several variables to the same value in a single statement because it is hard to read.

```
fooBar.fChar = barFoo.lchar = 'c';
```

Do not use the assignment operator in a place where it can be easily confused with the equality operator as shown in the example below.

```
if (c++ = d++) {


}
```

Must be written as shown below.

```
if ((c++ = d++) != 0) {


}
```

As shown in the example below, do not use embedded assignments in an attempt to improve run-time performance. That is the job of the compiler.

```
d = (a = b + c) + r;
```

Must be written as shown below.

```
a = b + c;

d = a + r;
```

## Naming Conventions

Using naming conventions within the code of a program makes it more understandable and easier to read. Naming conventions can also give information about the function of the identifier.

The developer is responsible for ensuring that functions, modules, components, etc., that generate files also follow the file naming conventions.

### Variable Names

Use the lower camel case naming convention.

Variable names must start with a letter.

Variable names must be short yet meaningful. The choice of a variable name must easily indicate to the casual observer the intent of its use. Avoid one-character variable names except for temporary variables. Always spell out the full representation of a variable when possible to increase documentation efficiency.

Although underscores are allowed syntactically, they are discouraged as a general rule because in some instances, underscores appear to be spaces and unnecessarily lengthen the name.

```
numOfProviders = 0

hostDataString

lastName
```

**Functions**

Functions must be verbs when appropriate and use the lower camel case naming convention.

```
run();
runFast();
getBackground();
```

**Constants**

The names of variables declared as function constants and of ANSI constants must be all uppercase with words separated by underscores _. ANSI constants must be avoided, for ease of debugging, and in this case deals with the ANSI character set (ASCII codes 128–255).

```
static final int MIN_WIDTH = 4;

static final int MAX_WIDTH = 999;

static final int GET_THE_CPU = 1;
```

**Abbreviations**

Capitalize the first letter of abbreviations and use lower case letters for the remainder. When the abbreviation begins the name, use the lower camel case naming convention.

```
issmLinkString

ipAddress
```

**Spaces**

Do not use embedded spaces in the filename.

### Underscores

Do not use underscores unless it is justifiable such as when you are following a developed and approved schema.

## Code Redundancy

Use functions or reusable Dynamic Link Libraries (DLLs) where there is redundancy in the code.

## VuGen Specific Standards

### Parameter Delineation

Always use braces, `{}` to delineate parameters (params) like the following example.

```
lr_eval_string("The number of rows is {rowCount}")
```

## Miscellaneous Practices

### Parentheses

Use parentheses liberally in expressions involving mixed operators to avoid operator precedence problems. Even if the operator precedence seems clear to you, it might not be to others. Do not assume that other software developers know precedence as well as you do.

Use this style.

```
if ((a == b) && (c == d))
```

Avoid this style.

```
if (a == b && c == d)
```

### Returning Values

Try to make the structure of your program match the intent. For example, do not use the format shown below.

```
if (booleanExpression) {

    return true;

} else {

    return false;

}
```

You should just write it as in the following example.

```
return booleanExpression;
```

**Expressions Before Question Mark in the Conditional Operator**

If an expression containing a binary operator appears before the question mark in the ternary `:` operator, it must be parenthesized like the following example.

```
(x >= 0) ? x : -x;
```

**Special Comments**

Use `REWORKME` in a comment to flag something that is bogus but works.

Use `FIXME` to flag something that is bogus and broken.

This is usually necessary when classes have not been completed and a workaround is coded until a better means is available.

Examples of these special comments are shown below.

```
// REWORKME This is deprecated and needs to be changed

String value = Object.getStringRepresentation();
```

Or

```
//FIXME The following is incorrect and needs to be reworked.

Class value = Object.getClass();
```

# 1.7.2 IBM Rational Application Developer

IBM Rational Application Developer (RAD) is an approved tool for Java Development.

## 1.7.2.1 Formatter Settings

The formatter settings are a list of rules that RAD uses to govern the format of code. There are specific formatters for different types of code (Java, Javascript, CSS, HTML, etc.). As an example, these rules will ensure the source code uses spaces instead of tabs. For formatting standards, refer to the section *Language Specific Standards* above. These formatters must be used to ensure ISSM formatting compliance.

There is a set of formatter XML files for Java and Javascript that must be imported into each RAD workspace to set up the formatters. They are located in the GitHub Enterprise InformationSystems/application-technical-standards repository, under the directory structure application-technical-standards/cfgfiles/RAD/formatters. The filenames are issm-java-formatter-standards.xml and issm-javascript-formatter-standards.xml.

To import them into RAD:

1. Select Window -> Preferences.

2. Type “formatter” in the filter field and select the appropriate type of formatter (either Java or Javascript).
3. Click on the “Import…” button and point RAD to the formatter XML file retrieved from GitHub Enterprise.
4. Click OK at the bottom of the dialog to complete the process.

The formatting standards for HTML, CSS, and XML are stored as a preference file not an XML file. The preference file must be imported into each RAD workspace to set up these formatters. It is located in the GitHub Enterprise InformationSystems/application-technical-standards repository, under the directory structure application-technical-standards/cfgfiles/RAD/formatters/issm-web-preferences.epf.

To import preferences into RAD:

1. Select File -> Import… -> General -> Preferences and click on the Next button.
2. In the “From preference file” dialog, enter or browse the location of the preference file retrieved from GitHub Enterprise.
3. Click on “Finish” to complete the process.

## 1.7.2.2 Template Settings

The template settings within RAD are a list of templates that are used to generate new files when needed. There are specific templates for different types of code (Java and Javascript). As an example, these templates will ensure each source file has the standard BlueCross BlueShield of South Carolina copyright statement at the top. For the standards these templates address, refer to the section *Language Specific Standards* above. These templates must be used to ensure ISSM compliance.

There is a set of template XML files that should be imported into each RAD workspace to set up the templates. They are located in the GitHub Enterprise InformationSystems/application-technical-standards repository, under the directory structure application-technical-standards/cfgfiles/RAD/templates The filenames are issm-java-codetemplates.xml and issm-javascript-codetemplates.xml.

To import them into RAD:

1. Select Window -> Preferences.
2. Type “code template” in the filter field and select the appropriate type of template.
3. Click on the “Import…” button and point RAD to the template XML files retrieved from GitHub Enterprise.
4. Click OK at the bottom of the dialog to complete the process.

# 1.8 Design Considerations

## 1.8.1 Batch/Online Considerations

This section discusses the major differences in concept and execution of systems design and programming between the typical batch and online application. When designing files to be shared with online or considering the applicability of sharing both data and procedure division code in the two executing environments, these considerations must be weighed. In the event that development and implementation should follow, then these considerations become rules, which must be followed if the endeavor is to be wholly successful.

### 1.8.1.1 System Considerations

When faced with the requirements of analyzing a system, which will impact the online environment, there are two major areas of possible contention. The first and most common is the creation of datasets to be used, both for inquiry and/or update, by online modules. The second area concerns duplication of effort.

#### File Considerations

Refer to *Technical Standards — Applications > Application Coding > File Design Considerations* for additional information.

#### Duplication of Effort

The current philosophy of the responsibility of the online system is that any data keyed into an online task should be edited completely to ensure the acceptability of the data but not to affect the actual update. This responsibility, master file update, is left to the tasks executing in the batch environment. This has been our executing philosophy since the inception of the online. This has caused some problems in the past and is being reviewed now with the intent of changing the basic concepts of online processing. However, the existing philosophy does tend to lead naturally toward duplication of effort, for the same edits performed online are also executed in the batch environment. To combat the resulting redundant programming, the concept of using common COPYLIB modules to be included by both batch and online systems was developed. Until such time as online systems are performing master file updates, this is the most viable solution available to reduce duplication of effort.

#### Common Edits

Line of Business (LOB) specific edits used in CICS and batch programming should be coded as includable edits.

Edits that do not relate to a specific LOB and are required by batch programming and CICS should be coded as includable edits by EMC systems.

Multiple reports will not be written from the same "FD" statement. Refer to *Procedures & Tools > Host Tools > Developer Tools > Host Development Environment* for the definition of report(s).

### Transaction Accountability Considerations

TIMS (Time Information Management System) accounts for all Host CPU usage. In order to achieve this, when developing online CICS transactions, the following standards apply.

Each online system must account for each "unit of work" by one or more CICS transactions being executed in the CICS TOR (Terminal Only Region). The CICS transaction(s) would represent the CPU path of the "unit of work" and provide a count of the number of transactions.

Using a MQ process in near time mode must also use a CICS transaction to account for any extension of online.

CICS transactions defined by the system to represent a "unit of work" in the AOR (Application Only Region) will capture CPU usage by sampling these transactions.

CPU usage for near time processes will be captured from the "batch" job running in near time mode.

For background processes that do not execute from a terminal in a TOR, MQ near time processes, or long running CICS transactions that process more than one "unit of work," dummy transactions must be "STARTED" in the count TOR.

Unit of work: Defined by each system (example: AMMS unit of work = claim)

For remote processes that access the Host only as database or FTP calls, "unit of work" must be defined in terms of files or data created and supplementary batch jobs created to track that data. These counts will be factored into the CPU time accounted to the transaction RACF IDs assigned to these processes.

## 1.8.2 Input Design Considerations

The purpose of this section is to provide input and output design considerations for electronic data received at and sent from BlueCross. With the passage of the Health Insurance Portability and Accountability Act of 1996, the US Secretary of Health and Human Services is charged with determining standards for health-related data transmissions. This includes setting up national standards for claims input/output, eligibility, remittances, etc. National standards, in the form of ANSI Accredited Standards Committee (ASC) X12 , already exist for MOST electronic transaction types. It is presumed by the Blue Cross and Blue Shield Association that the standards selected will be ANSI ASC X12. Therefore, ANSI ASC X12 standards or the UN/EDIFACT equivalent should be used for ANY dataset coming to BlueCross from an external entity or for ANY dataset leaving BlueCross to go to an external entity. Non-standard formats should be allowed only with I/S Director approval.

## 1.8.2.1 Input/Output Data

Industry standard claims formats must be utilized unless another format is approved by an I/S Director. Valid claims formats are:

- NCPDP for drug claims
- ADA for dental claims
- HCFA UB92 192 for institutional claims
- HCFA NSF 320 for professional claims

- ANSI ASC 837 for institutional and professional claims

Industry standard remittance formats must be utilized unless another format is approved by an I/S Director. Valid remittance formats are:

- NCPDP for drug remittances
- ANSI ASC 835 for institutional and professional remittances
- UN/EDIFACT equivalent of ASC 835

Industry standard eligibility formats must be utilized unless another format is approved by an I/S Director. Valid eligibility formats are:

- ANSI ASC 834 for institutional and professional
- UN/EDIFACT equivalent of ASC 834

There are currently no national standards for electronic report formats.

## 1.8.2.2 Input Data

All input fields should have description, source of data and validation criteria documented in the Data Dictionary with appropriate reference numbers documented on the copybook.

The copybook/include members should have valid values coded as '88' levels within the copybook.

Data fields should be initialized to a value that is consistent with the associated picture clause. Do not initialize it to a value that, if entered, has specific meaning. For instance, if possible values are "Y" or "N," do not initialize the field to either since those indicators might cause special processing to be performed. Instead, initialize the field to spaces. This can be very difficult with numeric fields since all numeric fields should be initialized to zeroes.

No field should be coded by the user or keyed in data prep that can be logically generated by the system. A perhaps trite example would be: no one should have to code or key both date of birth and age since age can be derived logically from the date of birth. There are many cases where a little imagination can preclude many thousands of keystrokes over a period of years.

## 1.8.2.3 Input Validation

Perform as many validation edits as possible before terminating the edit process. Capture and save as much data as possible to prevent re-keying of "clean" data.

Do not require input of fields that can be retrieved or derived from other data on input transactions.

The entire error record should be written to an error file that can be used for generating reports.

Always report on the validation process by supplying statistics on the number of transactions accepted and reports of those rejected or in error.

# 1.9 Date Considerations

The purpose of this section is to provide policies and guidelines concerning the accessing of date-related functions, processing control dates or other control data (e.g., Parm Cards).

## 1.9.1 General Discussion

The approved methods of obtaining the current date are:

- COBOL "ACCEPT" verb
- COBOL intrinsic function "CURRENT-DATE"
- Language Environment (LE) callable services
- Installation standard date shells CORPL207 and CORPL214 (Refer to *Procedures & Tools > Host Tools > Developer Tools > Application Interfaces & Utilities*.)

## 1.9.2 Special Considerations for Date Formats

1. File layouts must be designed with the four (4)-digit year for all date fields.
2. Any expansions or conversions of files must add space for the four (4)-digit years as appropriate.
3. All birth dates must be expanded to a four (4)-digit year (CCYY).
4. Avoid hard coding of the century date. When a program has hard coded date routines in it, attempt to eliminate these routines.
5. Do not use 1999 as an open-ended date. Do not use other potentially valid dates such as 01/01/01.
6. When initializing dates to Zeroes or Nines, use the entire date field, not just the year to determine if it is a valid date.
7. Include the century when performing date comparisons.
8. Do not use two position years as part of record keys in VSAM files, GTM tables or elsewhere.

When designing a system, which will require processing control dates, every effort must be taken to derive this date using program logic rather than date parameter cards.

The following guidelines should be used when considering the addition of a parameter card to a new or existing job:

1. JCL statements are not considered control cards and should not be handled as such. If different options are needed, separate jobs should be created.
2. Tape numbers should not be set up as a control card 0289 input. Bypass non-label processing or use label=expdt=98000.
3. If you are dealing with an out-of-house package, add program to the JCL to generate the control cards using existing subroutines.
4. JCL data that changes once a year should be updated by programming with a "JCL change to the 0289 permanent library."
5. Input data that is to be supplied to Scheduling should be set up as a table on CARDLIB to call in this data at each run.
6. All dates must be calculated through the appropriate date shell.
7. Any parameter card that cannot be automated must be justified.

This justification is free form and should be submitted at the same time the production control log is sent to Scheduling to make the documentation change. If justification is not received, the documenter will be

rejected. Included in the justification should be the reason the parameter card is necessary and why it cannot be automated.

When a parameter card is required by a given system and that card is used in several places within a given system, the Parm Card should be passed from one-step and/or job to another by placing the Parm Card in a dataset using an IEBGENER. Thus, the Parm Card will appear in only one execution deck and scheduling will only need to update the Parm Card in one place.

# 1.10 Data Names

Each system will define a standard data element to the dictionary to be used with all programs (e.g., Program A will not refer to Social Security number as "SSN" and program B call it "SOC-SEC-NR.").

All data element names that are contained in IMS databases and DB2 tables will be stored in the Corporate Data Dictionary.

# 1.11 Variable Data Storage

As a design philosophy, we will attempt to house variable program data external to the program to preclude unnecessary compilation. Tables, edit data, dates, and so forth should be stored externally and loaded at execution time. A suggested vehicle for this is the standard table maintenance system described in TS02L400-GTM System. (Refer to *Procedures & Tools* > *Host Tools* > *Developer Tools* > *Application Interfaces & Utilities*.)

# 1.12 File Design Considerations

The purpose of this section is to provide an explanation of the policies and guidelines to be followed when designing Host computer files.

All permanent master files will have file descriptions beginning with a 01 level. In the case of files used by CICS only or shared by CICS and batch, they must begin with 02. For AMB programs, all database and VSAM file layouts must begin with 02.

The file description will be defined to the most detailed element used in any program accessing the file. Due consideration must be given to editing (i.e., a payment field should be set up as a group level and defined in the elementary level. This allows the checking of the field for numeric or spaces and for performing arithmetic operations.) The file description must be checked and approved by the manager.

# 1.13 Standard Name and Address

All new files that have name and address data will be formatted according to the following standards:

1. Name will be 30 characters. Title (when applicable) will appear first, followed by first name or initial, middle initial (when present), last name, suffix (when present), and degree (when applicable). Title will not be used when a degree is present.
2. First address line will be 30 characters.
3. Second address line will be 30 characters.
4. City will be 20 characters.
5. State will be the standard two (2) character state code authorized by the U.S. Post Office.
6. Zip Code will be nine (9) numeric digits.

This design consideration will allow mailing labels and addresses on other computer printed correspondence, such as EOBs and checks, to be uniform.

# 1.14 Printed Output Design Considerations

## 1.14.1 Standard Mailing Labels

All mailing labels should, whenever possible, conform to Cheshire format and the post office department's suggestions. Standard Cheshire format is three inches wide by one inch deep. In order to conform to these requirements, the following rules should be considered:

1. No more than 30 characters should be printed on one line.
2. From three to six lines may be printed per label.
3. Each label should have as a minimum:

    a. Name
    b. One or more lines of address -- street, P.O. Box, or route number
    c. City, state, Zip Code

4. Identifying information, (i.e., Identification Number, specialty code, county code, etc.), if required, should be printed on the first line.
5. The last line on the label should contain city, state and Zip Code.
6. No more than two spaces separating state and Zip Code is preferred.
7. Blank lines should not be printed.
8. Whenever possible, labels should be sorted by Zip Code (major). This will facilitate postal handling and reduce mailing costs.
9. Special Cheshire paper (Stock #32047) is available for printing four-up labels. Four-up labels are recommended if feasible.
10. The use of gummed labels is not recommended.

## 1.14.2 Standard Report Header

This standard defines a standard report header that will be used on all printed output except special pre-printed forms.

### 1.14.2.1 General Discussion

The header as shown below will appear at the top of each page.

When defining a report, consideration must be given to the use of the common BlueCross Report Splitter. Refer to *Technical Standards — Applications > Tools > Host Tools > Developer Tools* for additional information.

| | | |
|---|---|---|
| Condition Report | Splitter | INFOPAC |

Report has identical data, single or multiple recipients.

### 1.14.2.2 Report

Report has multiple recipients for different segments of the report.

## 1.14.3 Report/Splitter

When Report Splitter is required, it will be included in the PROC for the job and will feed INFOPAC a file for display online reflecting which reports were written to each DD statement. Refer to *Technical Standards — Applications > Tools > Host Tools > Developer Tools* for the use of Automated Report Distribution and Report Splitter.

If no special customer requested date format is required, the date in format MM/DD/YYYY will be used.

The runtime will be placed directly under the date in the format HH/MM/SS (hour, minute, second) except for Easytrieve Plus.

### 1.14.3.1 COBOL and ASSEMBLER

The installation standard header will be coded in every program that produces printed output except when pre-printed forms are printed.

A standard end-of-form message (i.e., end of report XXXXXXXX-01) will be printed on a separate page, at least three spaces below the heading.

### 1.14.3.2 Comma Separated Files

When creating new management summary reports that produce totals and that are less than 2000 lines, the data for these reports should be generated to a comma separated file. Raw data (no headings) should be produced for the customer, and it will be the responsibility of the customer to download this data to a personal computer to create the reports that meet the specific requirements of the requestor.

In situations where this approach is implemented, it will be the responsibility of the I/S department that produces the data to provide a record layout of the file as a description of the data.

All questions specifically relating to the creation of comma-separated files should be directed to the Information Center.

## 1.14.4 Forms Alignment

The purpose of this standard is to establish a means for an operator to properly align pre-printed or special forms before the actual data is printed.

### 1.14.4.1 General Discussion

In the initial form development there must be a screened out character in the shape of an "L" outlining where the first character of the first print line must fall. The "L" will then be the instrument of appropriate form alignment. The "L" will appear on every page of the printed or special forms.

Proper alignment will then take place when the operator has the first character of the first print line resting in the angle of the "L" on the form.

Example: Joe Smith

## 1.14.4.2 Program Considerations

The program that generates the print for pre-printed or special forms will:

Cause an "X" to print in the first print position of the first print line before any actual data is printed.

The above "X" frame will occur three times before the actual data.

## 1.14.4.3 Special Notes

For the purpose of printer restart ability, a cumulative page count (without heading) will be printed on every page in the upper left margin. Any variation in the location of the page count will be documented in the documenter member of the print job.

This standard does not apply to check alignment. Several methods already exist to align checks and no standard will be imposed except that some method must be supplied for check alignment.

# 1.15 IMS Checkpoint/Restart Guidelines

## 1.15.1 IMS

If a program updates an IMS database, IMS checkpoint/restart MUST be used. Initial load programs are an exception (PSB must have PROCOPT=L).

If a program is "read only" to an IMS database AND is considered "long running," then IMS checkpoint/restart MAY be used. In this case, the usefulness of checkpoint/restart is the restart portion. What is considered "long running" may vary among different applications, but a rule of thumb is two hours (clock time).

### 1.15.1.1 DB2 (with or without IMS)

If a program updates a DB2 database AND reads or updates an IMS database, IMS checkpoint/restart MUST be used.

If a program updates a DB2 database (but does NOT read or update an IMS database), DB2 commits or IMS checkpoint/restart MUST be used. If DB2 commits are used, then the program must have its own restart logic.

If a program is "read only" to a DB2 database (but does NOT read or update an IMS database) and is considered "long running," then IMS checkpoint/restart MAY be used or the program could have its own restart logic. What is considered "long running" may vary among different applications, but a rule of thumb is two hours (clock time).

IMS restart "automatically" repositions IMS databases and GSAM input/output files. IMS restart does NOT "automatically" reposition DB2 databases. However, if the program is processing a DB2 database sequentially, IMS checkpoint can "save" the last key processed in the restart area to enable the program's own restart logic to reposition the DB2 database.

### 1.15.1.2 Checklist for Using IMS Logs

- ALWAYS specify (NEW,CATLG,CATLG) on the IMS dataset.
- ALWAYS use GDGs for the IMS log dataset.
- NEVER delete or un-catalogue an IMS log dataset. Let the GDGs roll off as you create more.
- If you specify "IMST" or "IMSP" as the first node of the dataset name AND "LOG" starting in position 20, your tape datasets will not be un-cataloged after one day if your job ABENDs. This is also true for GSAM tape datasets except that you specify "GSAM" starting in position 20. You may need these datasets later for a back out or restart.
- ALWAYS run the back out (if required) as soon as possible after an ABEND to release the locks on the IMS databases.

### 1.15.1.3 MQ

If you are running under IMS and using WebSphere MQ, you must run under IMS/BMP and use checkpoint/rollback logic. Simple MQ COMMITs and BACK Outs do not function under IMS.

Programs using IMS/BMP and WebSphere MQ must issue checkpoints at frequent intervals to be determined by the DBA staff. This will include times of no activity, specifically dealing with a WebSphere MQ return code of 2033, no message available.

If you are updating DB2 tables and/or IMS databases and also Message Queues, running the program under IMS BMP and utilizing IMS Extended Checkpoint (CHECKPOINT/RESTART) provides update synchronization of the queues, tables and databases.

DBA emphatically recommends using the IMS Extended Checkpoint (CHECKPOINT/RESTART) for update synchronization of applications updating both MQ and DB2. Otherwise these applications will have to attempt to duplicate and maintain the synchronization that is already provided by IBM Software via IMS commit coordination.

# 1.16 IMS Programming Considerations

The purpose of this section is to provide an explanation of specific program coding considerations for programs accessing IMS database files.

1. Procedures for coding new IMS programs (or converting existing program to IMS).
2. All newly developed IMS databases will be accessed through I-O modules. This will assure uniformity throughout the installation and facilitate any conversion to another database management system that might occur in the future. The responsibility for the development of the I-O modules lies with the application area, unless the Database Administrator determines otherwise.
3. Programs written using the AMB code generator may use the AMB Data Definition Interface (DDI) and "db obtain" commands to replace calls to an I-O module. The Software Developer will have a choice of either using the "db obtain" command, or calling an AMB/COBOL I-O module, which has been written to access the database. If the Developer wishes to use the "db obtain" command, database administration will generate a DDI for them. If the Developer decides to use an I-O module and one does not already exist, the application area will code the I-O module.
4. The Software Developer will fill out the PSB request email and submit to "Systems.Support-Data." The DBA will generate the test PSB within 48 hours of receipt of the PSB. Database Administration will file the PSB request. Refer to *Technical Standards — Applications > File Design > Enterprise Server – IMS > Design Considerations* for PSB naming conventions.
5. At the same time the PSB is created, the DBA will create IMS buffers for batch programs. These will be placed on IMSP.CDMS.CNTLLIB and will have the same name as the PSB and program. There will be no test version of the buffers. Test programs will access the production buffers.
6. If the program has update intent, the DBA will generate a GDG for the IMS log tape.
7. A representative from Database Administration should attend the program code review.
8. When modifying call logic in an existing IMS program and if PSB changes are required, a new PSB request form will be submitted to "Systems.Support Data."
9. All Program Control Blocks (PCBs) in the linkage section must be coded to their full length. This includes the key-feedback-area of each database PCB.

   Example:

```
01          DB-PCB.

05          DBD-NAME                PIC X (8).

05          SEG-LEVEL               PIC X (2).

05          STATUS-CODE             PIC X (2).

05          PROC-OPT                PIC X (4).
```

```
05          DLI-RESERVED            PIC S9 (5)  COMP.

05          SEG-NAME-FDBK           PIC X (8).

05          LGTH-KE-FDBK            PIC S9 (5)  COMP.

05          NO-O-SENSSEGS           PIC S9 (5)  COMP.

05          KEY-FEEDBACK            PIC X (?).
```

10. All segment search arguments (SSAs) will be coded by the Software Developer (except in the case of AMB). Careful consideration should be given to coding the SSA with the most efficient call structure.
11. It is required that each DL/I call be followed by a status code check. The program should check for specific status codes returned by DL/I (e.g., "ge," "gb," etc.). The only exception to this is the case of AMB, which provides "88 level" names for the most common DL/I status codes. AMB does allow interrogation of the actual status code returned by DL/I, if necessary. Proper DL/I status code checking will provide maximum flexibility in program handling of call situations. Software Developers are encouraged to become familiar with these status codes and their meanings.
12. IMS/BMP cannot run in DB2I. DB2I must run under IMS/DL1. DBAs will create back-out members in BC.CDMS.AMMSCNTL if there are multiple jobs that run the same program against different DB2 tables.
13. Because of IMS cleanup processing restrictions, all non-IMS files must be closed before the program issues a GOBACK. A "S0C4" will occur if all files are not closed.
14. Database Administration is responsible for setting up the mechanism with which AMB will generate SSAs, PCBs, and database segment I/O areas. DBA has control over how these associated data field names will appear in the computer program.
15. In programs that do both IMS updates and total updates, there is a potential problem of the database recovery and database integrity. The problem arises since IMS can do back outs of the IMS database updates, but total must be restored to its state at the beginning of the program. If such a program were to ABEND in a checkpoint-restart environment, the IMS database would be out of sync with the total database, and it would take a great deal of effort to restore them to a common "sync point." Any online transactions entered in the meantime would be lost.

# 1.17 DB2 Design Considerations

This section will outline general design considerations for DB2 SYST errors.

## 1.17.1 System Design

As a general rule, DB2 tables, which are accessed/updated online through CICS, will not be available to QMF queries during normal working hours. However, the manager of Database Administration in those cases where the databases are small and online response time is not a major issue can make exceptions on an individual basis.

## 1.17.2 Table Structure

The most efficient and accurate method to determine a table structure is to access the DB2 catalog. The test DB2 catalog is available to everyone. However, for reasons of database convention, the production DB2 catalog is only available to the Database Administration staff. Consult the IBM DB2 SQL reference for assistance in accessing the catalog for MVS.

The most efficient and accurate method to determine SUBSCHEMA structure is through AMB.

Both of these methods should be used on a routine basis.

# 1.18 DB2 Program/Query Design

The purpose of this section is to provide general guidelines for the design of DB2 programs and queries.

## 1.18.1 Program Design

IMS applications that access DB2 require special attention and must be brought to the attention of Database Administration.

## 1.18.2 Query Design

- Queries should be coded to request only the columns and rows required for the application. (No select * commands.)
- Query by primary key where possible. Fully qualify update and delete statements.
- Queries should be coded in a manner that will make maintenance as easy as possible. While it is generally a good idea to retrieve the required information in one query (to save I/O), if that causes the query to be overly complicated, the query should be broken into two or more queries that will accomplish the same task.
- It is better for SQL to eliminate unwanted rows and columns rather than application codes.
- Queries should be validated with QMF prior to insertion into application code.

---

**NOTE COBOL ONLY**

The purpose of this section is to provide guidelines for coding programs, which utilize the VSAM file access method.

The "FILE STATUS" clause, which while optional, should be used on all "QSAM" and must be used on ALL "VSAM" files. VSAM does not ABEND. When specified, the status of the file's I/O operations may be monitored by validating the file status key values; thus, ensuring correct execution before continuing processing. VSAM opens and closes as well as reads, etc.

When coding, you should avoid using the declarative section or "invalid" key option for these files as they supersede the status byte check.

---

# 1.19 RULEs Standards and Guidelines

## 1.19.1 What Is RULEs

The RULEs Database is a DB2 repository for storing table driven logic. The RULEs Engine applies active data from an application or user to table drive logic and returns applicable data for the given input.

**How is data stored on the RULEs database?**

Functionally similar data is stored under one of the following categories:

- RULE Types:
  - Used to group data of a similar function under a generic 12-key structure.
  - Limitations:
    - 12 keys max per RULEs Type
    - The keys can be a list of values, but a single key cannot contain a range of values
    - "Or" conditions are not supported between keys
  - Example of utilization: CSR benefit narratives for Blue Choice.

---

**NOTE** RULE types exist in the environment but are not maintained and should not be used for new designs.

---

- Code Group Types:
  - Used to group data of a similar function under a generic 12-key structure.
  - The keys can be a list of values and each key can have ranges.
  - "And/Or" conditions/relationships are supported between keys.
  - Limitation:
    - 12 keys max per Code Group Type
  - Example of utilization: Utilization Review RULEs for TRICARE (subsystem 8).
- Data Types:
  - Used to store data that has a one-to-one relationship between the keys and returned data.
  - Example of utilization: GTM table replacement.
- Scripts (RULEs Scripting):

- Used to group data of a similar function under an "unlimited" key structure by utilizing a simple six (6)-statement script language.
- Example of utilization: HEALTH script and return keyed data for AMMS Auto Adjudication Engine (AAE).

**The RULEs system also features:**

- History record generation
  - Automatically producing an audit trail of all data changes that occur during table maintenance
- Security controls
  - Providing the ability to restrict the maintainer of the RULEs data by DB2 subsystem (Unit through Production)
  - To limit the maintainer by individual RACF access and type of access such as data updates or migration approval
- Data Maintenance options
  Update with procedural migration, through automated utilities, moving the RULEs related data from environment to environment through the development life cycle
- Data refresh options
  - Upon user request, the system provides the ability to refresh Unit and System databases from Production
- Data retrieval performance options
  - Data can be kept in-core once it is retrieved (applies to Code Group types and Script types only)
  - Data can be retrieved from DB2 each time it is referenced
  - Data can be placed into Working Storage

## 1.19.2 Why Use RULEs

The primary driver for utilizing RULEs is to create an externally maintained, table-driven option to be deployed whether it goes to a customer or I/S maintained RULE.

## 1.19.3 When to Use RULEs

The decision to use or not use RULEs is an I/S decision. Our customers may provide input into their desire to support the RULEs data, but the final decision is based upon the technical architecture of the application and how best to solve the technical deliverable being considered.

## 1.19.4 RULEs Setup Options

Data Migration:

- Required for all tables except for data that is region specific (e.g., LinkTask for Desktops)

# 1.19.5 Balancing and Reconciliation Processing Requirements

## 1.19.5.1 Balancing and Reconciliation Defined

As defined in the Balancing and Reconciliation Concept, balancing is the act of accounting for all inputs, outputs, and exceptions related to a process at a distinct and reoccurring point in time. Reconciliation is the accounting for all balancing results, inputs, outputs, and exceptions across all interrelated processes at a distinct and reoccurring point in time. The purpose of Balancing and Reconciliation is to ensure the integrity of the data supporting I/S and business operation production environments through the professional programming best practice of Balancing and Reconciliation (B&R). Failure to implement proper B&R processes allows for significant risk of business error, customer dissatisfaction, and internal/external audit exposure resulting from questionable data integrity. Therefore, Balancing and Reconciliation points are intra- and inter-system dependent and are required for ALL production applications.

Balancing and Reconciliation processes must be automated whenever possible. The Automated Balancing and Reconciliation System (ABRS) is the balancing tool used for all automated processes and is discussed in the next section. If a B&R process cannot be automated, I/S Data Control must perform the B&R manual process.

However, good business judgment must be used in only applying B&R validation checks where they are needed. B&R is not to be performed for every file update but only at those critical processing points and cross-system process points to ensure proper data integrity and proper business operations. B&R reporting is intended to identify that something is wrong, not pinpoint what is wrong. B&R validations isolate the error to an identified set of processes or procedures that have occurred since the last B&R validation point.

## 1.19.5.2 Automated Balancing and Reconciliation System (ABRS)

To accomplish the Automated Balancing and Reconciliation Technique, BlueCross uses the Automated Balancing and Reconcilitation System (ABRS) as the corporate balancing validation system. Both Host and Non-Host applications must use ABRS for balancing. ABRS is a system that has the ability to compare data from several sources and make a decision whether an out-of-balance condition exists. ABRS interfaces with flat files that exist on the Host. Balancing & Reconciliation (B&R) Specifications must be created for all production systems that contain B&R processes/procedures.

These specifications will include:

- The proposed ABRS balancing rules.
- Detailed, step-by-step workflow procedures for any proposed manual balancing or reconciliation processes along with a justification of why they could not be automated.
- Out-of-Balance (OOB) Handling Specifications with identification of any related tolerances.

## 1.19.5.3 What Should Be Balanced

The use of the ABRS for Balancing and Reconciliation must occur for, but is not limited to, the following types of conditions:

- Major business process change from one process to another
- Files crossing platforms
- Files moving between external and internal networks
- Major data store loads and unloads
- Inventory paid against value of checks cut
- Any file where data has been transformed

Balancing and Reconciliation involves the comparison of inputs, outputs, and exception counts of items such as documents or dollar amounts. Examples of items that must be balanced are, but not limited to:

- Premiums
- Checks
- Claims
- Faxes
- Files
- Payroll Data
- Financial Data
- Amount Fields

## 1.19.5.4 Applicability

This standard applies to all systems. Changes to existing systems must be reviewed by the Work Request team for Balancing & Reconciliation considerations to determine whether adequate controls exist. When deficiencies are discovered, appropriate action must be taken to ensure the integrity of the data and the continuity of processing.

## 1.19.5.5 Required Method of Balancing

There will be total reliance on the ABRS to determine if an out-of-balance condition exists. If an out-of-balance condition exists, the ABRS will set the appropriate condition code, which the Work Request team can use to determine if an ABEND should occur. Refer to *Procedures & Tools > Automated Balancing and Reconciliation Tools > ABRS (Automated Balancing and Reconciliation System).*

## 1.19.5.6 Application Development Team Responsibilities

### Defining Balancing RULEs

The balancing rules are specific for each job and balancing step. The Work Request team will identify balancing requirements, rules, and the balancing process. These rules allow an audit to be completed between one or more controlled sources and the process requiring balancing. To describe the Work Request for the ABRS team, the Work Request team will have a JAD with the ABRS team to gather information needed for the balancing. The ABRS team will present the ACR Coding Request Form, containing the requirements gathered from the JAD, to the customer. The form includes the dates associated with the Work Request, inputs, balancing rules, and special instructions needed to complete the request. The ACR Coding Request Form will not only be used to describe new work, but it will also be used to describe changes to existing balancing processes.

The ABRS team takes this information and creates a balancing process within ACR.

## Balancing File Requirements

Host based flat files or report files are required for ABRS balancing.

Update access to the following ACR History file needs to be requested from RACF.Admin for *Summary* validation:

TESTVS.OP02.UNITECH.ACRHIST

Update access to the following ACR History file needs to be requested from RACF.Admin for *Detail* validation:

TESTVS.OP02.UNITECH.DETHIST

## Balancing JCL Requirements

All areas using ABRS for balancing should reference the standard PROCLIB ABRSC900 located at BC.NDVR.PROD1.PROCLIB when constructing a JCL to execute program ACR9000.

Application areas need to override the following symbolic parameters according to the requirements in their jobs:

```
JNAM – JOBNAME (Defined to ACR)

STEP – Balancing STEPNAME (Defined to ACR)

INFOPACV - INFOPAC ID for error report

                    – (A “V” must be in the fifth position of the error INFOPAC
ID)

INFOPACU – INFOPAC ID for balancing report

                    – (A “U” must be in the fifth position of the balancing
INFOPAC ID)

ABRSDD1 – PROCLIB member with input DD names as defined by the ABRS team and the
corresponding dataset names (DSNs).

See the example below

//AMMSI151 DD DSN=BC.AMMS.AMMSJ15D.UNITECH.AMMS0013(&GEN080),DISP=SHR

//AMMSI194 DD DSN=BC.AMMS.AMMSJ15D.UNITECH.AMMS0402(&GEN130),DISP=SHR

ABRSFILE PROCLIBs for the ACR control, definition and history files:

ABRSC901 – contains production ACR files

ABRSC902 – contains non-production ACR files
```

## Requesting INFOPAC Numbers

ACR produces two reports in the balancing process. It is the Work Request team's responsibility to request two INFOPAC report IDs required for ACR balancing (one *U* report (ACRREPT) for balancing status and one *V* report (SYSOUT) for errors). The *U* and the *V* must be in the fifth position of the new INFOPAC numbers as the reports are used for ACR Reporting. INFOPAC IDs are assigned by INFOPAC Administration for each new balancing step. The INFOPAC report request form is found in the TSC Self-Service. Search for *INFOPAC Add* under the Service Request tab. Examples of the INFOPAC numbers are AMMSV001 and AMMSU001.

## Validation Environment

To ensure data integrity of critical processes, all changes made to an application that affect balancing, or new development that requires balancing must be validated in each environment Unit, System and Qual. The validation of balancing processes is the responsibility of the Application Development team working with the ABRS team and System Support teams.

The balancing rules are stored in an ACR VSAM file. The test ACR VSAM file name will be supplied by the ABRS team. The test ACR VSAM file is included in the sample JCL balancing step provided by the ABRS team. The Work Request team must provide to the ABRS team test data for each balancing process and follow normal validation procedures to verify balancing results.

## Manual Balancing/I/S Data Control

If an application has both automated and manual balancing procedures, the application is required to supply I/S Data Control with the necessary information needed to make changes to their current manual balancing sheet before any ACR changes can be moved to Production.

## Production Implementation

Once the Work Request team has completed its validation and is ready for Production, it must coordinate the production moves with the ABRS team.

## Production Balancing Additions and Modifications

Any new ACR balancing must be coordinated with the ABRS team.

Reports or files used in balancing CANNOT be altered in any way. All changes, including changes to the balancing process, must be coordinated with the ABRS team.

## Out-of-Balance Process

The responsible Work Request team describes any out-of-balance situations resulting from application problems on the Production Control Log. Out-of-balance explanations from the application area are given to the manager of I/S Data Control and filed for Internal Audit.

## Rerun Procedures

If an ACR step needs to be RERUN to pick up previously stored totals from the ABRS History Database (e.g., totals from the prior run), then the word RERUN must be placed in the ACR PARM card as shown below.

```
PGM=ACR9000,PARM='JNAM=BB10J71W,STEP=BB10S910,RERUN'
```

It is not necessary to use RERUN if there are no history items or if you need to pick new history item totals. Refer to *Glossary* > *Glossary/Definitions* > *Glossary* for the definition of *History Items*.

## 1.19.5.7 Output Control

Where exact balancing is not possible but reasonableness criteria can be applied, range/tolerance of figures should be provided in order to apply to the output data. Control reports, report total figures, etc., are to be kept to document the result of each production cycle and form a basis for controlling subsequent processing.

## 1.19.5.8 ABRS Team Responsibilities

### Setup

The ABRS team sets up the Internal, History, Calculated items, Balancing Rules, ABEND codes, and special instructions as requested by the Work Request team. The ABRS team provides the Step Name, DD name, and the VSAM file to the Work Request team after the balancing setup is completed for validating.

### Validation and Production Support

Application validation and production support procedures must be supplied for all balancing processes.

### Production Implementation

Implementing balancing rules to Production must be coordinated by the Work Request team.

# 1.20 Online System Design Considerations

## 1.20.1 Router Navigation and Standards

### 1.20.1.1 Super Router

The Super Router (INTRP000) is used to enable systems to communicate and pass data from one system to another. To add a program to INTRP000, it must first exist in a local system router (or if this is a new system, a local router created for system). When exiting a system, the super router automatically writes out the COMMAREA of the system that you are exiting to temp storage. When returning, it restores that COMMAREA before transferring control (XCTL) back to the original system module plus any other data that may need to be returned.

When requesting a new router to be added to INTRP000

- The local router COMMAREA must be no longer that 4000 bytes.
- Local routers will only contain PF key logic for returning using PF3, PF12, and Clear Keys. Exceptions will only be made for packaged systems.

The Super Router (INTRP000) must be re-compiled if any local system COMMAREA router has been changed and the local router exists in INTRP000.

Validation of INTRP000 routing involving a cross-regional process must be validated and migrated through the Quality Assurance regions. Exceptions will only be made for packaged systems.

Contact the Leveraged Systems to request a change to INTRP000 or to add a new router.

### 1.20.1.2 Application Router

All programs that utilize an Application Router for transferring (XCTL) to another program must have the same COMMAREA length. This ensures that all Application Router COMMAREA data being passed has the same format, thus preventing amends. When an application needs to change to a different size COMMAREA, the changed length may be transitioned in as programs are modified. It is up to the application area to ensure that the transition supports all active programs without abending.

## 1.20.2 Menu Driven

Online systems will be designed in such a manner that all options and transactions within the system will be menu driven. A menu screen will direct the operator to various functions by different transaction IDs or by allowing entry of a specified character in a specified area to select the function to be performed.

## 1.20.3 COMMAREA Usage

The online COMMAREA will be used for passing data from one program to the next where feasible. COMMAREA size is to be kept as small as possible.

Due to the use of the first 40 bytes of the COMMAREA in AMB programming, the first 40 bytes are to be reserved and not used for any other purpose.

# 1.20.4 Online System Reviews

The online Software Designers and Developers will follow all reviews specified in the other areas of this standard. These will include but not necessarily be limited to

- An Online Definition Review,
  which in many cases, can be included in the Design Review.
- An Online Design Review,
  which must include a copy of the online Design Document, a copy of the transaction flowcharts and decision tables, and the use of program function keys. Other documents may be required if pertinent to execution of the system.
- An Online Implementation Review,
  which will be held after all validation is complete and prior to project implementation.

In many cases the Online System Review can be held as a part of an overall project review. In this case, participants must concur with and sign off on the online review.

# 1.20.5 Data and Transaction Files (Design and Use)

## 1.20.5.1 DLI (IMS) or DB2 Databases

DLI (IMS) or DB2 databases must be used for all files that are to be updated in an online environment.

## 1.20.5.2 VSAM

Where DLI (IMS) or DB2 databases are not used for file organization, VSAM files must be used as the option specified, and there can be no online updating of the files.

## 1.20.5.3 Other Type Files

- Transient data
  - Transient data files are not to be used for new project development. Use DB2 files instead.
- Temporary storage
  - All temporary storage is to be written to auxiliary storage. "Main Storage" is not to be used. The 'SYSIDENT' option is not to be used.
  - Temporary storage will be used when multiple screens of data are to be displayed. This will allow an operator to page forward and backward without additional file reads to obtain the data. File reads must be limited to reading only enough data to fill one screen at a time.
  - As of November 1, 2013, the 'LENGTH OF' statement will be required for temporary storage queues in order to dynamically determine the length of the queues and prevent any length errors when writing or reading queues. This standard applies to any new or existing programs modified after that date.

## 1.20.6 Includable Code

On many data entry projects, code must be written that can be used in both the online and batch programs. When this is needed, shareable code will be written for edits and some other functions. Systems should be designed with this capability in cases of this nature.

## 1.20.7 CEC Batch

Since a large portion of the claims received by BlueCross are electronic media claims (EMC), special considerations must be given to this method of claims entry.

## 1.20.8 Telephony

Telephony follows all of the existing online standards with the exception of the CTI/Callpath transaction for interaction with the telephone switch. The Callpath signon and detail screens perform telephone functions, and some of the specific PF keys are functionally predetermined by the specific telephone switch vendor.

## 1.20.9 Automated Claims Processing Systems Mill

The standard Automated Claims Processing Systems Mill (ACPSM) ACPSM must be placed within any program that creates a 3270 screen accessed by an ACPSM application. The "include" contains comments to provide an alert that the screen is accessed by an ACPSM application and will appear on Endevor scans when doing program research. When a 3270 screen is to be changed, the PGBA, Commercial Business and LS Non-Host areas must be notified for possible impact to ACPSM applications.

## 1.20.10 Data Migration

If application data is maintained in a test region and migrated to Production, the preferred migration method is to use a batch extract and load process. If an application has the requirement that data must be migrated in a real-time (i.e., immediate) manner from test to Production, the following standards apply:

- Data going to Production must be migrated to QUAL first.
- Changes must be specifically “Approved to move” in an auditable manner before migration to QUAL may occur.
- When data is to be moved to Production after the move to QUAL, changes must again be “approved to move” in an auditable manner before that migration.
- Separate application security must be put in place for the ability to move data to Production. Both the Production “approve to move” function and the request to initiate the production migration (if separate from the approval) must be secured separately from other application functionality.
- Units of work must minimize locking impacts. This must be managed against the need to commit at logical points so that interrelated data can be correctly rolled back in case of failure.
- Migration of data to QUAL and then to Production without validation in QUAL is discouraged unless it is an emergency.

- Migration of data during peak hours (10 a.m.–12 noon and 2 p.m.–4 p.m., M–F) is strongly discouraged unless there is an emergency. In particular, this applies for the following:

    - The data being migrated impacts time-critical functions (e.g., HIPAA 270/271 transactions).
    - The data being migrated exceeds volume limitations as determined by performance and load testing. Refer to *Technical Standards — Applications > File Design > Enterprise Server – DB2 > Design Considerations* for additional information. In particular, load testing for this process must be coordinated with Tech Support and the DBA area.
    - A batch migration method must be created as a backup process and for data volumes exceeding the determined limit.
    - On failure, it is recommended that any temporary tables be cleaned up, and the migration request be resubmitted once the problem is resolved. The application needs to be prepared to handle previous partial success. Any cleanup performed after the fact must be done in an auditable manner.
    - Refer to *Technical Standards — Applications > File Design > Enterprise Server – DB2* for additional standards.

# 1.21 Online Screen Standards

## 1.21.1 Screen Elements Required On All Screens

The map name will be placed in the second position of the first row.

---

**NOTE** AMB screens default to this position. It is not necessary to skip one position when defining the map name under AMB. Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > AMB Naming Conventions* for the map-naming standard.

---

Use the blue color attribute. See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

## 1.21.2 Panel Title

A title will appear centered on the first line of every screen. It will contain the four (4)-byte transaction code followed by a short (but meaningful) description of the screen data.

For example: HCSR - Health Care Service Records

Use the yellow color attribute. See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

## 1.21.3 Date

The current processing date will appear on the first line of the screen to the right of the panel title in the MM/DD format.

Use the white color attribute.

---

**NOTE** DO NOT include the century or year in this display. This is a display-only item not used for any processing. Therefore, there is no Year 2000 requirement to include the century or year. By continuing to display only the month and day, consistency across all screens will be maintained. See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

---

## 1.21.4 Time

The current processing time, expressed in eastern standard military time, will appear on the first line of the screen to the right of the date in the HH: MM format.

Use the white color attribute. See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

## 1.21.5 RPN Number

Regional processing numbers (RPN) are used to delineate blocks of business. RPNs must identify the same block of business throughout the corporation on all systems. RPNs must be present on all system screens where there is a possibility of more than one block of business being processed. See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

## 1.21.6 Page Number

The current page of data of the type being displayed will appear at the far right of the line on the screen immediately under the RPN Number.

Use the blue color attribute for the heading “page” and the white color attribute for the page number.

---

**NOTE**

If there is only one page of data of the type being displayed, the page element is not required.

It is not required that this element be of the format "page xx of xx" due to the potential I/O required to support such a display. Where this information is useful and available without excessive I/O, display in this format is encouraged.

---

## 1.21.7 Program Name

The name of the current program sending the map will be displayed in the second position of the second row.

---

**NOTE** AMB defaults to this position. It is not necessary to skip one position when defining the program name under AMB. Use the blue color attribute.

If multiple programs send the same map, the program name that sent the map will be displayed. In this case, the program name is considered data and will use the white color attribute. See Figure 1-1

---

**Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

## 1.21.8 Host Screen Access

All new screens should be programmed to allow direct access via a call string. This feature will allow screen scraping tools to have quick access to a particular screen without the requirement of navigating through several screen layers.

## 1.21.9 Host Screen Reference

DB2 tables must be used on any reference from screens. This allows ease of information transfer when GUI screens are set up and the referenced information is displayed.

## 1.21.10 Message Area

A message area will be defined on the line preceding the Navigation Bar/function key area at the bottom of the screen. Use a red or white color attribute as appropriate (See section 10.02.03.03 Color Terminal Standards). See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

## 1.21.11 Navigation Bar Area (For New Development)

For new development, “Point and Shoot” buttons will be used in place of Function keys (PF keys) on the bottom one or two lines of the screen. If not all buttons are viewable on the line(s), a **More=>** button must be used as the last “Point and Shoot” button on the line to toggle to the next line of buttons. If the More=> button is used, a **<=More** button should be used at the end of the last line of “Point and Shoot” buttons in order to toggle back to the original buttons.

Verbiage labels on buttons must be English derivations that are intuitive to a user. The labels should be protected to prevent type-over of the by the user. The Navigation Bar is to be used only for moving from screen to screen (“screen transition”) and not for PF keys. Existing PF keys will remain standard throughout the application and should be listed and explained on the application main menu. A button must be developed for the list of available PF keys called "PF KEY LIST" (Refer to the section *Function Key (PF key) Area* below.).

Alternate button colors starting with the first button.

See Figure 1-1 **Function Key (PF Key Area)**

in the section *Function Key (PF Key) Area* below.

# 1.21.12 Function Key (PF Key) Area

It is expected that PF keys will remain standard throughout applications but will be listed on the application main menu instead of the bottom of each screen. In existing applications where the Navigation bar has not been developed, place the function key descriptions on the last line of the screen. Use of only one line for PF key definition is recommended, but two lines of key definition is allowed.

The verbiage description of keys will be in the format 'FX=DESC' with the description length of four to eight bytes.

For the standard PF keys (Refer to the section *Online Function Keys* below.) the verbiage to be displayed is:

```
F1 = help

F2 = update

F3 = return

F4 = override

F5 = refresh

F6 = sticky

F7 = bkwd

F8 = frwd

F12 = exit

F23 = notes

F24 = menu – This function key should take the user to the menu for that
screen/application and that menu should allow F24 to access the next higher
menu.  For example, F24 from any RMIH screen takes a user to the RMIH menu, and
F24 from the RMIH menu takes a user to RMME.

Enter = prcss

Clear = exit
```

Once standardized PF keys (PF13-PF20) have been developed for an application, the verbiage to express these PF key definitions must be standardized. Only those keys that apply to the screen will be displayed.

Standard PF keys are not required to be displayed.

PF keys will be displayed in numerical order across the line. If two lines are used to display the PF key values, the keys are continued on the second line. As many as will fit will be displayed on the first line, and the remaining keys will be displayed on the second line.

Those on the second line will line up under those on the first line leaving no gaps or blank areas.

Enter will appear on the left before the PF keys. When there is enough space, clear will appear at the end of the PF key list. Use the yellow color attribute.

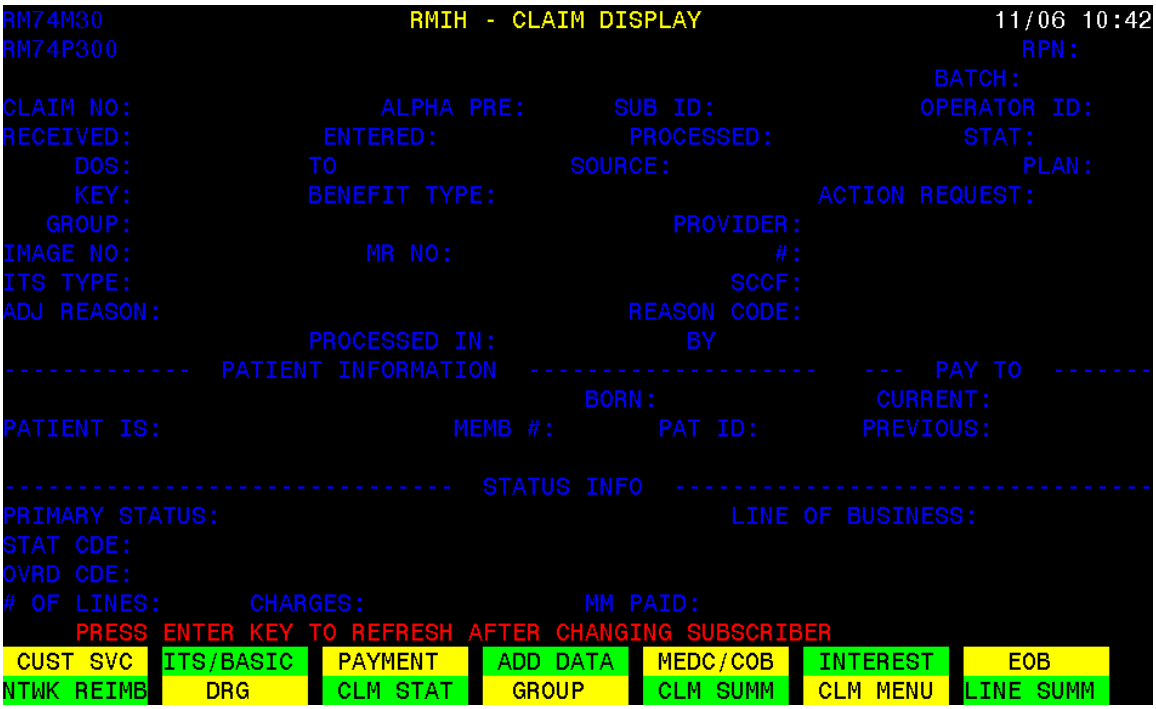



**Figure 1-1** Function Key (PF Key Area)

# 1.21.13 Help Information (For Future Development)

Screens must be designed to provide online information to the user about fields on the screen. Users must be able to place their cursor on a field and press F1 to view information about the field.

# 1.21.14 Requirements Specific to Screens by Type

## 1.21.14.1 Menu Screens

1. Selection description - Numbers or mnemonics to indicate selection choice will precede selection descriptions. Use the **blue** color attribute.

2. Selection entry field - The selection field will be to the right of the "select option" title. Use the **green** attribute.
3. Where the Navigation Bar has been developed (new development) and is being used in the application, standard PF Keys used in the application should be listed and explained on the main menu. (For information on "PF KEY LIST" Button, refer to the section *Navigation Bar Area (For New Development)* above.).
4. The F24 (menu PF key) must take the user to the menu for that screen/application and that menu should allow F24 to access the next higher menu. (For example, F24 from any RMIH screen takes a user to the RMIH menu, and F24 from the RMIH menu takes a user to RMME).

## 1.21.14.2 Data Entry Screens

Screen design must follow the type of data entry that will be performed and who will be performing the entry. There are 2 basic types of data entry.

1. Heads-Down Entry - This type of entry must follow the form from which data will be keyed as closely as possible. Since the person entering the data will be looking at the form and not the screen while keying, short data names and abbreviations will be used for data field names. This will allow entry of more actual data on a screen and less screen area used for data descriptions. Entry fields must be underscored to indicate the entry field positions. The underscoring must be cleared after the enter key is pressed.
2. GUI Screen Entry – This type of entry must provide more descriptive field names with the assumption the user will be looking at the screen, rather than a form, as a guide for entry.
3. Full screen editing must be used to provide the user with a view of all areas on the screen where input/reentry is required. Attention must be drawn to these edits. One example would be the use of reverse video red.

## 1.21.14.3 Inquiry Screens

Refer to the section *Color Terminal Standards* below for additional description of color usage.

1. Display screens (maps) must be designed in such a manner that the most frequently needed information is displayed near the top of the screen. Normally, the identification information will be shown first, followed by other data in descending order of importance.
2. The information on each screen must be intuitive based on the screen name. Overfilling the screen with data must be avoided.
3. Display screens (maps) will display messages that are meaningful to the user. Error codes alone are not to be used.
4. Field prompt (label) - A field identifier will be placed in front of each field on the screen. Field prompts will end with a colon and be right justified to align ending colons. Data fields will be left justified to meet the prompt alignment. A blue color attribute will be used.
5. Position all key fields at the top of the screen. Any display field that occurs more than twice must be in a column format.
6. Column headings - A column descriptor will be centered above the data fields. A blue color attribute will be used.
7. Group headings - A group descriptor will be placed above a group of fields with a common relationship (e.g., sponsor, patient). A turquoise color attribute will be used.
8. List (for summarization of data)

a. Column headings - A column descriptor will be centered above the data fields. A **blue** color attribute will be used.
b. Group headings - A group descriptor will be placed above a group of fields with a common relationship (e.g., sponsor, patient). A **turquoise** color attribute will be used.

# 1.22 Color Terminal Standards

## 1.22.1 Standards

Certain desktops have the option of displaying a terminal emulator with a white or black background. The color standards will vary depending on the background color. When appearing on a black background within a terminal emulator session, seven colors are available: red, blue, green, white, yellow, pink, and turquoise. These colors are specified by the color byte and are independent of the attribute byte. The color variation for white backgrounds is listed in parentheses if a variation exists.

Extended colors are to be used for all new screen panels. Following (Table 1-15) are the required uses of extended colors.

**Color Terminal Standards**

| Extended Colors | Required Uses |
|---|---|
| **Red** | Modifiable fields that are in error. For example, the terminal operator enters data into a field. The program determines that this data is incorrect. This field will now be turned "red" to indicate that there is an error. Error messages will be displayed in red. |
| **Green** (Dark Green) | Data fields for data entry or for data modification. |
| **Blue** | Field titles of normal interest value or un-modifiable data fields, which are to be treated as background data.<br>For example, a field title such as "address." A display only data field such as "I-20 at Alpine Rd." |
| White (Black) | Un-modifiable data fields of normal interest value.<br>Warning messages or special guidance messages. |
| **Pink** (Fuchsia) | Data fields of high interest value. |
| **Turquoise** (Teal) | Field, group, or column headings of high interest value. |
| **Yellow** (Burgundy) | Screen panel title and PF key prompts. |

**Table 1-15** Color Terminal Standards

Care must be taken in establishing the use of colors on the screen so that the purpose of color usage is achieved. The extended colors highlight and draw attention to the data being displayed on the screen. The customer may request the use of color, but the developer must understand the use of the screens and the data to make appropriate recommendations on the use of color.

# 1.23 Online Function Keys

## 1.23.1 Function Key (PF Key)

Refer to the sections *Navigation Bar Area (For New Development)* and *Function Key (PF Key) Area* above.

# 1.24 Content Management

The standard for content management of website information is IBM ***Web Content Manager*** (***WCM***). Any website being created or undergoing a change must use ***WCM*** as the content management tool.

Any existing websites using MyContentManager or any other content management tool will continue to be supported, but as those websites are modified, the content manager will be changed to ***WCM***. Any exemption from this standard must be approved by the Enterprise Architect Office and a waiver granted by the I/S Governance Governor.

# 1.25 Content Manager Moves Process

The purpose of the Leveraged System Non-Host (LS Non-Host) controlled move process is to allow Web "Content Managers" to have greater control over the websites for which they are responsible. By having the websites replicate on a set schedule, Web "Content Managers" know when to expect content and "content documents" to move from one server to another, and they can provide better customer service to their clients. However, there are times when a move needs to be made outside of the schedule. An Emergency Move process is in place to cover these situations.

For a normal move, the approval process between UNIT-SYSTEM and QA is done during the scheduled replication times where the customer can approve/disapprove the "content document" by visually verifying moves between UNIT-SYSTEM-QA during the replication times. Once Management and the Customer have approved a "content document," the "Content Manager" will mark it APPROVED, and the "content document" will automatically move to the next migration control phase based on the replication schedule. The "Content Manager" will track the "content document" along each migration step by spot-checking it ensuring correct data transfer between servers. At any time, automatic replication can be turned off to stop movement of data to "System," "QA" or "Production". A content manager can keep a document in "Ready for Approval" status, which only keeps it in "System" until a client has approved it.

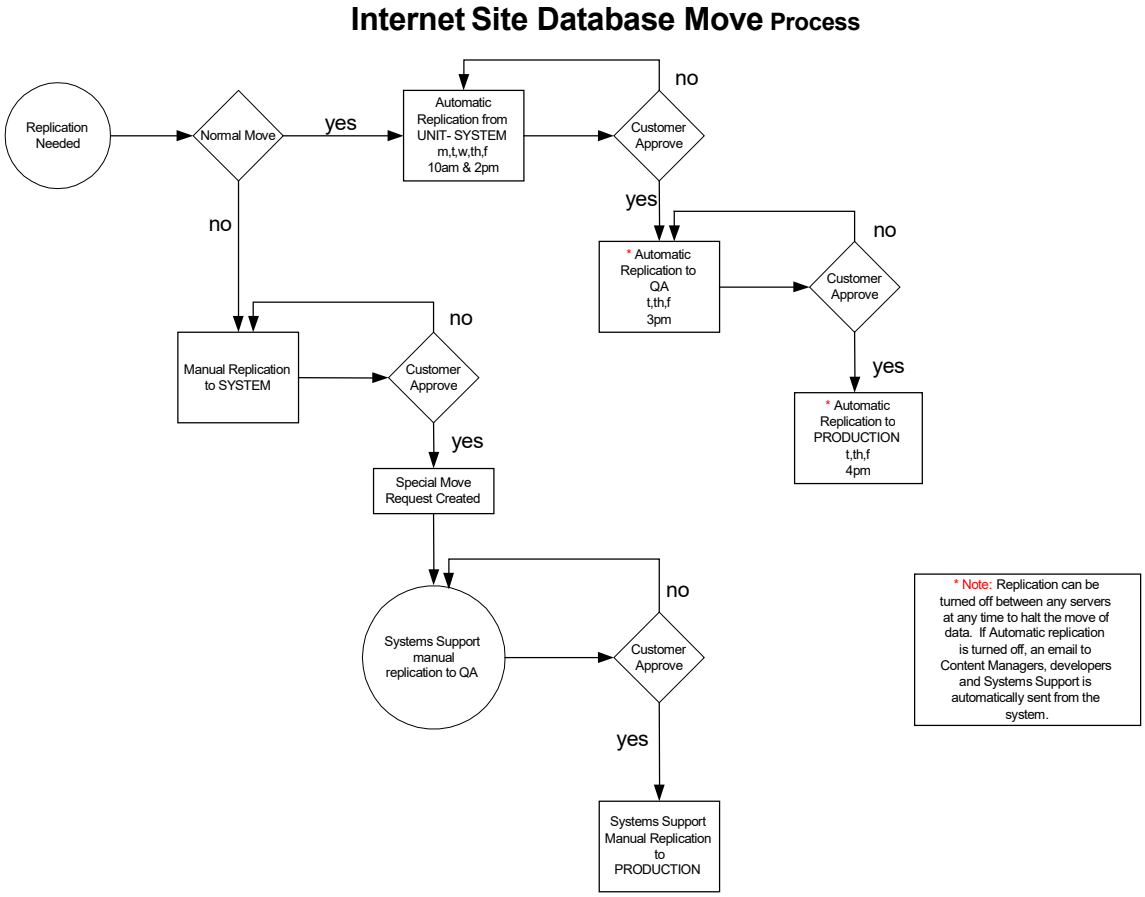



**Figure 1-2** Internet Site Database Move Process

# Chapter 2 Application Validation

This chapter contains standards that are applicable for validation application systems at BlueCross BlueShield of South Carolina.

# 2.1 Program Validation Considerations

Programs should be designed for ease of validating. Use of special validation and debug statements should be included in every program for use during initial validation. Following the modular concept, special test sub-programs can be used during initial validation and throughout the program life cycle to independently validate a module.

# 2.2 Test/Production Updates

The purpose of this section is to provide the sequence of events to be followed when installing a "PRODUCT" into production, and to provide procedures to be followed when performing test or production updates.

Test jobs (operating in ZEKET) will follow the same standards as production with the exceptions described below.

## 2.2.1 Production Installation Sequence

The following is a general checklist of the actions, which must take place in order to place a 'PRODUCT' into the production environment. The steps to follow to complete these actions are explained in subsequent paragraphs.

### 2.2.1.1 Regular Lead Time

If the installation timetable allows the required lead time (Refer to *Technical Standards — Infrastructure > Operating Systems > Production Acceptance > Production Acceptance — Job Documentation*) before the first production execution of a given "PRODUCT," the following events MUST take place:

1. Create/Update "EXECUTION JCL."
2. Create/Update Production "PROC."
3. Create/Update Production "CARDLIB,COPYLIB,VSAMLIB."
4. Create/Update "DOCULIB" MEMBER. Refer to *Technical Standards — Infrastructure > Operating Systems > Production Acceptance > Production Acceptance — Job Documentation* for a description of the contents of this member.
5. Create Production Control Log (PCL for production; TESTPCL for Test). Refer to *Technical Standards — Infrastructure > Operating Systems > Production Control Log* for an explanation of the steps to be followed to complete this action. Once a PCL is delivered to SCHEDULING, the production acceptance procedures described in *Technical Standards — Infrastructure > Operating Systems > Production Acceptance* are followed. Submit one PCL per action requested.
6. Create/Update Production "PROGRAMS." When updating existing programs, this action should be taken the morning of the first scheduled production execution.
7. Create/Update Report Distribution procedures (INFOPAC). Refer to *Procedures & Tools > Host Tools > Developer Tools > Host Development Environment* for further details.

### 2.2.1.2 Immediate Production Execution

If the installation timetable requires immediate production execution of a given "PRODUCT," the following events must take place:

1. During the lead time required to place a given "PRODUCT" into production, refer to *Technical Standards — Infrastructure > Operating Systems > Production Acceptance*, the "PRODUCT" must be submitted as a pseudo production job. Refer to *Technical Standards — Applications > Application Coding > Enterprise Server - JCL*. This does not apply to the Test System.

2. Create/Update Production "PROGRAMS." When updating existing programs, this action should be taken the morning of the first scheduled production execution.
3. Create/Update "EXECUTION JCL."
4. Create/Update Production "'PROC."
5. Create/Update Production "CARDLIB,COPYLIB,VSAMLIB."
6. Create/Update "DOCULIB" member. Refer to *Technical Standards — Infrastructure > Operating Systems > Production Acceptance* for a description of the contents of this member.
7. Create Production Control Log. Refer to *Technical Standards — Infrastructure > Operating Systems > Production Control Log* for an explanation of the steps to be followed to complete this action. Once a Production Control Log is delivered to SCHEDULING, the production acceptance procedures described in *Technical Standards — Infrastructure > Operating Systems > Production Acceptance* are followed. Submit ONE PCL per action requested.
8. Create/Update Report Distribution Procedures (INFOPAC). Refer to *Procedures & Tools > Host Tools > Developer Tools > Host Development Environment* for further details.

# 2.2.2 General Update Instructions — All Systems

Online test programs must be linked to BC.CIVS.TESTLIB.

When vendor packaged systems are added to our systems, any exceptions to the QA Standards Audit must be made known to the person in charge of the audits.

## 2.2.2.1 Execution JCL

1. When creating new execution JCL, the JCL must be added into Endevor and then moved through the Endevor environment into production. A Production Control Log (PCL for Production; TESTPCL for Test) must be submitted to Scheduling before the scheduling system will be able to execute the new job. Refer to *Technical Standards — Infrastructure > Operating Systems > Production Acceptance* for details and lead times for the PCL.
2. When modifying "existing" production JCL, the JCL must be retrieved from the appropriate Endevor production JCL library. After the JCL is modified, it can then be added back into Endevor and moved through the Endevor environment into production.
3. Alterations to the production schedule, whether for new or existing JCL, require that a Production Control Log (PCL) be submitted to Scheduling. The Scheduling department will then update the scheduling system accordingly. Refer to *Technical Standards — Infrastructure > Operating Systems > Production Acceptance* for details and lead times for the PCL.

Elements under Endevor that are shared with systems not under Endevor must be changed using Endevor procedures.

## 2.2.2.2 Text Scans

Interactive System Productivity Facility (ISPF) option 3.14 (SRCHFOR) should be used to scan an Endevor library for a specific text string.

Endevor-based text scans during normal work hours are not permitted. IT Business Systems (ITBS) Host SHOULD BE NOTIFIED of any jobs that are not conforming to this standard.

ELEMENT SCANS — Endevor can be used to scan for elements (e.g., copy books, programs, etc.) during normal business hours without adversely affecting system performance.

Endevor scans, if long running, should be run in class "C" with a priority of 0. Smaller scans may run in appropriate classes.

## 2.2.3 General Instructions for Systems Under Endevor

All development and maintenance for systems under Endevor will be done using Endevor procedures.

Retrieve with sign-out must always go to the production name. Retrieve without sign-out must always go to a test name.

Do not retrieve an element with sign-out until you are ready to make a change.

If an element has already been retrieved and signed out by someone else, you must coordinate your activities with that person. That person has control of what happens to that element until he/she is finished with it.

- Multiple copies of an element in Unit Test must all use test names. That means that when a second copy is retrieved, and the retriever coordinates with the person who signed-out the element, the production named member must be renamed to a test name. Also, any output from the source, like load modules, must be named to correspond to the name of the source that was used to create them.
- Only when an element is ready to be added to STAGE-1 will it be renamed to the production name. Output from the source, like load modules, must also be renamed to correspond to the name of the source that was used to create them.
- If the person having done the initial retrieve with sign-out is not the one ready to add it to Endevor, he/she must sign in the element with sign-out to the person ready to add. After the element is moved to Production it must be retrieved again by the first person, who must merge their changes into the retrieved copy.
- If modifications for more than one CCID are needed in STAGE-1 at the same time, they must be added with their own CCID. If they are moved together to STAGE-2, they must be moved "WITH HISTORY" so that the lines changed can be identified with the appropriate CCID.

Batch system validation must use BC.NDVR.SYST1.EXECLIB.

New elements may be added from any library, but modified elements must come from the same library and member they were retrieved to. Retrieved elements must be retrieved to C.NDVR.UNIT1 (libraries for non-AMB elements) and to BC.CADSGLB (libraries for PS elements).

The Endevor option to print changes only must be used by the user to print the changes for presentation at the Code Concurrence or System Support Acknowledgement Review to verify that the only changes that went in were those desired. Refer to *Application Systems Management > Application Systems Management Framework > Design, Development and Validation Phase > Validation* for further details.

Every Endevor action requires the input of a valid CCID and a meaningful comment.

The "DELETE INPUT SOURCE" switch on your add screen must be a "Y" unless justification is given to your manager for any exception. All Unit Test modules corresponding to an added element must be

deleted. Following the Add, if naming standards are followed and the switch is set to "Y," Endevor processors will automatically delete load modules and program source on an “ADD.”)

Any changes to be made to a STAGE-1 element require the element to be retrieved from STAGE-1.

An authorized person must do moves to STAGE-2 after receiving an Endevor move EMAIL form from their approver. Approvers and EMAIL forms vary by I/S department.

The practice of the separation of responsibilities requires that no one associate be allowed to create and approve code to move to Production. For this reason, approver groups are established and must consist of at least two associates. Based on the business need, exceptions can be granted to allow one associate in an approver group. However, the associate cannot reside in multiple groups.

# 2.2.4 Specific Instructions by Type — Endevor

## 2.2.4.1 Programs

- All AMB programs must be retrieved from Endevor to Enterprise Developer for zSystems (EDz) AMB using the Endevor Tools in the EDz Developer Toolkit.
- All AMB programs must generate and compile cleanly in EDz prior to moving to Endevor.
- If the program is a component of a composite module, the LINKDECK must be modified also and can be updated using ISPF and the Endevor panels.
- AMB batch programs must be added to Endevor using the Endevor Tools in the EDz Developer Toolkit. The Endevor processor group will take care of copying the generated artifacts to the appropriate test environment.

# Chapter 3 Application Systems Support

## 3.1 Operational Support & Monitoring

### 3.1.1 Production Procedures

#### 3.1.1.1 Production Claims Stripping Process

In order to meet the critical daily claims adjudication processing window, claims that cause an Abend condition are removed from the cycle as a work around to allow the adjudication cycle to complete. Claims that have been stripped during the daily process are reviewed and corrected for the next processing cycle. The goal of the claims stripping process is to re-establish the production cycle within the twenty minute time period.

---

**NOTE** These procedures only apply to non-GSAM (sequential) records.

---

#### Decision to Strip Claims from Cycle

There are separate support areas for each line of business. These areas are responsible for supporting the daily claims production cycle. During the course of the adjudication cycle, when an Abend occurs, operations will notify the appropriate support area. The support area verifies Abend information provided by operations and also researches the abending job in JHS. After the support technician completes researching the Abend, a decision is made whether to initiate the claims stripping process. When the technician encounters user abends or SOC7's, the claims stripping process is initiated to save processing time. Other critical abends require intervention and system analysis.

#### Claim Strip Picture

Research the abending job using JHS. Locate the job by looking at the run-date and time. Also look at the condition code and the error-id information provided by JHS. Select the abending job from the JHS list. Obtain the following information, the name of the abending step, program name and Abend code.

Once the information from JHS is obtained, research the dump dataset to obtain the social security number/numbers from the abending claim. The social security numbers drive the claims stripping process. To find the dump dataset, refer to the dataset naming standards for your line of business. Also the abending job name must be known. Then using TSO option 3.4, search for the most recent dump dataset. Wildcarding the dataset provides all archived dump datasets for this production run. `Ex(BC.DMPX.JOBNAME.*.*.*)`.

#### Locating the Social Security Number/Numbers

D — View the abending job in JHS.

T — Document the abending step, program name and error codes.

- Locate the abending social security number using the DDNAME to search the dump. The DDNAME can be obtained from the abending proc or from retrieving the information from TSO option 10.27 JHS.

- Use the DDNAME to position to the previous and current record in the dump dataset. The record layout in the dataset can be mapped using the layout of the WIP file to locate the claim number and the social security number.

The file type of the abending dataset, as a general rule, can be determined by looking at the 5th position of the DDNAME.

D — VSAM

---

**NOTE** This process is not applicable for this file type.

---

I — Input or Output file

N — Input only

T — Output only

Once the social security number/numbers are located, this number or range of numbers will be used by the claims stripping jobs set up by your line of business.

When the appropriate claim strip job is located, the technician needs to scrutinize the JCL. Becoming familiar with the JCL aids the technician in the claim stripping process.

There are two claims storage files that are used during the batch processing cycle. These two files are swapped in an out during the cycle. Verify that you are selecting the proper storage file.

## File Stripping

1. Back up the storage file to disk before initiating the claims stripping process.
2. Using the back- up storage file, delete the social security number/numbers that produced the Abend. This modified file will replace the storage file used to process the daily claims cycle.
3. Place the stripped records into a separate file for research. Once researched, the stripped records will be inserted into the next day's claims processing cycle. A Project Manager from each line of business needs to be assigned to oversee the timely correction and reprocessing of the stripped claims.
4. In the process of stripping claims, a critical errors report is produced. This report is used to identify and log stripped claims. This report is generated automatically and issued via the email system using the mail-away function. The system managers and Project Managers from the respective lines of business will receive a copy of the email and be responsible for monitoring the timely correction and reprocessing of stripped claims.
5. Since there are specific claims stripping needs for the various lines of business, please contact your management for additional procedures pertaining to your line of business.

# 3.2 Test System Management

## 3.2.1 Procedures for Placing Test Jobs and Test Cycles in the ZEKE Test Scheduling System

The purpose of this section is to provide detailed information and procedures for placing test jobs and test cycles into the ZEKE Test Scheduling System (ZEKET). The basic procedures consist of:

- Adding DOCULIBs, JOBDECKs, and any necessary Proclibs to Endevor.
- Creating an on-call support group in the TSO Toolbox.
- Submitting TESTPCLs to Scheduling using TSC Self-Service.
- Obtaining access to the ZEKET.

### 3.2.1.1 Initial Setup of New Jobs

1. Create a DOCULIB for each job. It is suggested that you clone from production and make the necessary changes or refer to *Technical Standards — Infrastructure* > *Operating Systems* > *Production Acceptance* > *Production Acceptance — Job Documentation* for current DOCULIB requirements.

    a. For items that don't apply, you must specify NA. All topics and sections must be contained in the test DOCULIB just like production.

2. Move DOCULIB into the Endevor production environment using the Endevor type of DOCULIBT.

3. Create and move PROCLIBs into the Endevor production environment using the Endevor type of PROCLIB or PROCLIBT.

4. Create a JOBDECK for each test job as follows:

    a. Jobname must contain a valid system prefix, such as AMMS, etc.
    b. Last character of the jobname must be an **$** or **#**. Medicare Systems may use these characters as well as an @ sign in the 3rd or 4th position of jobname.
    c. Jobname must match PDS member name and DOCULIB name.
    d. Job class must be **CLASS=8**.
    e. USERID must be an authorized test USERID, such as AMMSTST (SC) or AMMSTS2 (WI).
    f. Be sure to include a **JCLLIB ORDER=(dsn)** statement to direct your job to the appropriate PROCLIB library if other than BC.NDVR.PROD1.PROCLIB.

5. Move JOBDECK into the Endevor production environment.

    a. Add to STAGE-1 of Endevor using Endevor type JOBDECKT.
    b. Continue promoting the JOBDECK to the Endevor production environment by following the same instructions for a normal production move. Test JOBDECKs will be written to the ZEKE Scheduling System (ZEKE) scheduling library, BC.SCHD.ZEKE.JCLLIB, when the JOBDECK is moved to the Endevor production environment.

## 3.2.1.2 On-Call Support Group in the TSO Toolbox

I/S Operations will monitor test cycles just as they do production. In the event a job or cycle abends, you must have an on-call group defined who will be responsible for resolving the problem. Below are instructions for creating an on-call group for your ZEKET jobs if one doesn't already exist.

From the TSO Toolbox System Callout on-call main menu, enter the following:

1. Option 3 (Update Support Area List) — to define the Support Area and contact information.
2. Option 2 (Update system call List) — to define additional on-call information and phone numbers.
3. Option 1 (Update Support Area Job List) — to identify jobs to be associated with the support group. The first time you attempt to add a job to a new Support Area, you'll first have to key the name of the Support Area as a job and then enter **ADD** on the command line. Then you can add other jobs as necessary.
4. Option 6 (Update I/S Employee Phone List) — to add employees to your Support Area. The first time you attempt to add an employee to a new Support Area, you'll first have to key the name of the Support Area and then enter **ADD** on the command line. Then, you can add employees as necessary.
5. Option 5 (Update On-Call List For Jobs) — to specify starting dates/times for the primary and secondary on-call personnel.
6. Option 4 (Find Person On Call For Abending Job) — to ensure that you've set up your on-call Support Area correctly.

---

**NOTE** See Support Area **TCG** for an example.

---

## 3.2.1.3 Submitting TESTPCLs to Scheduling

Send a TESTPCL form found on TSC Self-Service for each job.

---

**NOTE** Be sure to include an activation date in the comment section. This is the date that you want the jobs to begin running from ZEKET.

---

## 3.2.1.4 Obtaining Access to the ZEKE Test Scheduling System (ZEKET)

To obtain access to the ZEKE Test Scheduling System (ZEKET), send an IT Business Systems Help form and include the following information:

- Name(s) of persons needing access
- TSO USERID(s) of persons needing access
- Reason for access

Once you've been authorized to ZEKET, you can sign on to ZEKET by following the procedures as outlined on the email conference FAQZEKE.

# 3.2.1.5 Additional Supporting Documentation

## Restarting a Job

Copy the JOBDECK from BC.NDVR.PROD1.JOBDECKT and put into BC.SYSTEMS.STAGING.JOBDECK. Make the necessary changes to the job in BC.SYSTEMS.STAGING.JOBDECK.

On INFOrm, locate your abended job ticket by the jobname. Use the batch response code and enter your response as to how you want the abend handled, such as restart from staging, fdone, etc.

## Modifying JOBDECKS in the Cycle

Follow the same procedure as a new production move.

## Temporary Changes to JOBDECKS

Copy the JOBDECK from BC.NDVR.PROD1.JOBDECKT and put into BC.SYSTEMS.STAGING.JOBDECK. Make the necessary changes to the JOBDECK in 'BC.SYSTEMS.STAGING.JOBDECK.

Send a TESTPCL through TSC Self-Service to Corporate Data Center (CDC) Operations.

**NOTE** There is a place on the form to indicate an override test job. This temporary change is good for one run/execution only.

## Marking Jobs Complete

On INFOrm, locate your abended job ticket by the jobname. Use the batch response code and enter your response as to how you want the abend handled, such as restart from staging, fdone, etc.

If a job is not in Abend status, send a TESTPCL through TSC Self-Service to CDC Operations.

**NOTE** In the request, be sure to specify that the job be in the ZEKE Test Scheduling System (ZEKET) and provide your name, your extension, and the jobname.

## Support

For all questions, issues, or problems related to ZEKE or ZEKET, please submit an IT Business Systems Help form.

# Chapter 4 Application Development Support Tools

This chapter contains standards that are applicable to the application development support tools used at BlueCross BlueShield of South Carolina.

# 4.1 Host Application Languages

## 4.1.1 General Discussion

COBOL can be developed in two ways:

- Directly using Enterprise COBOL syntax
- Generated using AppMaster Builder (AMB) S-COBOL

This section defines the guidelines for the maintenance and support of Enterprise COBOL and AMB and its generated COBOL application programs. These standards are designed to apply to both vendor-supplied code and code written by BlueCross BlueShield of South Carolina (BlueCross) personnel. For standards on Other Host languages, refer to *Technical Standards — Applications > Application Coding* and this section *Host Application Languages*. For standards on Non-Host Programming Languages, refer to *Technical Standards — Applications > Application Coding >Application Development Standards* and *Technical Standards — Applications > Application Coding > Platform Development Standards*.

### 4.1.1.1 Programming Languages

Enterprise COBOL for z/OS is the standard language.

If using AMB to generate COBOL, the use of the Micro Focus product Enterprise Developer for zSystems (EDz) may be required. AMB is the upgraded version of APS, which is installed and executed via Citrix within the EDz. AMB allows for generation and compilation of COBOL code on the PC for use in the Host environment.

The AMB preprocessor will be set to generate structured COBOL. The resulting generated source must be compiled through the Enterprise COBOL for z/OS compiler. This standard applies to both new development and whenever an existing AMB program is modified.

EasyTrieve Plus is acceptable for reporting programs in informational subsystems (excluding the Information Management System [IMS]).

New development in any other language, such as those listed below such as those listed in the section *Special Cases* below, must have a waiver approved by the area VP, the I/S Governance Governor, and the Sr. VP of ICT Infrastructure Management.

### 4.1.1.2 Special Cases

Assembler languages may be used only for situations not readily handled by AMB or COBOL. In these cases, the assembler language routine will be written as a subprogram to be called by an AMB or COBOL root program.

## 4.1.2 AMB/COBOL Online Considerations

### 4.1.2.1 AMB Coding Guidelines

#### Coding Guidelines Common to Online and Batch

## NOTE COBOL ONLY

The purpose of this section is to define the guidelines for maintenance and the support of COBOL application programs. These standards are designed to apply to both vendor-supplied code and internally developed code.

### Identification Division

- Included items:
  - Program ID – Eight character Program ID
  - Author – Name of group that is developing or supporting the program
  - Installation Data Center name
  - Date Written – Date Program Written (e.g., June 14, 1999)
  - Date Compiled Assigned by compiler
- Remarks

  The Remarks key word is no longer valid in COBOL. In place of this key word, an asterisk will be placed in column seven to denote remarks.
  For common coding standards, see below in this subsection.

```
*
* AMB APPLICATION:
*
* FUNCTIONS:
*
*
*
*
* INPUT SUMMARY -
*
* PARMS:
*
* FILES:
*
* SCREENS:
*
* OUTPUT SUMMARY -
*
* FILES:
*
* REPORTS:
*
* SCREENS:
*
* RETURN CODES:
*
```

```
* EXITS:
*
* NORMAL:
*
* ABNORMAL:
*
* ABEND CODES:
*
* GTM TABLES USED IN THIS PROGRAM:
*
* INCLUDE/COPY MEMBERS UDED IN THIS PROGRAM:
*
* SPECIAL CONSIDERATIONS:
*
* MODIFICATIONS:
* MOD-ID PGMR DATE DESCRIPTION
*
EJECT
```

## Environment Division

### Configuration Section

No exceptions or restrictions have been identified for the Configuration Section.

### Input-Output Section

External data names will be structured in accordance with *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > COBOL DDNames.*

## Data Division

### Data Name Development

In order to improve program maintainability; wherever possible, assign a unique prefix to each 01 Level and every data name under it.

The prefix, with the exception of 88 Levels, should be a meaningful acronym.

Examples:

- WS-Working Storage
- LS-Linkage
- WA-Work Areas
- Lt-Literals

**Data Name**

When creating a data name, consider what information it conveys to the next Software Developer viewing the program for the first time. Choose names using terms from the Program Specifications instead of names describing relations to the solution of the problem. The characters comprising the Data Name should not be minimized rendering the meaning of the name obscure.

**File Section**

The following information is provided for the File Section:

- Wherever possible CODE FDs in the same order as Select/Assign clauses.
- Recording Mode is the physical record description.
- When variable length records are used, code Record contains n to x Characters, where n is the size of the smallest record and x is the size of the largest record.
- Block contains zero records must be used for all input and output files.

**Working Storage**

Wherever possible, Working Storage should begin with the code illustrated below.

```
WS01   Filler X(44) V 'PROGRAM XXXXXXXX Working-Storage Begins
Here'
```

```
(XXXXXXX equals the program name specified in the PROGRAM-ID
clause)
```

```
Wherever possible, Working Storage should end with the code illustrated below.
```

```
WS01   Filler PIC X(42) Value 'Program XXXXXXXX Working-Storage
Ends Here'
```

```
(XXXXXXX equals the program name specified in the PROGRAM-ID
clause)
```

### Level Entries

All level numbers will be two digits. If explicitly coded, the first subordinate level under a "01" will be level "05." Each additional subordinate level will be incremented by five. (Original programs only)

### High Activity Data

Wherever possible, place high activity data at the beginning of the Working Storage for efficiency.

### PIC Clauses

Wherever possible, align all PICTURE, USAGE and VALUE clauses on the same column. Define numeric data items as signed. This eliminates a machine instruction to set the sign digit when the item is computed. Also when defining numeric data, ensure the length of the field allows for future expansion (e.g., dollar field expansion).

Non-displayable Hexadecimal values are not to be directly embedded within COBOL or AMB source. Hexadecimal Notation Format will be used in VALUE clauses and literals to specify non-displayable hex values.

Example 1:

```
  05 WS-HEX-01    PIC X VALUE X'01'.
```

Example 2:

```
   IF FIELD1 = X'01'
```

All existing non-displayable hexadecimal values should be converted to the formats above when a program is modified.

### Field Length

Keep fields used together the same length and decimal points aligned between sending and receiving fields whenever possible.

### Indexing

Whenever possible, use indexing rather than subscripts. Index names contain a common prefix (e.g., X-).

### Indicators

Do not use numeric fields for codes and indicators that can be defined as PIC X. For example, code with values 1, 2, or 3 are defined as alphanumeric. If arithmetic is not performed on a given field, define it as PIC X rather than PIC 9.

### Variable

Avoid use of variable length fields (e.g., OCCURS DEPENDING ON data field). The program executes special code each time a variable length data item is referenced. The code to manipulate variable length data is less efficient than for fixed size data items and therefore slower.

### Linkage Section

Although there are functional differences in the use of this section in the online and batch environments, the same rules apply to both.

### IMS Database Considerations

IMS database segments are maintained as a COPYLIB member on Endevor with a member name equal to the segment name (Refer to *Technical Standards — Applications > File Design > Enterprise Server – IMS > Design Considerations*.) and will be automatically included via the AMB subschema.

Database Administration will coordinate any changes to segments.

## Procedure Division

A program consists of a control routine (mainline) and performed paragraphs structured in a top-down hierarchical fashion. Any paragraph being performed physically resides below the paragraphs that perform it; do not use upward PERFORMS.

Avoid duplication of function or duplication of code. For example, read a file in ONE paragraph but have multiple performs of that paragraph.

Each module has a single entry point and a single exit point except as required by vendor software.

Procedure Division section name follows the same rules as paragraph names.

## Paragraph Names

**NOTE** These standards apply only to internally created code.

1. Paragraph names are as descriptive as possible.
2. The following standards are to be followed wherever possible:

    a. Prefix paragraph names with a sequence number that is from four to eight digits long followed by a dash character.
    b. Within a program, all prefixes should be the same length.
    c. The paragraphs should be arranged in ascending numerical sequence in the program.

d. Hierarchical numbering within a perform range is encouraged (e.g., Para 1000 performs Para 1100 and 1200, Para 1100 performs Para 1110 and 1120, and so on). The lowest level sequence numbers should be initially assigned in minimum increments of 10.
e. Within an AMB stub, the second node of the paragraph names should reflect the last four characters of the stub name for ease of identification.
f. Paragraph names are on a separate line from the coding.

3. Wherever possible, add a flower box before or after the paragraph name with a brief description of the paragraph's function.

## Perform

PERFORM syntax is consistent within a program. The following are permitted:

- PERFORM nnnn-paragraph-name
- PERFORM nnnn-section-name

## Other Coding Guidelines

- Do not use ALTER statements.
- Eliminate any paragraphs or sentences that are not executable by the program.
- Use GOBACK at the end of the mainline.
- Do not use the STOP RUN statement.
- Use SEARCH ALL rather than coding your own binary search.
- Internal sorts are not allowed.
- The SUPPRESS parameter on the 'COPY copy-book' statement is not allowed.
- The "ENTER" statement will not be used except within special user rules approved by IT Business Systems (ITBS) Host.
- AMB Programs must not use COBOL Procedure Division Copybooks unless the code is shared with programs written in COBOL.
- AMB code should not reference or modify AMB-generated variables or labels, except for those created and referenced by the AMB DB-processing statements.
- Use of Nested Copy statements is not allowed.
- The following instructions that shall not be used in a copybook or stub that may be used in both batch and online modules are:

    - Input/Output
    - Open
    - Close
    - Read
    - Write
    - Accept
    - Display
    - Start
    - Rewrite
    - Stop run
    - GOBACK
    - Delete

### Dynamic Call Statement

The dynamic call statement is used when a called subprogram is not link-edited with the main program but is instead link-edited into a separate load module, and at runtime, is loaded only if and when it is required (that is when called).

The dynamic call is formatted using a Working Storage data name that contains the value of the program being called. It is best to contain the program name within the defined data name for ease in identifying the called program. The program in a dynamic call statement in the procedure division is not contained within quotes. Moving a value to the defined Working Storage data name being called in the procedure division is discouraged.

```
Dynamic Call Example:

WORKING-STORAGE

01  WS-CORPL201                PIC X(08)    VALUE  'CORPL201'.



PROCEDURE DIVISION

CALL WS-CORPL201 USING...
```

Dynamic calls are required. Static calls must be converted to dynamic calls and link decks eliminated. The conversion of static calls to dynamic calls must be made when maintenance is done on an existing program. The following is the only known exception where STATIC CALLS are technically required:

Software used by BlueCross BlueShield of South Carolina but not developed by BlueCross (Vendor supplied software) may have a requirement to be called statically. Vendors supplying statically called software should be contacted to see if they have a version of software that can be called dynamically. Link decks may be required and are allowed to be used to support vendor code.

Stand-alone link decks are disallowed unless needed for installation of vendor code. (See above in this subsection.)

# 4.1.2.2 Online Coding Guidelines

The following standards in this section are to be followed for ONLINE development. All standards are effective immediately except for those denoted “For Future Development” which are to be adopted as any new upgrade or development work is done. An Enterprise Architect must approve any exceptions to these standards.

## CICS Program Coding

### Layered Framework

A framework must be utilized in which modules are designed for integration of differing applications across all channels. The goal of this framework is the capability to quickly design or customize a new interface without affecting the core application.

1. Applications must be assembled utilizing pre-existing modules where possible
2. Reuse of existing components must be a primary design goal
3. Newly constructed modules must be designed with reuse in mind
4. The major layers of the application must be cleanly separated (as noted below)

For example, as technology changes, and BlueCross moves to displaying information on devices other than terminals, presentation of the information may require a change, but the other core applications should not require changes. To accomplish this, a Layered Framework of development must be utilized. The layers must be separated in a manner where any of the layers may be replaced as needed without affecting the other layers. These layers include:

**Presentation Layer** - This layer handles the rendering of data to an interface device such as a browser interface or 3270 interface. Enhancements, additional support capabilities, or external system integration must be accomplished without adversely affecting the existing application processing logic.

**Business Logic Layer** – This layer handles the business process required by the customer such as claims adjudication. This layer is the most durable and must not require changes, as technological advances require adjustments in the other layers.

**Data Layer** – This layer handles the storage and retrieval of data such as IMS files or DB2 databases. The segregation of the data allows for a flexible and scalable data repository independent of the Presentation and Business Layers.

## Transaction ID Assignment

1. New transaction IDs will not begin with the letter **"C."** This letter has been reserved for IBM systems usage.
2. When selecting a new transaction ID, it must first be verified through the GTM CU Table that the selected transaction ID is not currently assigned. The GTM CU table is a reference table that identifies the function of the transaction and the system area that supports the transaction. This table is also used for analytical reporting of CICS transaction usage. Technical Support owns this table and is responsible for maintenance and updates.
3. When a new transaction is to be added to the GTM CU table, the following information must be provided: a new transaction ID, the function of the transaction, the System ID (Prefix) that the transaction belongs to, and the responsible manager's two-digit code. The manager's codes can be found in the CARDLIB - CDMSD505. Send an email to CICSADM to specify the new TRANSID.
4. Creation of a new transaction ID requires approval from an Enterprise Architect.

## Security

Each system, such as AMMS, PIMS, TMCS and RULEs, must utilize a uniform security layer and no additional security layer should be added.

For example, the security standard for AMMS is the Automated Medical Security Menu (AMSM).

AMSM is in the main online application router program RM70P000. Any transactions defined to route through the router will check AMSM security using a call to the RM75D124- AMSM security access module, which will verify transaction level security. Additional security checks must be performed to ensure that an end user has proper access to view claims or membership data within a given

membership group, base group or agency (Refer to *Technical Standards — Applications > Application Coding > Online System Design Considerations > Router Navigation and Standards* for detailed information.)

Another example is the security system utilized by PGBA. For CMMS, the security standard is the CMMS Security System in conjunction with RACF authority.

The CMMS system uses module CM10P000 to route the transaction to the appropriate program. RACF authority controls the ability of the operator to route to the entered transaction, and the programs that are executed will check the level of CMMS security the operator must have to be able to perform the transaction function.

## Program Number (Name)

Online Software Developers must follow naming conventions set forth in *Technical Standards — Applications > Application Development Support Tools > Naming Conventions*.

To avoid duplicate program names, the Software Developer must also check Endevor inventory to verify the program name is not already in use.

Map (screen) names will be validated in the Design Review to avoid duplicate names.

## Date/Time

If the ACCEPT command is used in a CICS program, it must specify either DATE or TIME.

## Temporary Storage Usage

An eight character Temporary Storage Queue ID will contain the four character, Transaction Identifier followed by the four character Terminal Identifier.

## Screen Display Size

Since many of the Enterprise System application screens are scraped, screen size standardization is essential.

All future development must utilize the normal default of MOD 2, which is 24 rows X 80 columns display.

MOD 3, 32 X 80, will not be utilized.

## GETMAIN

The 'GETMAIN' CICS Command will not be used in application programs except where technically necessary.

## ENQUEUE

ENQUEUE will not be used unless required for technical reasons.

### EIB Block

The EIB block within CICS will be used as read-only. No updates will be performed to the EIB block within CICS.

### Menu Driven Map (Screen) Design

All online screens will follow the standards as defined in *Technical Standards — Applications* > *Application Coding* > *Online Screen Standards.*

### COMMAREA Usage

The CICS COMMAREA will be used to pass data from one program to another. When a program is accepting a COMMAREA from another program, the EIB length function must be checked to assure that the correct COMMAREA length has been given by the prior program.

## AMB Program Coding

The following standards will apply to all online AMB programs.

### Native CICS Coding in AMB Transaction

No native CICS coding is permitted in an AMB program. Any 'Exec CICS' instruction or "DFH" macro must be invoked by an AMB user rule that has been approved by ITBS Host.

### Function Key PA1, PA2, or PA3

All AMB online modules must check to determine if the user has depressed the function key PA1, PA2, or PA3 because map fields will be overlaid. This occurs because PA-keys return low values and AMB always reads the map before executing any user-written code. Therefore, Software Developers must check for this condition and rebuild their map fields if necessary. For existing AMB online modules, this must be corrected when maintenance is performed. There are two recommended techniques:

- Save your map off in TS queue every time you return with TRANSID (you can retrieve the TS queue map if it gets overlaid).
- Rebuild your key fields from COMMAREA.

## AMB Online Express

AMB ONLINE EXPRESS is a non-procedural, menu driven facility for defining, documenting, and generating complete online, COBOL-based programs.. Access to IMS DB, VSAM, DB2, and IDMS is supported. Programs may access single or multiple database systems. Applications that conform to a standard online transaction flow are quickly and easily developed.

ONLINE EXPRESS reads your application, screen and data view definitions from the AMB application and provides a series of fill-in-the-blanks program specifications windows. You define your program with predefined and individualized functions, customized processing, or other simple to complex specifications. ONLINE EXPRESS uses this information to write complete programs to the AMB program painter. You can immediately generate your program or further modify it using all AMB

programming facilities. It is required that all modifications to express programs be done thru ONLINE EXPRESS and never thru the program painter.

All custom code will be stored in the "Misc User Code" event found on the Action/Event Browser window. ONLINE EXPRESS allows for the insertion of custom code in many events. This allows for the scattering of code throughout EXPRESS..

All custom code will adhere to existing standards.

Since ONLINE EXPRESS does not provide a panel for describing what the program does, it is recommended that the standard remarks box be included in "Misc User Code."

The recommended format of the "Misc User Code" is:

```
FIRST -- REMARKS BOX
SECOND -- WORKING STORAGE FIELDS
LASTLY -- CUSTOM WRITTEN PARAGRAPHS
```

All custom written paragraphs will have a flower box describing their functions. This box will also contain the names of the express panels that perform this paragraph.

It is recommended that customer written code only reference custom written paragraphs and user defined fields. There will be exceptions to this. Every exception will be documented in the paragraph flower box:

- Indicate where the referenced data element originates
- Define the data element
- Indicate where the referenced paragraph originates and what functions it performs

In Express Generation Options, check the box for **Generate Control Point Comments** to enable comments for express-generated paragraphs.

## CICS Testing Environment

The CICS test regions have been structured in such a way as to provide two environments for validation. There is a Unit Test environment and a System Test environment in all Level of Effort (LOE) and development regions. The Unit Test environment should be used for the initial validation of program modifications and new programs, first by the Software Developer and then by a customer. Application validation and debugging tools such as Smart Test and CEDF are available in the Unit Test regions. However, to use these tools, the Software Developer must sign on directly onto the CICS region where the transaction will execute. The System Test regions should be used to perform system validation involving the customers as a final phase of validation prior to production implementation.

## Online Validation Procedures

### Program Review

A program review must be held prior to system validation of any new or modified module.

### CICS Table Updates

All CICS table updates (adds, changes or deletions) must be submitted to the "CICSADM" email account not later than 10:00 a. m. on Wednesday or Friday of the week prior to the start of testing. This allows Tech Support to add the entries to all tables involved.

Software Developers must use email to send a form to the "CICSADM" account to update all CICS tables. CICS tables normally updated are:

- Program Control Table (PCT).
  Form = CICSPCT.
- File Control Table (FCT).
  Form = CICSFCT.

### Operator Instructions

The Project Manager is responsible for seeing that the terminal operator instructions are written and delivered.

### Function Key Use

Refer to *Technical Standards — Applications > Online Function Keys*.

### Customer Validation and System Validation

The Project Manager has the responsibility of having the customer perform validation of the program and the transaction. The customer must validate to see that the program performs the functions specified and in the manner expected.

The customer validation will also be used for systems validation. This systems validation will apply when data entry is involved, and any other I/S area must process this data. The batch system is to run a full test cycle to validate the data that has been entered.

### Quality Assurance Review

The Project Manager must furnish the following information for transaction validation:

- Transaction flowchart
- Printed program listing
- Operator instructions
- Test data with results expected

Customer validation and a Design review must be held prior to the Quality Assurance review.

## Online Implementation Procedures

### CICS Table Updates

Technical Support has the responsibility for all CICS table updates. The CICS tables must have all Program Control Table (PCT) and File Control Table (FCT) entries prior to execution of any CICS

program. All updates to these tables must be sent to "CICSADM" no later than 10:00a.m. on Friday a week prior to program implementation.

Should a CICS table entry not be present at the time a program is moved to Production, the program will be removed from the production system and will not be moved to the production environment until the following week. There can be no exceptions to this standard without the written approval of the Senior I/S Operations Management.

### Move List

Programs to be implemented must be moved to Endevor Production prior to 5:30 a.m. on Friday of each week. If a production move is made after that, the special move procedures are to be followed.

### Customer Production Validation When Moved

The implementing Business System Analyst has the responsibility to set up a customer to perform validation in the morning after a program is moved to Production. Refer to *Technical Standards — Infrastructure > Operating Systems > Production Update Schedule* for more procedures.

## Online Maintenance

### Reviews and Documentation

The Software Developer will perform a review of any change with a senior member of the Applications Development staff. In many cases, this may be an informal review, but a senior staff member should initial the change/error sheet to show that the code meets the change requirements. Documentation within the program must show the reason for the change and the date of change. These comments should be in the body of the program for easy reference.

### Program Changes

The project code or change sheet number must document all changes to production modules.

When changes are made to a production module, the Software Developer should make use of comments in the program. These comments should show the reason for the change, the project code or change sheet number, and the approximate date of change.

### Validation Procedures (Program and System)

The Software Developer who has changed the code must perform Unit validation. After Unit validation, the customer will perform systems validation.

In systems validation, data will be passed to an adjudication area for completion of the System Validation where appropriate. Any changes that affect claims adjudication must be sent to the system involved for validation. No changed program will be moved to Production without a full systems validation.

# 4.1.3 Easytrieve Plus

## 4.1.3.1 Programming Guidelines

The purpose of this section is to provide an explanation of Easytrieve Plus coding conventions.

- All production Easytrieve Plus programs must be link edited and NOT executed as "compile and go."
- All production Easytrieve Plus programs must be under Endevor control.
- Adhoc Easytrieve Plus programs may be executed as "compile and go."
- EZ+/SQL may be run against DB2P and DB2T within the guidelines of DB2 programming standards (Refer to *Technical Standards — Applications > File Design > Enterprise Server – DB2*.). Authority will not be granted for EZ+ against production tables, only against informational and test. Table authority will be based on TSO ID or RACF group.
- EZ+/SQL must not conflict with CDMS image and refresh jobs.
- For EZ+/SQL, code reviews with the I/S area familiar with the tables being accessed is encouraged.
- All EZ+ logic statements must be coded within columns seven (7) through 72.

### Easytrieve Plus Coding

#### Environment Section

An Easytrieve Plus program name will be created as described for a supervisor program (Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Program Names*.).

The 'Snap' option of the 'ABEXIT' PARM must be used in Production and Test versions of a program. The following 'DEBUG' PARM options must be used in Production and Test programs:

- FLOW
- FLDCHK
- FLOWSIZ(20)
- STATE
- XREF(SHORT)

For EZ+/SQL the following should also be added:

- SQLID ('R____')
- SSID ('DB2P') (or 'DB2T')
- BIND DYNAMIC

#### Library Section

The name of a 'FILE' keyword will be created as defined in *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > COBOL DDNames*.

An Easytrieve Plus file definition copybook member will be named as defined in *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Copybook Names*.

All WORKING-STORAGE data elements should be coded in an area of a given Easytrieve Plus program under the heading WORKING STORAGE. This area should be after all input and output file definitions and before the activity section.

#### Activity Section

The 'GOTO' statement must only be used to 'GOTO JOB' or to 'GOTO' the exit of a performed procedure.

#### Report Section

For production EZ+, multiple reports will not be written from the same "FILE STATEMENT." Multiple reports will be written to separate DDs. Refer to *Technical Standards — Applications > Tools > Host Tools > Developer Tools* for the definition of report(s). Each LOB has its own report heading standards.

### Include/Copybook Names

#### Easytrieve Plus Includable Code Copybook Members

Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Copybook Names*.

## 4.1.3.2 Design Considerations

### VSAM Coding

Easytrieve Plus controls the "status" keyword that, while optional, must be used on all “VSAM” files when input by the Software Developer and not automatically. When specified, status of the file's I/O operations may be monitored by validating the file status key values; thus, ensuring correct execution before continuing processing.

## 4.1.3.3 Easytrieve Plus/SQL Coding

There Are No Special Considerations Other Than Normal Standards When Using SQL Within An Easytrieve Plus Program. However, It Should Be Stated That **Any Easytrieve Plus Program Should Update No Tables** Unless The Program Is A One-Time Run To Initially Propagate The Tables.

# 4.1.4 SAS

SAS is an application software product that provides access, management, analysis and presentation of an organization's most strategic asset, its data.

## 4.1.4.1 Host Access

On the enterprise platform, only batch components of SAS are currently used.

## 4.1.4.2 Workstation Access

SAS is installed on Windows and WIN/NT Workstations in both local client and server modes. It is also installed on various IBM AIX RS/6000 models running in server mode. SAS is accessed locally through BlueCross/GPC networks, intranets and the Internet. The current production SAS version is release 9.1.3 Service Pack 3. SAS is used interactively in several areas throughout BlueCross and the subsidiaries.

## 4.1.4.3 SAS - Batch

### Programming Guidelines

#### Batch/Online Coding

**Program Considerations**

At BlueCross and Palmetto GBA, SAS batch programs are not compiled prior to execution. Production SAS code is stored as source in PDS 'BC.NDVR.PROD1.SAS' and is fetched and included as part of the JOB set at runtime. There are no online SAS programs currently executing in BlueCross production systems on the Host. Any SAS online applications that currently reside on RS/6000 servers were developed and are maintained by the Medicare Statistical Analysis Department. Some SAS code executed in online applications requires compilation.

**Program Structure**

Below the library and file assignment levels, SAS coding is basically free form. There are no strict organizational requirements such as separate sections/divisions as found in COBOL and other common high level languages. This allows a very high degree of flexibility.

### Other Considerations or Guidelines

Batch Enterprise SAS only runs on the following LPARs:

- SYSF – BlueCross
- SYSG – Palmetto GBA

# 4.1.5 REXX/ISPF

## 4.1.5.1 Programming Guidelines

### REXX/ISPF Programming Considerations

#### Overview

REXX (Restructured Extended Execution Language) is a free-formatted, portable language, with syntax in English words. It is not an approved programming language for end-user processes but can be extremely helpful and a great productivity enhancement tool to automate many of your day-to-day tasks. REXX has advantages over other utility languages in the way it reads and writes records, uses the system's data stacks, provides controllable iterative processes and parses string data. REXX is an

interpreted language as executed from a PDS but has the capability of being compiled using the REXX Tools compiler. There are several built-in functions included in REXX, and REXX has the ability to run functions and processes external to the language. For example, a REXX routine can request and trap a LISTCAT request on a dataset or GDG in order to determine DCB, space, G0000V00 number, etc. Quick Reference contains a REXX manual to lookup syntax and verb usage. ISPF Dialogue Syntax and verb usage can also be looked-up in Quick Reference as well as in the online manuals under ISPF Development.

ISPF (Interactive System Productivity Facility) program product is an extension of the MVS Time Sharing Option (TSO) Host system on which it runs. ISPF services complement those of the Host system to provide interactive processing. A dialog receives requests and data from a user at a terminal. The dialog responds by using ISPF services to obtain information from, or enter information into, a computer system, REXX being the primary interface. All dialogues written will use the Common User Access (CUA). This allows your panels to act and react just like any other panel within your ISPF session. For example, PF3 will end the process; PF7 and 8 will control scrolling.

## Standards

As in any program, place documentation at the beginning of the routine to provide a brief overview of the code and other comment blocks just prior to each paragraph.

Comments should at a minimum include:

- A list of all DD-names used in the REXX and if appropriate, the datasets those DD names are to be referencing.
- A list of all PANEL names attached to the REXX, which it is invoking, or being invoked by.
- Code REXX in a structured style.
- Unless waived by management, do not code REXX programs for production processing cycles. BlueCross uses REXX as a utility language.
- Do not code LIBDEF or ALTLIB statements within your routines. BlueCross uses TRX to allocation/reallocate files to your ISFP session.
- Do not code an imbedded function more than four levels deep. This is strictly a maintainability issue.
- Enclose commands to external environments in double quotes. This is for readability.
- Enclose all literal strings in quotes. This assures strings are treated as strings.
- Although REXX is a free-flow programming language, do not code more than one statement per line. This is strictly a maintainability issue.
- Although REXX is a free-flow programming language, structure your code using an indented, block coding style. This is strictly a maintainability issue.
- Do not hide information in variables set far away from where the variable is referenced. This is an advantage to free form coding that assists in maintainability.
- End all internal procedures with an unconditional RETURN statement. This makes it easy to follow procedure calls.
- Use single letter variables (e.g., I, J, K, etc.) for iteration control within a loop structure. TSO/REXX has optimization for this type of DO constructs.
- Internalize external routines that are called repeatedly, as the REXX interpreter will load and tokenize each one every time it is called.
- When defining ISPPROF variables commonly used with VGETs and VPUTs, be aware of variable names already assigned within the current or executing ISPF Apl.
- New ISPF Apl should only be created upon a written request to IT Business Systems (ITBS) Host who will coordinate through Tech Support for approval. Be aware that everyone's ISPPROF

is a pre-defined size and will not expand past a pre-determined size; each new ISPF Apl will create at least one new member in the ISPPROF PDS dataset.

## 4.1.6 Notes Help Standards

The following Coding Standard is to prevent a "Storage Violation" that can bring down a CICS Region by program modules that interface with the NOTEs application.

One way to determine if a module is affected is by the presence of the COPYLIB member,"BLNTI100." Program modules that interface with the NOTEs application have the following coded in the Linkage Section.

```
      01  USER-HELP-BUFFER.

                  05 USER-HELP-LINE     OCCURS 100 TIMES

                       INDEXED BY HELP-LINE-INDX.

              10  USER-HELP-MSG.

                        15  USER_HELP-MSG1                PIC  X(30).

                              15  USER-HELP-MSG2                 PIC  X(35).

                              15  USER-HELP-SQL-DISPLAY    PIC  ------999.
```

USER-HELP-BUFFER cannot exceed 32767. The length of this area will be used in a GETMAIN instruction.

This coding standard requires all program modules that interface with the NOTEs application contain the additional code documented below.

Coding Specifications to meet this standard:

```
Change the timing for performing the GETMAIN paragraph

Make this the first paragraph done in the PROCEDURE DIVISION

Modify the GETMAIN paragraph to include the following three statements:

IF LENGTH OF USER-HELP-BUFFER > 32767
                        PERFORM 6000-LENGERR

COMPUTE BL-PASS-HELP-BUFFER  =
            BL-PASS-HELP-ROWS * BL-PASS-RECORD-LEN
```

```
IF BL-PASS-HELP-BUFFER > 32767
            PERFORM 6000-LENGERR
```

Place these statements before the "COMPUTE" statement for BL-PASS-BUFFER-LEN, but after the assignment of the various BL-PASS fields

```
Modify the GETMAIN LENGERR paragraph

Add the following statement:
          BL-RETURN-FLAG = 'A'

Delete/Comment out ALL assignments to USER-HELP-LINE (SUBSCRIPT)

MODIFY THE GETMAIN NOSTG PARAGRAPH

Add the following statement:
          BL-RETURN-FLAG = 'A'

Delete/Comment out ALL assignments to USER-HELP-LINE (SUBSCRIPT)

MODIFY THE VALUE OF &ASSIST-HELP-ROWS
If any program exceeds the 32K boundary, DECREASE the value being assigned to
the &ASSIST-HELP-ROWS

If necessary, add/change the symbolic at the top of the program to a value that
will reduce the size of USER-HELP-BUFFER and the resulting GETMAIN storage
request
```

# 4.2 Naming Conventions

The purpose of this section is to describe the various naming standards in effect for systems and system components.

For the naming conventions for IMS/SB Database Definitions (DBD) and IMS/DB Program Control Blocks (PCB) and IMS/DB Program Specification Blocks (PSB) and IMS/DB Segments refer to *Technical Standards — Applications > File Design > Enterprise Server – IMS > Design Considerations.*

## 4.2.1 Job Names

### 4.2.1.1 Job Names

This section does not apply to Easytrieve jobs.

A job name will be eight characters long and have the following breakdown (Figure 4-1):

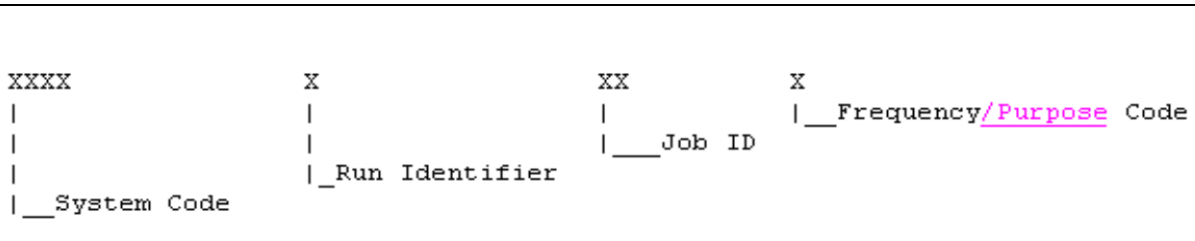



**Figure 4-1** Job Names

#### System Code

Refer to *LOB Management > Financial Management > Budgeting & Cost Control > Accounting and System Codes* for valid system codes.

#### Run Identifier

Run Identifier

J - All

#### Job ID

The Job ID will either be a two-digit number, which is unique within the system code, or alphabetic characters with meaning. Example: Strip jobs - CHAPJS1D.

# Frequency/Purpose Code

**Frequency/Purpose Codes**

| Code | Frequency or Purpose |
|---|---|
| A | Annually |
| B | Biweekly/Bimonthly |
| C | Compiles and assemblies |
| D | Daily |
| L | Library maintenance |
| M | Monthly |
| N | I/S job which does not create production files, but requires more resources than usual test (T) jobs. |
| P | I/S pseudo-production job, which may create or modify production files |
| Q | Quarterly |
| S | Semiannually |
| T | Test |
| U | As requested |
| W | Weekly |
| X | Documenter |
| $ | Test job through ZEKET. Last character of the jobname must be $ or #. Medicare Systems may use these characters as well as an @ sign in the third or fourth position of the jobname. |
| # | Test job through ZEKET. Last character of the jobname must be $ or #. Medicare Systems may use these characters as well as an @ sign in the third or fourth position of the jobname. |

**Table 4-1** Frequency/Purpose Codes

During validation, the Job ID will be assigned by the Portfolio Manager responsible for the system. Upon system turnover, operations may elect to change the Job ID. Job names, particularly prefixes, cannot be changed without approval from an I/S AVP. If changed, upper management must be notified.

## 4.2.1.2 Easytrieve Plus Job Names

An Easytrieve job name will look as follows:

EASYJXXT

Where XX is any two characters to help identify your job.

For valid account numbers to be used on Easytrieve job cards, ask your manager. If in doubt on assigning a proper account number, call the Information Center.

If any Easytrieve is incorporated into production systems or part of the validation of a system, the job prefix for that system can be used with the approval of the I/S manager responsible for that system.

## 4.2.1.3 Connect:Direct Job Coding Standards

Connect:Direct (C:D) is a file transfer product from IBM. C:D was formerly called Network Data Mover (NDM), and this name is still widely used. This section provides guidelines and standards for coding jobs that use C:D.

### Connect:Direct System Datasets

C:D system dataset names may change during product upgrades, or for other business reasons. C:D DMBATCH procs, which include these system datasets, are located in SYS2.PROCLIB. If the system dataset names are changed, these procs will be updated concurrently. Jobs which invoke C:D must use one of these procs rather than hardcoding the system datasets within the job. This will minimize the risk of job failure due to changes to the system C:D datasets.

In the following STEP example,

```
//STEPS010  EXEC NDMxxxxB
```

refers to the appropriate C:D DMBATCH proc. NDMxxxxB is the C:D DMBATCH proc for the C:D region to be used.

The following C:D batch procs are available in SYS2.PROCLIB (Table 4-2):

**C:D Batch Procs Available in SYS2.PROCLIB**

| Proc Name | System — Region | C:D Node Name | Line of Business |
|---|---|---|---|
| NDMBCBSB | SYSA – NDMBCBS | SCA.A70NDM.BLUE | Primary C:D for BlueCross BlueShield of South Carolina |
| NDMCHAMB | SYSA – NDMCHAM | SCA.A70NDM.CHAM | C:D for CHAMPUS |
| NDMPGBAB | SYSA – NDMPGBA | SCA.A70NDM.PGBA | C:D for PGBA |

| | | | |
|---|---|---|---|
| NDMDEERB | SYSA – NDMDEER | SCA.A70NDM.DEER | C:D for DEERS |
| NDMHCFAB | SYSG – NDMHCFA | A70NDM.BXSC | C:D for HCFA |
| NDMMEDCB | SYSG – NDMMEDC | SCA.A70NDM.MC | C:D for Med |
| NDMTBB | SYSG – NDMTB | SCA.A70NDM.TB | C:D for TrailBlazer |
| NDMEDCAB | EDCA – NDMEDCA | CD3.EDC1 | C:D for Fee for Service |
| NDMEDC4B | EDCA – NDMEDC4 | CD3.TSK4 | C:D for Task 4 |
| NDMEDCBB | EDCB – NDMEDCB | CD3.EDC2 | C:D for Fee for Service |
| NDMNLRB | EDCH – NDMNLR | CD3.EDC5 | C:D for NLR |

**Table 4-2** C:D Batch Procs Available in SYS2.PROCLIB

## Connect:Direct Process Libraries

You must include a DMPUBLIB DD statement in your job to point to the dataset that contains the NDM process(es) you will run. This statement should point to a dataset which is owned by the appropriate application area, and protected by RACF. The dataset pointed to by the default DMPUBLIB statement in the DMBATCH proc cannot contain application-owned C:D processes.

## C:D Notify Jobs

When you run a job to submit a C:D process, completion of the job only reports whether the process was submitted, not the result of the process. The example below provides a method for submitting C:D jobs that will provide notification of transmission success or failure.

To use this job, replace the $NDMVAR1xxxxxxxx as indicated. If necessary, this job can be used as a trigger for any needed successor.

```
//JobNameX JOB (ACCOUNT CODE,,,PROD),'ndm verification job',

// CLASS=8,

// USER=XXXXXX,

// MSGLEVEL=(1,1),MSGCLASS=0

//JOBLIB DD DSN=BC.NDVR.FIXP1.EXECLIB,DISP=SHR

// DD DSN=BC.NDVR.PROD1.EXECLIB,DISP=SHR

//*$NDMVAR2JobNamex

//STEPS010 EXEC PGM=$NDMVAR1JobNamex
```

The job will be assigned an operator okay of "yes" on the event master record once the job has been added to the scheduling database. The event will be dropped at the next schedule load if not run. If the

C:D transmission runs correctly, then the JobNameX will execute an IEFBR14. If the transmission fails, JobNameX will ABEND.

## 4.2.2 Job Step Names

If used, a job step name will have the following breakdown (Figure 4-2):

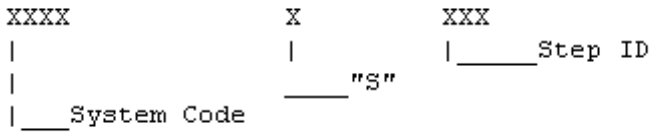



**Figure 4-2** Job Step Names

The Step ID will be a three-digit number in ascending sequence, initially starting with 010 and incrementing by 10. Subsequent updates that insert steps may use intermediate Step ID numbers that fit between the previously existing steps.

## 4.2.3 Cataloged Procedures Names

A cataloged procedure name will have the following breakdown (Figure 4-3):

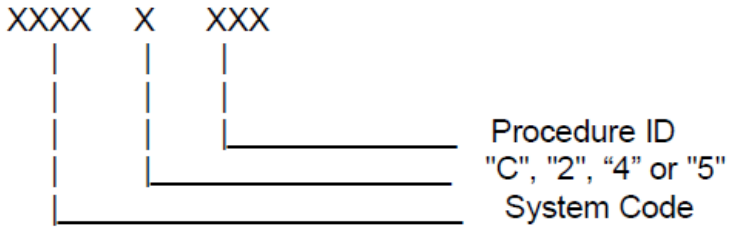



**Figure 4-3** Cataloged Procedures Names

TRICARE (CHAMPUS) PROCS will also use the following format:

The Procedure ID will be an alphanumeric field with the following breakdown (Figure 4-4):

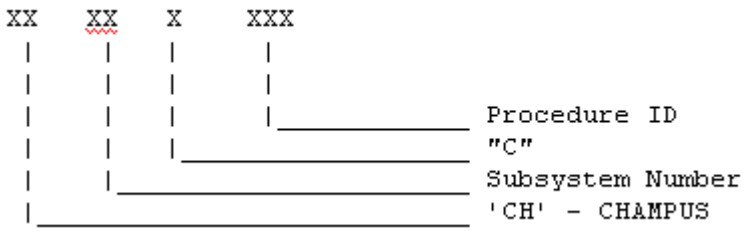



**Figure 4-4** Procedure ID

1. In the case where a PROC will be used in only one job and is the only PROC used in that job, this field will contain the last three characters of the job name.
2. In the case where a PROC will be used in more than one job, a Project Manager will assign this field.
3. In the case where a PROC will be used in only one job but is not the only PROC used in that job, the Project Manager will assign this field.
4. In the case where a PROC originally was used in only one job and was the only PROC used in that job, but later met conditions specified in Item 2 or 3 above, all references to the PROC name must be changed to meet the standards outlined in Item 2 or 3 above.

## 4.2.4 Program Names

A program name will have the following format (Figure 4-5):

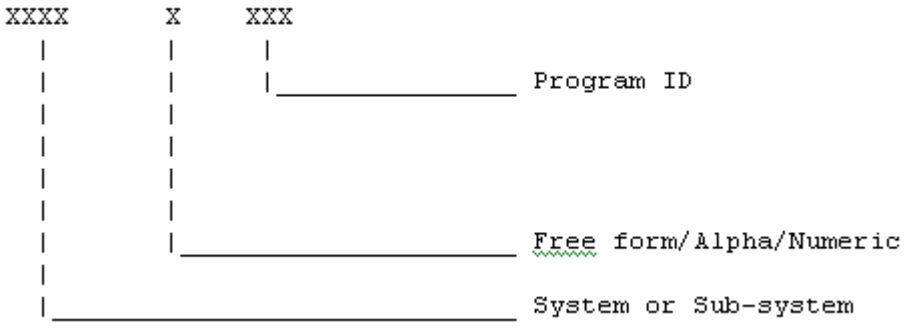



**Figure 4-5** Program Names

The system or subsystem ID is entered into the first four positions.

The fifth position of the program name is a free form field used to indicate the type of program. Assignment of the field value is determined by each BlueCross system. Prior to modification of this standard, the following values were in effect and may be in use in existing program names:

"D"- Database Supervisor Program

"E"- Entry Point

"L"- Called Module

"P"- Non-Database Supervisor Program

"V"- VSAM I/O Module

“Z”- MQ Parameter Module

---

**NOTE** An Entry Point will be the same as the Program ID for the module in which the entry point appears. If more than one entry point is used in the module, each succeeding Entry Point ID will be incremented up by 001. Entry points are not a recommended coding technique.

---

The Program ID will be a three character alphanumeric identifier that is unique within system code. The preferred standard is to provide a three (3)-byte numeric field.

To avoid duplicate program names, the programmer must check the Endevor inventory to verify the program is not already in use.

# 4.2.5 COBOL DDNames

## 4.2.5.1 Non-IMS DDNames

The external-name in the COBOL assign clause and the DDName in the DD card of the JCL for that program will be the same and will have the following breakdown for Non-IMS/DB applications (Figure 4-6):

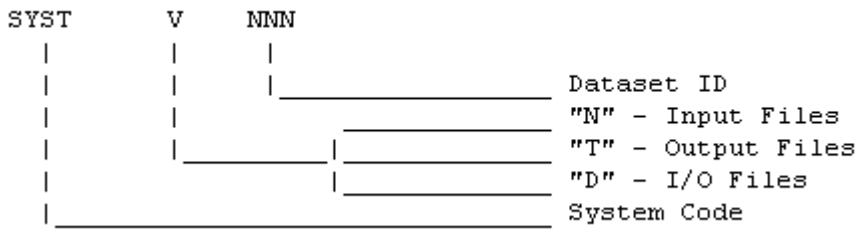



**Figure 4-6** Non-IMS DDNames

The Dataset ID should start at 010 and be incremented by 10. Package systems developed and written outside of this organization are exempt from the above standard.

Refer to *LOB Management* > *Financial Management* > *Budgeting & Cost Control* > *Accounting and System Codes* for valid system codes

### 4.2.5.2 IMS DDNames

The external-name in the COBOL assign clause and the DDName in the DD card of the JCL for that program will be the same and will have the following breakdown for IMS/DB applications (Figure 4-7). The database administration area will assign the names.

- Maximum length is eight characters
- Format

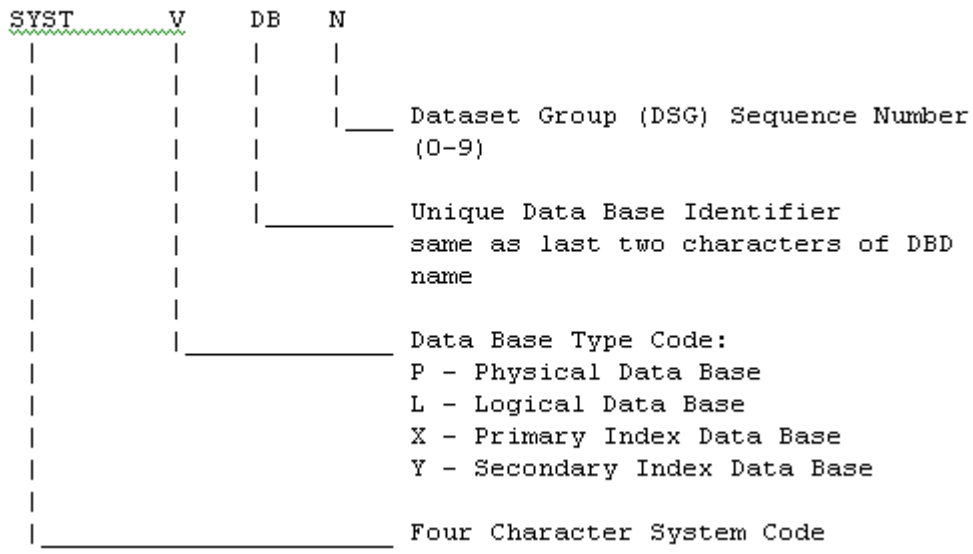



**Figure 4-7** IMS DDNames

## 4.2.6 Copybook Names

The purpose of this section is to provide an explanation of the naming conventions used for copybook PDS members.

### 4.2.6.1 Copybook Members

**AMB/COBOL** and **Easytrieve** copybook members will have the following naming format (Figure 4-8):

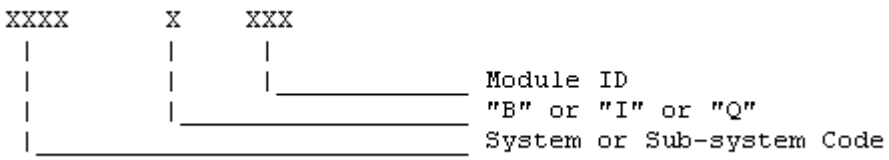



**Figure 4-8** Copybook Members

1. The System or subsystem ID is entered into the first four positions.
2. The fifth position indicates the type of copybook. The valid types for new development are:

   a. "B" - Non-Easytrieve Plus File and Working Storage Definition Copybooks
   b. "I" - AMB/COBOL and Easytrieve Plus Procedural code
   c. "Q" - Easytrieve Plus File and Working Storage Definition Copybooks

3. The Module ID will consist of three alphanumeric characters that are unique within the system. The project manager responsible for the system will assign the unique Module ID.

---

**NOTE** IMS copied COBOL and Easytrieve source members are referred to as "segments" and are named based on standards defined in *Technical Standards — Applications > File Design > Enterprise Server – IMS*. Further, IMS segment contents are generated from the Data Dictionary by the DBAs and are not manually created by other Software Developers.

---

Refer to the section *AMB Naming Conventions* below for other elements unique to AMB.

## 4.2.7 General System Library Names

A Systems Library will have the following breakdown (Figure 4-9):

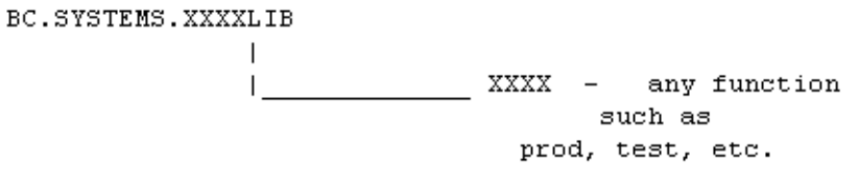



**Figure 4-9** General System Library Names

Only Technical Support personnel may assign a Systems Library name.

Under Endevor/MVS, a Systems Library will have the following breakdown:

```
BC.SYSTEMS.ENVXXXX.YYYYYYY

XXXX =          UNIT, SYST, PROD

YYYYYYY =       LOADLIB, EXECLIB, DOCULIB, VSAMLIB,

                LINKDECK, COPYLIB, CARDLIB, PROCLIB,

                LIBRLIB.
```

# 4.2.8 CARDLIB Member Names

CARDLIB Member Names

All members in the production CARDLIB PDS will be defined in the following manner (Figure 4-10).

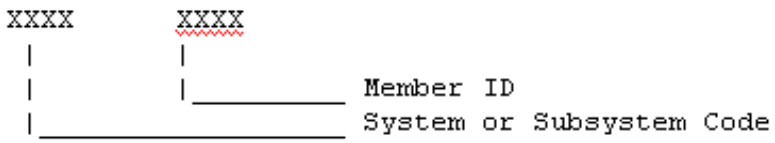



**Figure 4-10** CARDLIB Member Names

Refer to *LOB Management* > *Financial Management* > *Budgeting & Cost Control* > *Accounting and System Codes* for valid system codes.

# 4.2.9 File Names

## 4.2.9.1 Non-Temporary Dataset Names

General JCL rules for coding Non-Temporary Qualified DSNames

All production and non-temporary test datasets, created or retrieved, must be cataloged and thus named according to the JCL standards for coding non-temporary qualified datasets.

Standard rules to be followed are:

1. A Non-Temporary Qualified Dataset Name consists of one through 44 characters (including periods), except when the qualified name identifies a generation data group. In this case, the

dataset name may consist of only one through 35 characters (including periods), since the last nine characters are reserved by OS for the relative generation number (.G9999V99) pointed to by the index coded in the DSN parameter (+ or - n).

2. Each level of qualification, subsequent to the first level, must be preceded by a period.
3. Each level of qualification may consist of one through eight characters, not including a preceding period.
4. The first character of each level of qualification must be an alphabetic or national (, #, c) character.
5. The remaining characters of each level of qualification can be any combination of alphanumeric or national characters, a hyphen, or plus zero (12-0 punch).
6. GSAM files require pre-allocation. To ensure that this space is not released prematurely, the fourth or fifth node in the dataset name should be "GSAM." System managed storage will honor this "GSAM" qualifier and not release the excess DASD. This will be honored for both test and production dataset names.

For further details, see MVS JCL Reference Manual.

## 4.2.9.2 Special In-House Rules for Coding Proc DSnames

In addition to the general JCL rules for coding Non-Temporary Qualified Dataset Names, the following special in-house rules must be observed for identifying all non-temporary I/O datasets defined either in PROCLIB cataloged procedures or in in-stream procedures.

## 4.2.9.3 Production Generation and Non-Generation Datasets

### Non-IMS/DB Datasets and NON-TRICARE Datasets

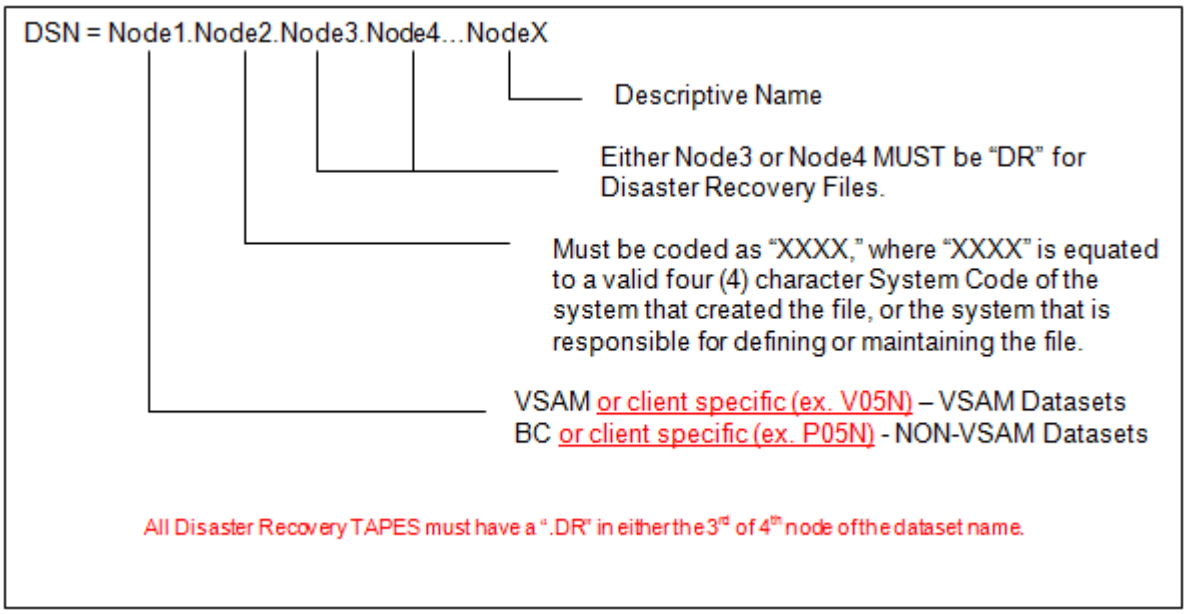



**Figure 4-11** Non-IMS/DB DATASETS and NON-TRICARE DATASETS

Symbolic Generation Parameters should be used to make job restarts easier and are applicable to the following standards; Figure 4-11 and Figure 4-13 through Figure 4-14. Symbolic Generation Parameters should be coded as follows (Figure 4-12):

```
IGN05='-1'
PGN20='+1'
|   __
| |
| |_________________ Step where Gen is
|                    used or created.
|
|_______________ IGN = Input Generation
                 PGN = Output Generation
```

**Figure 4-12** Symbolic Generation Parameters

The advantage to coding step numbers in the Generation Parameter Symbolic is that is easy to override generations already created with a '+0' override for jobs that must be restarted in a certain step. For example, if a job goes down in step S050, all created generations in any previous one can be easily identified and overridden by simply examining the Generation Symbolic Parameter name. In our example above, all references to PGN20 could be overridden with one simple statement: PGN20='+0'

## TRICARE Datasets

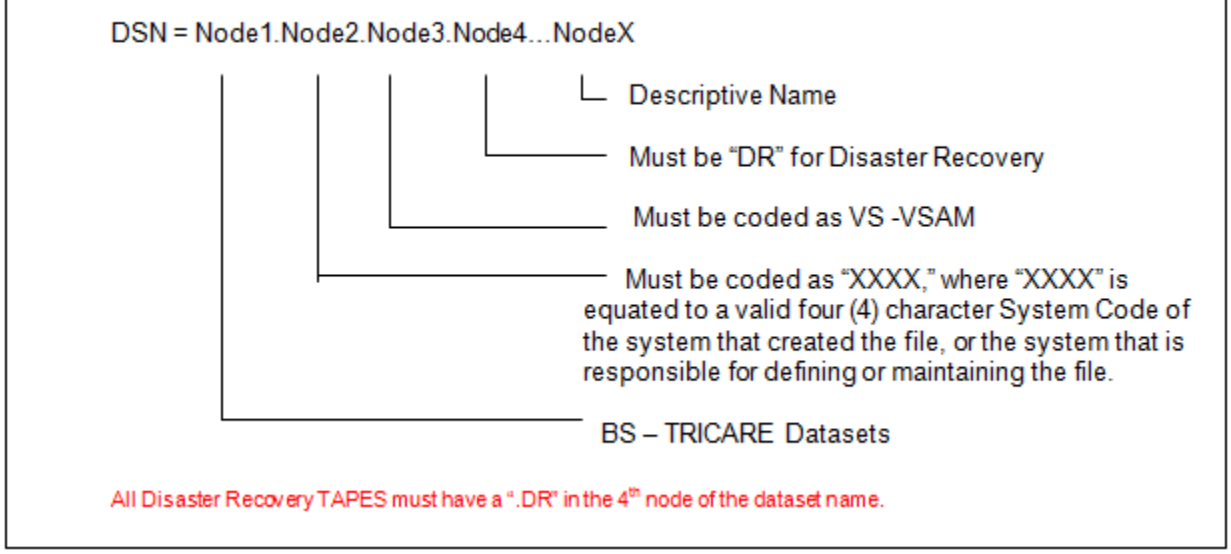



**Figure 4-13** TRICARE Datasets

## IMS/DB AND DB2 Datasets

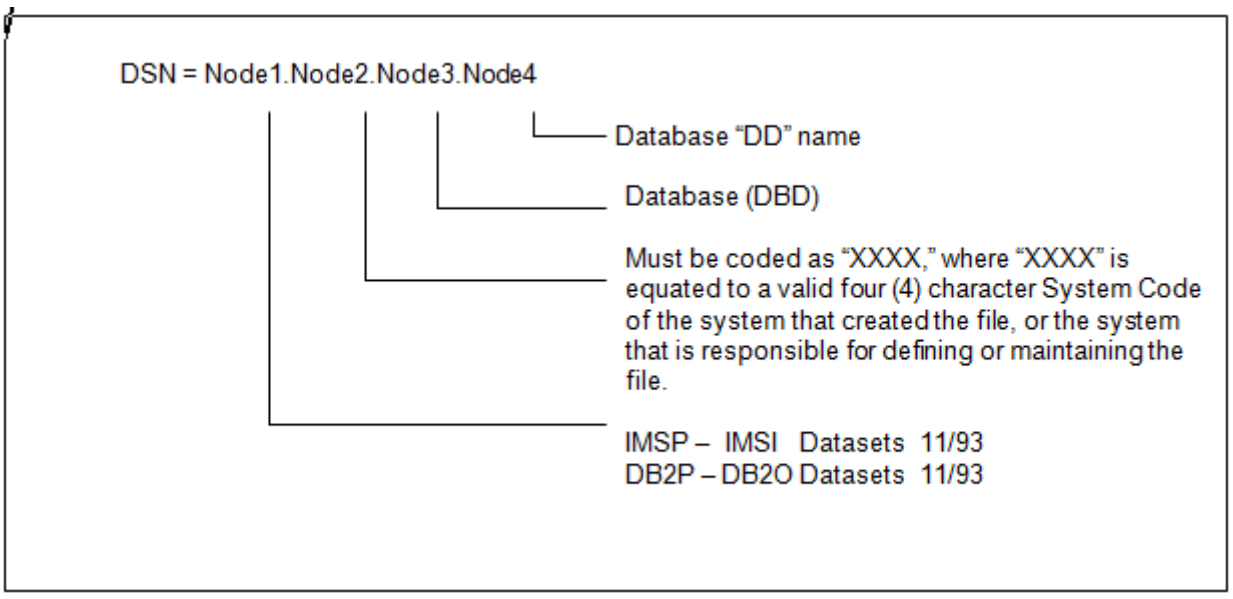



**Figure 4-14** IMS/DB and DB2 Datasets

* The system code is a unique set of four (4) characters established and maintained by Technical Support to be the accounting prefix for all datasets related to a particular application. (Refer to *LOB Management > Financial Management > Budgeting & Cost Control > Accounting and System Codes* for a listing of valid system codes.)

If no system code is used, or if an invalid system code is used, OS will automatically reject the Dataset name via a JCL error. Further, should a Non-Temporary Dataset nonetheless be generated in a test mode, Technical Support will automatically scratch it at the earliest opportunity.

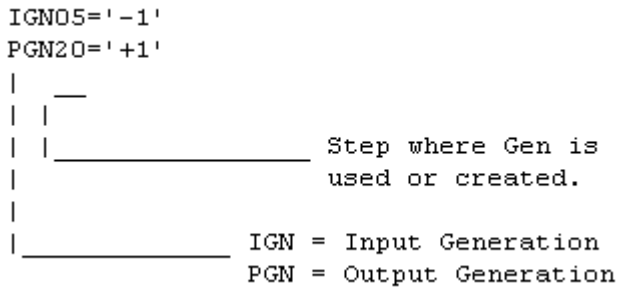



**Figure 4-15** Symbolic Generation Parameters

The DMS/OS DASD Management System will be the exception to this dataset naming convention, as the automatic Archive/Restore feature requires DMS/OS to interface with TMS and control all of its Archive

Datasets via Archive Index Maintenance. This is a feature of the software product that cannot be modified to shop standards. These datasets will follow this naming convention.

```
DSN =    DMS.ARCHPRIM.DMSN.CXXXXYYY.TZZZZZZ

DSN =    DMS.ARCHCOPY.DMSN.CXXXXYYY.TZZZZZZ

Where:   XXXX      is the creation year

         YYY       is the creation Julian day

         ZZZZZZ    is the creation time, hhmmss
```

## Test Generation and Non-Generation Datasets

### Batch

The following naming conventions must be observed when naming a batch, non-IMS/DB dataset

```
DSN= Node1.Node2.Node3...Nodex
```

Where:

```
Node1 =                    'TESTVS' for VSAM datasets

                           'TEST' for non-VSAM datasets

                           'BS' for TRICARE datasets only

Node2 =                    Must be a valid four (4)-
                           character system prefix (e.g.,
                           AMMS,CHAP,etc.)

                           For client specific datasets
                           this node may be 2 nodes (e.g.
                           AMMS.A2)
```

```
Node3 =                    Must be coded according to the

                           following breakdown:

             Non-TRICARE:  "TGnnnn" where:

             T =           "T" Constant for test datasets

             G =           Management group prefix

             Nnnn=         Employee ID

Nodex                      Other descriptive names to
                           help identify the dataset
```

## Online

The following naming conventions must be observed when naming an online, VSAM dataset:

```
DSN =     Node1.Node2.Node3...Nodex
```

Where:

```
Node1 =     'TESTVS' for VSAM datasets

Node2 =     Must be valid four (4)-character system
            prefix

            (e.g., AMMS,CHAP,etc.) For Client
            specific datasets this may really be 2
            nodes (e.g. AMMS.A2).

Node3 =     Must be the name of the CICS test AOR
            associated with the file

Nodex =     Other descriptive names to help identify
            the dataset
```

## Other TRICARE Datasets

Some TRICARE Test Datasets will follow the same test dataset format as above 11/93. Others follow the format below (Figure 4-16).

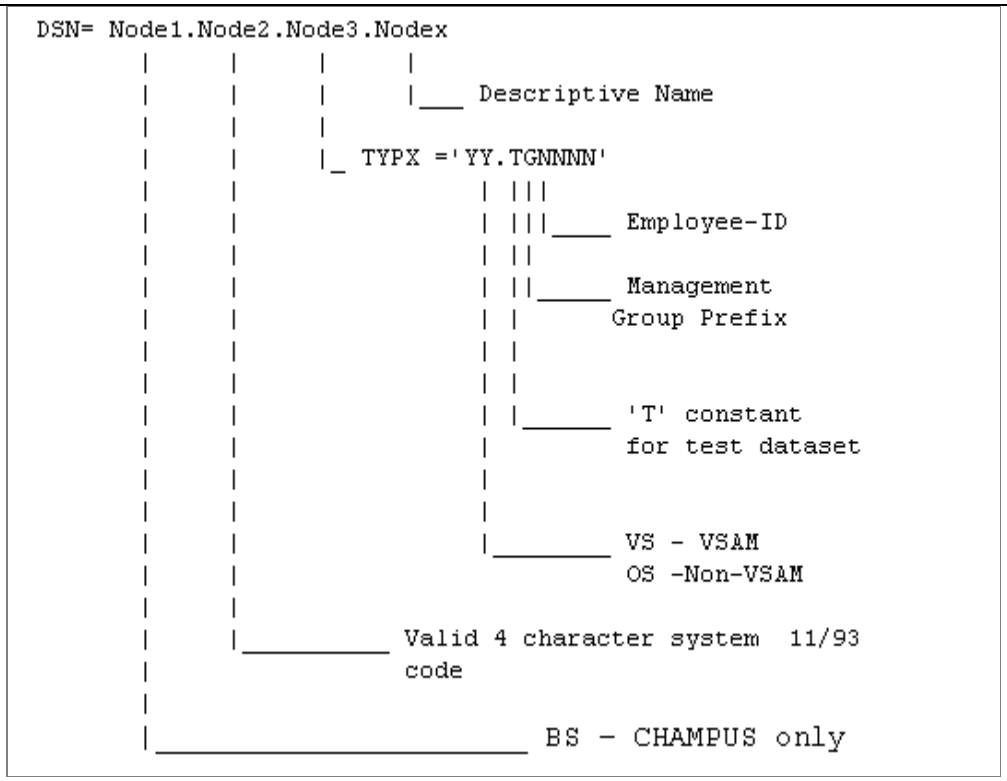



**Figure 4-16** Other TRICARE Datasets

## IMS/DB Test Datasets

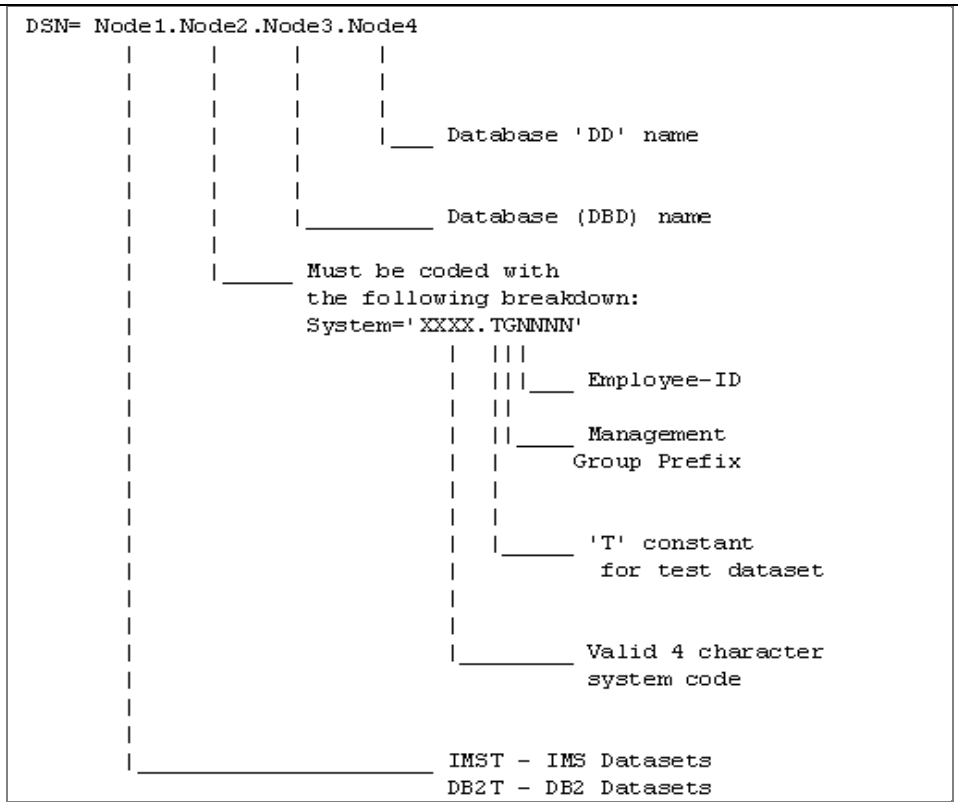



**Figure 4-17** IMS/DB Test Datasets

## Special In-House Rules for Coding Non-Proc

### DSNames

All Non-Temporary Non-PROCS Datasets must also be qualified and thus named in the same way as Non-Temporary PROC I/O datasets.

Datasets not defined in PROCS should be defined in the same format as those in PROCS. The only difference is the elimination of the Symbolic Parameter.

## Special In-House Rules for Coding External

### DSNames

In cases where external datasets are being used as input, the file should be copied over to a file with a dataset name that conforms to the above standards.

In cases where files are created for external agencies, the file name should conform to the specifications of those agencies. If no specifications are given, the in-house standards for naming datasets sed.

### 4.2.9.4 DOCULIB Member Names

The purpose of this section is to explain the assignment of DOCULIB member names.

Any member of the production DOCULIB will be named the same as the production job for which it contains operational documentation. Refer to the section *Job Names* above for further information on how to construct a job name.

## 4.2.10 Load Module Names

A Load Module will be named according to the following conventions:

1. If it is a "Called" Module, it should have the same name as the source code member that was compiled to create it.
2. If it is an "Executable" Module, it should have the same name as the source code member that was compiled to create it.
3. "Composite" Modules should have the same name as the source code member that was compiled to create the "Supervisor" module of the composite module.

**NOTE** For an explanation of the naming conventions for source code members, refer to the section *Program Names* above.

## 4.2.11 AMB Naming Conventions

The purpose of this section is to provide an explanation of naming conventions to be applied to the various entities associated with AMB.

### 4.2.11.1 General Discussion

Programs and maps (screens) coded in AMB will use standard naming conventions. Refer to the section *Program Names* above.

### 4.2.11.2 Application Name

An AMB application name will have the following breakdown (Figure 4-18):

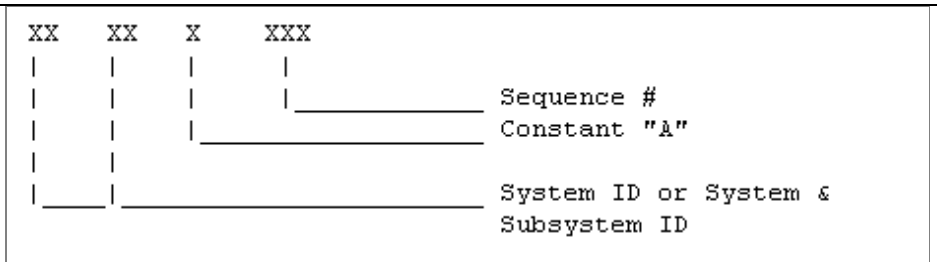



**Figure 4-18** AMB Application Name

## 4.2.11.3 Data Structure Name

An AMB Data Structure Name will have the same breakdown as a copybook member. Refer to the section *Copybook Names* above.

## 4.2.11.4 AMB Stub Name

An AMB Stub Name will have the following breakdown (Figure 4-19):

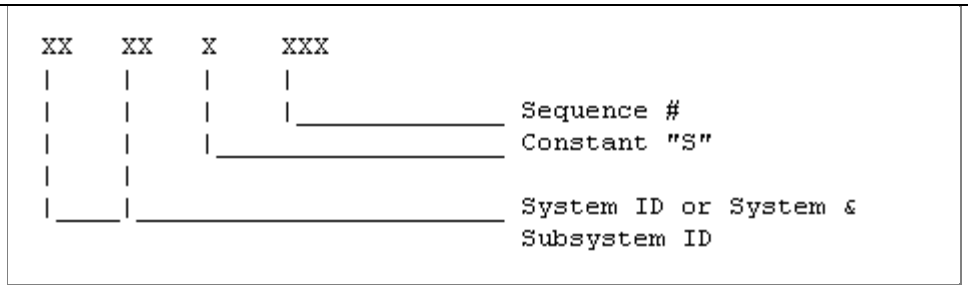



**Figure 4-19** AMB Stub Name

## 4.2.11.5 Report Mockup Name

An AMB Report Mockup Name will have the following breakdown (Figure 4-20):

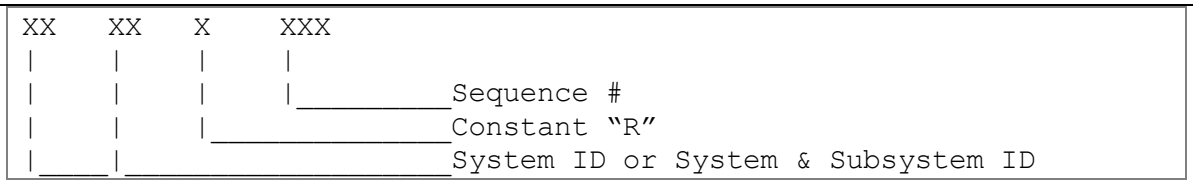



**Figure 4-20** AMB Report Mockup Name

# 4.2.11.6 Regional Processing Number (RPN)

Regional processing numbers (RPNs) are used throughout the corporation to delineate blocks of business. RPNs should have no intelligence built into the number, and each RPN should identify the same block of business throughout the corporation.

An Enterprise Architect should approve new RPNs during the Design Phase of a project.

Existing RPNs and a descriptive title will be documented in the DESCRIPTION on the Corporate Data Dictionary under the generic (no system ID) DD- 21183. As new RPNs are assigned, the project team requesting the new RPN must update this element description.

Additionally, the DB2 RPN-to-LOB cross-reference table (PRODCDB.RP_MAST_LOB_XREF) should be maintained to relate the RPN to the Corporate line of business.

# 4.2.11.7 Online Map Names

## General

The following rules are in effect for naming Map Sets, Maps and AMB Screens for online applications.

- Map Set, Map and AMB Screen names must be seven (7) characters in length.
- Map Set, Map and AMB Screen must be unique for the online systems to process.

To avoid duplicate Map and Map Set names, the Software Developer must check the Endevor inventory to verify the name is not already in use as a program, Map, or Map Set.

## Map Set Names

A Map Set containing a single Map must have the same name as the Map.

Map Sets containing multiple Maps must have the same name as the first Map within the set, with a dollar sign **($)** in position five (5).

For example:

```
MAP name – PIMSM03

MAP SET name (single map)      – PIMSM03

MAP SET name (multiple maps) – PIMS$03
```

## Map and AMB Screen Names

It is recommended that all Map and AMB Screen names conform to the following format:

| | |
|---|---|
| POS. 1-4 = | System code (Preferred) |

| | |
|---|---|
| POS. 5 = | 'M' designating map |
| POS. 6-7 = | Open - Where possible, a 2-character field matching the associated program name.. For example:<br>Program name – PIMSP030<br><br>Map name – PIMSM03_ |

Maps and AMB Screen Names must not contain a dollar sign ($) in the fifth position.

Maps created by AMB Screens will always have the same name as the screen.

# 4.2.12 Print Operations Standards

## 4.2.12.1 Advanced Function Presentation Resource Change Control Standard

Any new or existing Advanced Function Presentation (AFP) resource requiring creation or change, whether it is a PageDef, FormDef, Overlay, or PSEG, must be requested using the **AFP Resource Request** via the TSC Self-Service. This form must be filled out in its entirety and clearly state all required information. If the form is incomplete, the request will be rejected, and a new form will be requested with the missing required fields completed. This will result in a delay in the change control process. Upon completion of resources, refer to *Technical Standards — Applications > Print Operations > ICT Print Operations and Corporate Mail Services > InfoPrint Workflow Validation Procedure* for validation procedures.

## 4.2.12.2 StreamWeaver and Automated Document Generation System Integration

StreamWeaver is a post composition tool that takes a precomposed data stream and applies post composition elements. The post composition elements include — but are not limited to — insertion barcodes, page sequence numbers, and a duplicate copy watermark for CMOD.

A Unique Identifier (&DOCUID) must be used for all output that is run through the Automated Document Generation System (ADGS) process. This Unique Identifier must start with the same first four characters as a valid Job Prefix that is associated with the output.

All documents received from the ADGS/DOC1 process must contain the No Operational (NOOP) record for balancing and all applicable Tagged Logical Elements (TLEs). The NOOP record contains the total documents and logical pages within the DOC1 Advanced Function Presentation Data Stream (AFPDS). StreamWeaver determines what a document is based on by the system-generated Generation Unique ID (GUID) that is passed in the file from the ADGS process.

The TLEs include those shown in the tables below (Table 4-3, Table 4-4, and Table 4-5).

### IPW TLEs

| Mandatory/ If Applicable | TLE Name | Description | Notes |
|---|---|---|---|
| Mandatory | ADFDATE | Date | Format MM/DD/YY |
| Mandatory | ADFACCT | Account Code | Min. four characters/Max. 16 |
| Mandatory | ADFADDR1 | Address line 1 | Normal address line |
| Mandatory | ADFADDR2 | Address line 2 | Normal address line |
| Mandatory | ADFADDR3 | Address line 3 | Normal address line |
| Mandatory | ADFADDR4 | Address line 4 | Spaces if not used |
| Mandatory | ADFADDR5 | Address line 5 | Spaces if not used |
| If Applicable | ADF_MINOR_LOB | Route Code | Required if multiple LOBs |
| If Applicable | ADFCKNO | Check Number | Required for checks |

**Table 4-3** IPW TLEs

### Print TLEs

| Mandatory/ If Applicable | TLE Name | Description | Notes |
|---|---|---|---|
| Mandatory | PRTTYPE | Type of Doc | Letter, EOB, Remits, etc. |
| Mandatory | PRTPMODE | Print Mode | Simplex/Duplex |
| If Applicable | PRTBUSEC | Bus Sector | Group within LOB |
| If Applicable | PRTXREFID | Cross Ref ID | Subs ID/Card ID |
| If Applicable | PRTSIGNAME | Signature | Signature Code Name |
| Mandatory | PRTDUP | Y/N | Duplicate Copy Watermark |
| Mandatory | PRTIND | Y/N | Print Suppression Indicator |
| If Applicable | PRTSTACK | EOB Stack Ind | Settlement Only |
| If Applicable | PRTZIP | IF sort by ZIP | Zip only field, one per envelope |

**Table 4-4** Print TLEs

**CMOD TLEs**

| Mandatory/ If Applicable | TLE Name | Description | Notes |
|---|---|---|---|
| Mandatory | REPORT_ID | CMOD Report ID | Name of the Report ID |
| If Applicable | VERS_NO | CMOD Version | Version of the CMOD Report ID |
| If Applicable | POST_DATE | CMOD Posting Date | Posting date of document |

**Table 4-5** CMOD TLEs

## 4.2.12.3 StreamWeaver JES Monitor

The JES Monitor is a started task that monitors the JES queue destination for PRT9900 and PRT9901 in Production and STRW1 in Test. All jobs received at these destinations are checked against the StreamWeaver Job Table to determine if the secondary StreamWeaver job should be executed.

This StreamWeaver Job Table contains the following fields that are specified in the I/S application area's job:

- JOBNAME
- SYSOUT
- FORMS

ICT Print Operations will need to be notified with what the Production, rerun and Test job names will be for all output using StreamWeaver. ICT Print Operations will then update the StreamWeaver Job Table with the required information.

## 4.2.12.4 Production Reruns & Alternate Job Naming Standards

The Alternate Job Name convention for InfoPrint Workflow (IPW) is a process by which print jobs can be run to match up with the Service Level Agreements (SLAs) that are maintained in IPW. It is also the means by which jobs will be validated using existing Production configurations on IPW. Alternate job names must be included on Production reruns and test jobs intending to mimic the behavior of Production jobs. Print Jobs that are submitted without the proper naming conventions or wrong SYSOUT class will not print or will print improperly. IPW will only check for the alternate job name parameter in JOB NAMES ending in "P."

When the print that is being produced is intended to be a Production print, in z/OS, the alternate job name is to be included in the first eight bytes of the programmer name field on the rerun job card. This means that it is to be handled with the same timely standards that would be applied to a Production print.

An example of this would be Production job- FS00J40D. In order to rerun any of the print output, the alternate job name of FS00J40D needs to be placed in the FS00J40P rerun job. The rerun JOB must end in "P." The alternate job name is to be placed in rerun JCL as the example below illustrates.

```
//FS00J40P  JOB  (11806,326,00000,PROD),'FS00J40D',
//       USER=FS00APP,
//   MSGLEVEL=(1,1),REGION=4000K,CLASS=7,MSGCLASS=0
/*JOBPARM K=90
//*
//JOBLIBDD DSN=BC.NDVR.PROD1.EXECLIB,DISP=SHR
//*
//FS00S480 EXEC PGM=IEBGENER
//OUT1 OUTPUT FORMDEF=1727,PRMODE=DUPL
//SYSUT1       DD DSN=BC.FS00.COMBINED.EOB.G3763V00,DISP=SHR
//SYSUT2       DD SYSOUT=6,FCB=T580,OUTPUT=*.OUT1,RECFM=FBA,
//                            LRECL=111,OPTCD=J
//SYSIN          DD  DUMMY
//SYSOUT      DD SYSOUT=C
//SYSUDUMP DD SYSOUT=4
//SYSPRINT   DD SYSOUT=C
//*
```

## 4.2.12.5 Test Print Alternate Job Name Standard

For test jobs that are to be printed with the same characteristics as existing Production print, the alternate job name must be placed in the first eight bytes of the Software Developer name field on the test job card as illustrated in the example below. For test print that is to be routed to Inserter Operations for validation, the room parameter must be 'MAIL'. Test print that does not have a room parameter of 'MAIL' will be ineligible for Inserter testing. An example of this would be a Software Developer wanting to validate a change to remits that are normally produced from FS00J40D. In the test JCL, the alternate job name of FS00J40D would be used.

```
//FS00J40T  JOB  (11806,326,00000,PROD),'FS00J40D',
//       USER=FS00APP,
//   MSGLEVEL=(1,1),REGION=4000K,CLASS=7,MSGCLASS=0
/*JOBPARM K=90,ROOM=MAIL
```

```
//*
//JOBLIB          DD DSN=BC.NDVR.PROD1.EXECLIB,DISP=SHR
//*
//FS00S480     EXEC PGM=IEBGENER
//OUT1             OUTPUT FORMDEF=1727,PRMODE=DUPL
//SYSUT1         DD DSN=BC.FS00.COMBINED.EOB.G3763V00,DISP=SHR
//SYSUT2         DD SYSOUT=6,FCB=T580,OUTPUT=*.OUT1,RECFM=FBA,
//                       LRECL=111,OPTCD=J
//SYSIN            DD  DUMMY
//SYSOUT        DD SYSOUT=C
//SYSUDUMP   DD SYSOUT=4
//SYSPRINT     DD SYSOUT=C
//*
```

A print job that is to be considered TEST must follow one of the test rules listed below:

- The eighth character of the JOB NAME is T, $ or #,
- The fourth character of the JOB NAME is $ or # or @, or
- The SYSOUT class is E, F, X, 8 or 9.

A print job that is to be considered PROD must follow one of the PROD conditions listed below:

- The eighth character of the JOB NAME is A, B, D, L, M, N, P, Q, S, U, W or X, or
- The fourth character of the JOB NAME is J.

If none of the above conditions are met, the job is considered test.

Based on these criteria, IPW will place the TEST Overlay to print in the background of each document as well as enforce the use of form 2440 (pink test paper). A "Willowby IPW Test Request" must precede all print tests (Refer *Technical Standards — Applications > Print Operations > ICT Print Operations and Corporate Mail Services > InfoPrint Workflow Validation Procedure*.). For requests that are intended to be Inserter validated, the "Willowby IPW Test Request " will allow the IPW Admin staff to enable a onetime pass to allow the job to be eligible for Insertion. Failure to send a "Willowby IPW Test Request" will result in Test Print not being validated by Inserter Operations.

## 4.2.12.6 Inserter Mail Input Validation Standard

An Inserter validation is warranted when any programming, form or systems changes affect the following:

- Changes to address composition or positioning
- Changes to the form or body of the letter that affect logos, lines, graphics, or add/delete lines from the body of the letter
- Changes made in mail logic coding (e.g., OMR, 3of9 or 2D)
- Changes made to the programs that may increase/decrease page counts
- Any new letters or correspondence
- When a new system or program creates the same or new print file

To assure integrity in the Production mail stream, test materials and/or INFOPAC reports that simulate manual or automated Inserter mail input must be printed on pink paper. Test materials excluding INFOPAC reports must have the word "TEST" printed as an Overlay in the background of every document. Any test output requiring Inserter validation that does not meet these requirements will not be forwarded to manual processing (Mail Center) or automated processing (Inserter Operations) for further validation.

Scope of responsibility: Corporate Mail Services is responsible for the following:

- Reviewing the document format assuring it fits within the BCBSSC Address Template version 11/19/10,
- Machine readability,
- Presence of route code or minor LOB within Mail Run Data File (MRDF),
- Presence of Zip Code within MRDF, and
- Compliance with USPS regulations.

The customer is ultimately responsible for individual mail piece review to insure the contents are correct.

## 4.2.12.7 InfoPrint Workflow Standard

To assure integrity between lines of business within BlueCross, InfoPrint Workflow (IPW) allows designated individuals' entry to view only that information specific to their LOB while on the IPW system. A TSC Self Service Ticket named IPW ADD REQUEST FORM (located on the Self Service Portal) is required for any "add," "change," or "removal" of an individual from the IPW system. When validating IPW, providing test information to IPW, the INFOPRINT Workflow Validation Procedure needs to be followed in order to maintain integrity within the Production mail stream. Refer to *Technical Standards — Applications > Print Operations > ICT Print Operations and Corporate Mail Services > InfoPrint Workflow Validation Procedure*.

Changes to Production must be submitted to IPW.ADMIN no less than two weeks prior to Implementation to allow for coding and validation. If the output is to be mailed, validation and approval through the mail process must be enacted. All requests will require the manager's name, individual's name and line of business in which they need to be added. Any requests that do not meet the requirements will be considered incomplete and rejected with the reason for the rejection located in the comments field at the bottom of the form in order to prevent anyone from being added inappropriately.

### InfoPrint Workflow Change Control Standard

All new and existing documents that are to be produced with the intention of being mailed from Corporate Mail Services must be set up and validated through InfoPrint Workflow (IPW). In the event that the essential data TLEs are not embedded in the file from ADGS, then the IPW Admin staff must

create AFP Control files to identify each piece of essential data that resides on a document handled by Corporate Mail Services. Essential data is defined as document date, account number, check number and address. This is needed to achieve document-level tracking.

After setup is complete through IPW and the job has been successfully processed through IPW, the job will be printed in parallel (z/OS with IPW), Inserter or manual mail validated, and upon internal acceptance, ready for review from the corresponding customer area. The IPW Admin staff must be aware of any changes to new or existing documents at least two weeks in advance of any Production move once the job has been transitioned into the IPW system.

## 4.2.12.8 Standards for Production Mail Jobs

All Production mail jobs must run through InfoPrint Workflow/File Base Insert (IPW/FBI) process steps. The IPW/FBI system allows for mail job and mail piece tracking. Corporate Mail Services will reject mail Production not running through this system. Failure to validate through IPW/FBI can lead to processing failure, HIPAA violations or Service Level Agreements not being met.

Communication with ICT Print Operations and Mail Services is critical with changes related to Production mail jobs. All communications related to Production mail jobs are addressed to "IPW.ADMIN" and "INSERTER.OPERATIONS." Refer to *Technical Standards — Applications > Print Operations > ICT Print Operations and Corporate Mail Services > Inserter/Manual Mail Validation Procedure*.

### Data Properties/Minimum Requirements for IPW Processing

Production mail jobs must have mail set identifiers, document dates, address lines, IPW account numbers and check numbers (if applicable) on the first page of each mail set.

**NOTE** Future modifications of a mail set requiring a change in location of the key elements in the mail set must be validated and approved prior to implementing the change to Production.

1. A mail set identifier is required for each IPW/File Based Insertion (FBI) definition. A mail set identifier is a trigger identifying the start of the first page of a new mail set. A mail set identifier is three text characters in the same location on the document every time a new document set starts. For example, three asterisks residing in the same location on the first page of every mail piece could be used as a mail set identifier. The mail piece identifiers should be placed near the top of the document for ease of identification and less risk of error. It may be placed elsewhere if customer requirements dictate. The location used cannot vary. For documents created by ADGS, a system generated GUID is used as the mail set identifier.
2. Document date, address lines, IPW account numbers, and check numbers (if applicable) must be in fixed locations in the data and on the printed page relative to the mail piece identifier/trigger. The absence or presence of data cannot change the relative position of any of these fields. For example, if a letter has a three line address and the IPW account number starts three lines below the address, then when the address changes to four lines, the IPW account number should not be pushed down a line because of the fourth address line. It should only be two lines after the last address line in this case.

3. Mail sets are required to have "Page 1 of x" printed on the pages of the mail set. This must be in a fixed location.
4. IPW account numbers are defined as Provider IDs, Member numbers or any uniquely generated number, etc.
5. IPW account numbers must be at least four characters and no more than 16. IPW account numbers can consist of alpha and/or numeric characters but may not contain any spaces or special characters.
6. A Production mail job is defined in IPW as: z/OS job name, SYSOUT class, and PageDef. Additional print jobs coming from the same z/OS job must have a different SYSOUT or PageDef. Failure to do so will limit IPW's ability to detect duplicate print jobs, which could potentially cause a duplicate mailing.

## Special Requirements for Statement Jobs with Matching Checks

1. In the case of statement jobs with corresponding checks, every attempt should be made to keep the check field and "EFT" field separate. If check numbers and "EFT" numbers must exist in the same location, then they must be clearly defined as check or "EFT" on each document.
2. IPW account number and check number must match exactly between the statement job and the check job.

## Physical Properties

All printed work must be 18" x 11", continuous form white paper. This form number is 2075 for standard or 2369 for micro-perforated forms.

- All multiple page documents must use duplexing (printing on both sides of the paper) to reduce paper, postage and production costs. Pages are "physical" sheets of paper.
- All finished pieces must be 8 ½" x 11" and allow for a closed-end-tri-fold ("C" fold).
- All finished pieces must fit into a #10-size window business envelope.
- Mail sets smaller than eight physical pages are processed by automated mailing machines. Mail sets consisting of more than seven pages have two options.
- Programmatically split the mail set to multiple sets for automated insertion.
- Programmatically sort the print to a different SYSOUT class for manual insertion.
- All jobs that contain documents seven physical pages and less, containing 300 mail pieces or less, must be SYSOUT G or SYSOUT 7.
- All jobs that contain documents seven physical pages and less, containing 301 mail pieces or more, must be SYSOUT 6 or Sysout 2.
- Jobs that contain eight physical pages and above mixed with less than seven physical pages in the same print job will not be accepted for processing in Corporate Mail Services.

## Barcode Read Technology

All Production mail jobs on z/OS must have either a 3of9 or 2D barcode. Production mail jobs on IPW use only a 2D barcode.

- A 3of9 barcode contains machine instructions readable by a laser. This code includes:
    - Four characters, unique to the mail piece used for identification.
    - Demand feed/select insertion character.
    - Total physical page count within the mail set.
    - Sequence number.

- Modulo 43 check digit.

For application of these marks, you would need to contact the AFP group at PRINT.AFP.

- 3of9 barcodes are read in the upper left hand corner of the documents. 3of9 barcodes are 1 ¼" from the top and ¼" from the left side and must not overprint any other data on the page. There must be a clear zone around all 3of9 barcodes of no less than 1/2". Barcodes must not be more than 10 characters long. 3of9 barcodes are required on all checks where the Production mail requires matching payments. Templates can be provided by Corporate Mail Services at the email address "INSERTER.OPERATIONS."
- 2D barcodes are read from the upper left hand corner of the documents. The barcode is located 2" from the top and 1/6" from left and consists of the following information: AAAAAABBBBBBXXXXYYYYZZZZZZZ whereas the AAAAAA = job ID (zero filled on z/OS); BBBBBB = mail piece number; XXXX = page within the mail piece; YYYY = total pages in the mail piece; and ZZZZZZZ = the 3of9 data (See above in this subsection.) minus the Modulo 43 is at the end.

## Addresses

1. All documents scheduled to mail must contain a properly formatted destination address. The destination address should only appear on the first page of the document. Refer to *Technical Standards — Applications > Other Considerations > Corporate Mail Considerations* for proper address composition.
2. Addresses are allowed to contain up to four lines. If you need additional lines for addresses, the address block will need to be printed eight lines per inch (LPI) and be validated and approved for clearance in the window envelope.
3. Addresses cannot exceed six lines and the City, State and Zip Code must appear on the last line together.
4. Documents must be validated thoroughly for address placement and composition.
5. All new Production jobs use a current Production-outgoing envelope. All letters must be configured to be mailed in the standard P27191 envelope. The first page must adhere to the following standards:

    a. The address must be in a static location and cannot move from mail piece to mail piece.
    b. All letters must have a correctly formatted return address in the upper left hand corner of the document.
    c. A correctly formatted delivery address must be the only thing visible in the window of the envelope.
    d. The return address position, delivery address position, barcode position, and clear zone must meet the standard BCBSSC Address Template (Revised 11/19/10). Contact INSERTER.OPERATIONS for a copy of the template.
    e. All documents in the mail piece must contain set and page numbers (e.g., 1 of 3, 2 of 3, 3 of 3; in addition to sequential page numbering). Set numbers consist of the number of pages within a mail set or envelope. Sequential numbering is where every physical page within an entire job has an incremental number. For example, a print job with 200 pages total has 50 mail pieces. The first sheet would carry a sequential number 1 with the next page being number 2 finally finishing on number 200.

**NOTE** Printing at a network or desktop printer may not provide the same placement as ICT Print Operations' printers. A true validation for placement must be printed by ICT Print Operations and validated against the BCBSSC Address Template version 11/19/10. Corporate Mail Services can provide a copy of this template upon request.

## Validation/Approval

1. All jobs destined for processing at the Willowby facility must meet the minimum standards established to ensure accuracy and efficiency in print and mail processing.
2. All mail pieces must have a 3of9 or 2D barcode for z/OS print, and be configured to the File Based Insertion (IPW and DFWorks) process.
3. All jobs must pass Inserter validation.
4. For validation standards, refer to *Technical Standards — Applications > Print Operations > ICT Print Operations and Corporate Mail Services > InfoPrint Workflow Validation Procedure*. Questions regarding validation can be directed to the email addresses IPW.ADMIN and INSERTER.OPERATIONS.
5. New jobs and changes to current Production jobs must be validated through IPW/FBI and approved by ICT Print Operations/Inserter Operations prior to moving to Production. Corporate Mail Services must participate in the validation and sign-off before moving into Production. Validation request forms can be found in "Willowby IPW Test Request" via the TSC Self-Service Portal. This form must be filled out completely and submitted prior to validation.
6. Test material must be submitted as a test job to obtain the “TEST" watermark. Test material must be printed on pink paper (form 2440) as specified in the Willowby IPW Test Request. When form 2440 cannot be used (e.g., when the mail piece is to be printed on form 2369 micro-perforated stock), then justify the requirement in the Willowby IPW Test Request.
7. Test requests/jobs must have mail piece counts (number of envelopes to be produced) furnished by programming for accuracy. Validation should simulate a true Production run using Production data if possible. If this is not available, a minimum of 1000 mail pieces is required for all work in SYSOUT 6 and 2 and 500 mail pieces for work in SYSOUT G and 7.
8. Before any test job can be printed and released to Corporate Mail Services for validation, the mailing attributes need to be set and/or verified. This is accomplished via the Mail Enrollment Form which Corporate Mail Services will send to the customer. Once the mailing attributes have been defined, Corporate Mail Services will provide the attributes to ICT Print Operations prior to print and release.
9. Prior to Production approval, the customer must validate the contents of the test envelopes to ensure proper insertion.
10. Once validation is successful and approval has been obtained from IPW.ADMIN, Corporate Mail, and the customer, a Mail Enrollment Form must be completed by the business Unit/Cost Center Manager before the job will be mailed from Corporate Mail Services. Cost Center and LOB must be validated by the costing staff. (The Mail Enrollment Form may be obtained by contacting INSERTER.OPERATIONS.)
11. Corporate Mail Services reserves the right to refuse jobs based on noncompliance of standards.

## Subscriber and Member ID Cards

This section refers to plastic ID cards for members and subscribers. Production ID card files have to be provided from I/S Membership through the SEZI/ICRD system. The standards below refer to the layout and design of the physical cards.

A customer will provide a hard copy picture or electronic layout of the desired design of the ID Card containing both **constant data and variable data**. **Constant data** is the information such as the client's name, logo, etc., that appears on every ID Card. **Variable data** is the information such as the patient's name, ID number, or employer group name.

The ID Card Production staff will then recreate the layout of the constant data as well as the variable data that is to appear on the ID Card. This can take up to 72 hours per card. Once that is accomplished, ID Card Production will then place the ID card against a template to make sure the card meets specifications. When specifications are met, ID Card Production will require approvals from Corporate Communications and the customer to ensure what has been recreated is correct. Once approval is received from both areas, the ID card may then be placed into Production. A test card must be ordered before Production begins to ensure all data matches the corporate and customer specifications. The card will then be Production ready.

### Subscriber and Member ID Cards

| Card Specifications | |
|---|---|
| Actual Dimensions of the ID Card | Width — 3.3"<br>Height — 2.1" |
| Actual Dimensions of Printable Space Front | Width — 760 pixels<br>Height — 500 pixels |
| Actual Dimensions of Printable Space Back Non-Magnetic | Width — 760 pixels<br>Height — 500 pixels |
| Actual Dimensions of Printable Space Back Magnetic | Width — 760 pixels<br>Height — 350 pixels |
| Dots Per Inch (DPI) | Horizontal — 300<br>Vertical — 300 |
| Logo Format | File Type — PCX (PC Paintbrush)<br>Color — B&W (1-bit) |
| Allowable Colors | Front (Data) — BLACK. All constant data is to be printed on the front of the ID Card in BLACK ink.<br>Front (Logos) — PMS#7462 BLUE, RED, TEAL or GREEN. All constant data is to be printed on the front of the ID Card in BLUE ink.<br>Back — BLACK. All constant data is to be printed on the back of the ID Card. |

**Table 4-6** Subscriber and Member ID Cards

In addition to the above specifications (Table 4-6), a certain amount of customization can be done to meet customer specifications or requirements. All cards must adhere to Workgroup for Electronic Data Interchange (WEDI) standards and Blue Cross Association (BCA) mandates. Those standards and mandates are maintained by ID Card Production.

# 4.3 Package Naming Conventions for Endevor

All packages created for Endevor will follow the naming conventions as stated herein and will have one of the two following formats for the first 16 characters (Figure 4-21).

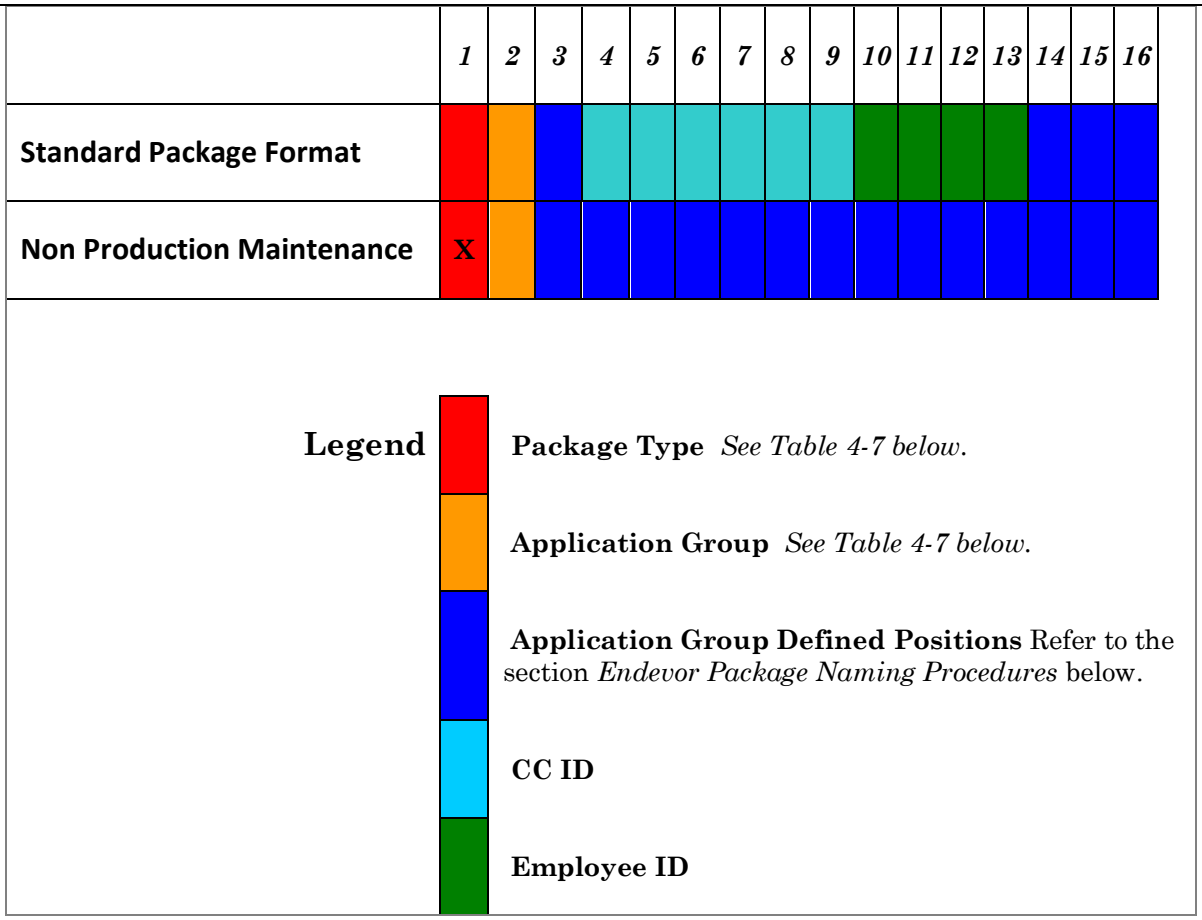



**Figure 4-21** Package Naming Conventions for Endevor

# 4.3.1 Non Production Maintenance Packages

**Non Production Maintenance Packages**

| Pos | Field Description |
|---|---|
| 1 | Package Type — X |
| 2 | Application Group — Same as “Standard Endevor Packages” |
| 3-16 | Application Group Defined Positions are defined by the Application Group. |

**Table 4-7** Non Production Maintenance Packages

# 4.3.2 Enterprise Workbench Packages

Enterprise Workbench (EWB) Packages will follow the standard 16-character format as defined above.

# 4.3.3 Additional Package Standards

1. **Enforcement:** Managers, Team Leads, Support, and all Approvers are responsible for enforcing the standard. No package should be approved that does not meet the standard.
2. **Audit Trail:** All executed packages must be retained for audit purposes. Executed packages cannot be deleted or reset. When a package fails for external reasons, the package can be re-executed. Failed packages that require modification of SCL or a CAST of the SCL will remain in EXEC-FAILED Status. A new package must be created with the SCL necessary to complete the intended actions. (An executed package is one that is IN-EXECUTION, EXEC-FAILED, EXECUTED, or COMMITTED.) Packages using the Promotion Option are also protected from RESET or DELETE after their first execution even though they show a status of IN_APPROVAL or APPROVED.
3. **Package Retention:** Packages are kept on the active Endevor Package file for 30 days. Executed packages are archived monthly. Unexecuted packages are deleted. Packages using the Promotion Option are always archived. Once the package is archived or deleted, the package name can be reused.

# 4.3.4 Production Procedures

## 4.3.4.1 Endevor Package Naming Procedures

### Overview

This chapter defines the package naming standard for all Endevor packages.

Position two of the package name defines the application group. Each group is provided four positions in the package name so that the application groups can utilize the standard and remain autonomous.

# Standard Endevor Packages

## Standard Endevor Packages

| Pos | Field Description |
|---|---|

**1** **Package Type** indicates the package content's destination.

| Package Type | Code |
|---|---|
| Special Production (Non Thursday Online) | # |
| Archive | A |
| Moves to FIXP1 | F |
| Production | P |
| Quality Test | Q |
| Promotion | R |
| Systems Test | S |
| Unit | U |

**2** **Application Group** identifies the area of I/S that controls the software or is mainly responsible for the move when multiple areas are involved in a move. These designations are determined by the I/S Standards Committee and must be one of those listed below.

| Application Group | Code |
|---|---|
| Companion Life | C |
| EDIG | E |
| G & A Systems | G |
| Leveraged Systems | H |
| Commercial Private Business | I |
| Medicare | M |
| ICT — Technical Support Areas | S |
| TRICARE | T |

**3** **Application Group Defined Positions** is defined by the Application Group. Refer to the subsections *Application Group C — Companion Life*, *Application Group E — EDI Gateway*, *Application Group G — G&A*, *Application Group H — Leveraged Systems*, *Application Group I – Commercial Private Business*, *Application Group M — Medicare*, *Application Group S — ICT and Technical Support Areas*, and *Application Group T — TRICARE* below.

**Standard Endevor Packages**

| Pos | Field Description |
|---|---|
| 4-9 | **CC ID** is the project or change sheet code. |
| 10-13 | **Employee ID** identifies the employee responsible for the move. This is normally the four character employee ID. For those areas with an Employee ID greater than four characters, the four characters used must be consistent within the Application Group. The Application Group's Employee Identifier is defined below in this subsection. |
| 14-16 | **Application Group Defined Positions** is defined by the Application Group. Refer to the subsections *Application Group C — Companion Life*, *Application Group E — EDI Gateway*, *Application Group G — G&A*, *Application Group H — Leveraged Systems*, *Application Group I – Commercial Private Business*, *Application Group M — Medicare*, *Application Group S — ICT and Technical Support Areas*, and *Application Group T — TRICARE* below. |

**Table 4-8** Standard Endevor Packages

## Application Group C — Companion Life

### Usage

All Endevor elements with a system code that begins with COMPAN0 are required to use this procedure.

**Companion Life Systems** are required to use this procedure for all Package Types.

### CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

### Employee ID Validation

The Employee ID of the person creating the package is to be used.

### Application Group Defined Positions

- Position three is defined as the Application Area Code.
- Position 14 is defined as the Package Content Code.
- Positions 15–16 are defined as the Package Identifier.

### Application Area Code

As shown in Table 4-9 below, this code is assigned based on the Endevor Subsystem being updated. If multiple systems are being updated, use the code for the system with the most updates.

**Application Group C Application Area Code**

| Code | Application Area |
|---|---|
| E | LCIS — Examiner |
| G | GCOM — Group-Com |
| L | LIFE — Membership and miscellaneous reporting |

**Table 4-9** Application Group C Application Area Code

### Package Content Code

The Package Content code is assigned based on the type of execution. See Table 4-10 below.

**Application Group C Package Content Code**

| Code | Package Content |
|---|---|
| B | Batch |
| O | Online |

**Table 4-10** Application Group C Package Content Code

### Package Identifier

The Package Identifier makes each package unique using any combination of characters (except blank) available for use in Endevor Package names (e.g., A-Z, 0-9).

The Identifier is not an indicator or execution order. An execution order must be stated when an execution is requested.

## Application Group E — EDI Gateway

### Usage

All elements in ECOMMER and EDIGATE System codes are required to use this procedure for all package types.

### CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

### Employee ID Validation

The Employee ID of the person creating the package is to be used.

### Application Group Defined Positions

- Position three is a constant value of "E."
- Positions 14–16 are defined as the Sequence Number.

### Sequence Number

Possible codes for Positions 14–16 are all numeric values 000–999.

## Application Group G — G&A

### Usage

All elements in CORPORT System codes are required to use this procedure for all package types.

### CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

### Employee ID Validation

The Employee ID of the person creating the package is to be used.

### Application Group Defined Positions

- Position three must be either "F" for Financial Systems or "H" for HR systems.
- Positions 14–16 are defined as the Sequence Number.

### Sequence Number

Possible codes for Positions 14–16 are alphanumeric values (e.g., V01-ZLR). Any unique combination of alphanumeric characters is allowed.

## Application Group H — Leveraged Systems

### CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

### Employee ID Validation

The Employee ID of the person creating the package is to be used.

### Application Group Defined Positions

- Position three is defined as the **Application Area Code**.
- Positions 14–16 are defined as the **Sequence Number**.

### Application Area Code

As shown in Table 4-11 below, this code is assigned based on the Endevor Subsystem being updated. If multiple systems are being updated, use the code for the system with the most updates.

**Application Group H – Application Area Codes**

| Code | Application Area |
|---|---|
| A | ADGS, ALGS |
| B | ABRS |
| O | BLNT |
| C | ARSS, CS98, IMCH, IMCN, IMCS, IMC1, IMC2, IMC7, IMC9, WMSI |
| M | MAIL |
| V | EVMN |
| W | ESDW |
| I | INFO |
| H | LCAH |
| L | LEDS |
| N | NSCS |
| 1 | PA01 |
| P | DGPR, GUCS, GUC1, PIMC, PIMS, PVDR, PRIC |
| Q | QMMS |
| R | RULE, GTMS |
| T | AUTO, TMCS |
| Z | ZIPC |

**Table 4-11** Application Group H – Application Area Codes

### Sequence Number

Possible codes for Position 14–16 are all numeric values 000–999.

## Application Group I – Commercial Private Business

### Usage

All elements in PRIVATE and PRIVVND System codes are required to use this procedure for R, U, S, Q, P, #, F, and A package types.

All Elements in SHARED0, ADVTECH, and TPHONE0 may use this procedure for the same package types.

## CCID Validation

All CCIDs used in the package name and for all actions within the package must be made up of the first six (6) characters of an active effort in the ISC TC Database.

All Non-JCL elements in the package must be in the effort inventory (i.e., ISC Consolidated Inventory) for that effort.

## Employee ID Validation

The employee ID in the package name must be found on the TAO extract file.

## Application Group Defined Positions

- Position three is defined as the Package Content Code.
- Positions 14–16 are defined as the Uniqueness **ID**.

## Package Content Code

As shown in the tables below (Table 4-12 through Table 4-20), the Package Content Code is assigned based on Package Type, Element Type, other element properties, and bundle considerations.

**Package Names For Unit, System, and non-Bundle QUAL Moves**

**Application Group I – Membership Package**

| Code | Description |
|---|---|
| Any | All Element Types |

**Table 4-12** Application Group I Membership Package

**(Types U or S or Q <non bundle> or R <non-PROD*>)**
***R packages moving to PROD must follow PROD standards***

**Package Names for Bundle Qual Moves**

**Application Group I Package Type Q**

| Code | Description |
|---|---|
| B | Proclibs |
| C | Doculibs |
| D | All other JCL components (jobdecks, cardlibs) |
| G | Elements types that generate components that can be included in other element types |
| I | Plan and Pack Binds |
| K | X elements |

**Application Group I Package Type Q**

| Code | Description |
|---|---|
| L | Onlines |
| M | Batch |
| W | Link decks |
| Y | Stand alone |

**Table 4-13** Application Group I Package Type Q

**Application Group I Element Type/Package Content Code Table (Bundle QUAL Moves)**

| Type | PCC | Type | PCC | Type | PCC |
|---|---|---|---|---|---|
| PROCLIB | B | PLANBIND | I | BCOBIMS | M |
| PROCLIBT | B | APSEXPS | K | COBOL | M |
| DOCULIB | C | XAPS | K | EASYTREV | M |
| DOCULIBT | C | XAPSDB2 | K | SASPGM | M |
| CARDLIB | D | XAPSIDB2 | K | LINKDECK | W |
| CARDLIBT | D | XAPSIMS | K | LINKMQ | W |
| JOBDECK | D | XASSEM | K | APSDATA | Y |
| JOBDECKT | D | XCOB | K | DB2FORMS | Y |
| VSAMLIB | D | XCOBDB2 | K | DB2PROCS | Y |
| VSAMLIBT | D | XMQPARM | K | DB2QUERY | Y |
| APSAPPL | G | CGCOBOL | L | DBDSRC | Y |
| APSREPT | G | OAPS | L | DBDSRCT | Y |
| APSSTUB | G | OAPSDB2 | L | DCLGEN | Y |
| COPYCICS | G | OAPSIMS | L | DDISRC | Y |
| COPYDB2 | G | OASSEM | L | DDISYMB | Y |
| COPYEZT | G | OCOB | L | IMSLIB | Y |
| COPYLIB | G | OCOBDB2 | L | ISPCARD | Y |
| NOSOURCE | G | OCOBIMS | L | ISPMLIB | Y |
| OAPSSCRN | G | BAPS | M | ISPPLIB | Y |
| OBJECT | G | BAPSDB2 | M | ISPSLIB | Y |
| OMAP | G | BAPSIDB2 | M | ISRCLIB | Y |
| PSBSRC | G | BAPSIMS | M | OTABLE | Y |
| SOURCE | G | BASSEM | M | SPUGLIB | Y |
| USERMACS | G | BCOB | M | | |

**Table 4-14** Application Group I Element Type/Package Content Code Table (Bundle QUAL Moves)

### Package Names for All JCL Prod Moves

**Application Group I Package Types P or R**

| Code | Description |
|---|---|
| M | Near Time related Elements. Any element that is used in a Near Time cycle that would cause a problem to the cycle if it were moved to PROD when the Near Time Cycle was active. |
| J | All Production JCL<br>PROCLIB, DOCULIB, JOBDECK, CARDLIB.<br>When a DOCULIB and JOBDECK with the same name are to be |

**Application Group I Package Types P or R**

| Code | Description |
|---|---|
| | moved, the elements must be in the same package. |
| T | All Test Cycle JCL<br>PROCLIBT DOCULIBT, JOBDECKT, CARDLIBT.<br>When a DOCULIBT and JOBDECKT with the same name are to be moved, the elements must be in the same package. |

**Table 4-15** Application Group I Package Types P or R

### Package Names for Prod Moves

**Application Group I Norman Moves (Thursday Move) Package Types P or R**

| Code | Description |
|---|---|
| M | Near Time related Elements. Any element that is used in a Near Time cycle that would cause a problem to the cycle if it were moved to PROD when the Near Time Cycle was active. |
| O | A package that contains at least one online element must use O. The package can contain other types, but must have at least one of the following types.<br>OAPSSCRN, OMAP, OTABLE, OAPS, OAPSDB2, OAPSIMS, OASSEM, OCOB, OCOBDB2, OCOBIMS, XAPS, XAPSDB2, XAPSIMS, XASSEM, XCOB, XCOBDB2, XMQPARM, and Online PLANBIND.<br>This allows O packages to contain related types and even batch types.<br>An Online Planbind is any Plan with an owner of CICSAPP. |
| P | Batch Planbinds and all Packbinds |
| Z | All other element types |

**Table 4-16** Application Group I Norman Moves (Thursday Move) Package Types P or R

**Special Moves (Other than Normal)**

### Application Group I Package #

| Code | Description |
|---|---|
| O | A package that contains at least one online element must use O. The package can contain other types, but must have at least one of the following types.<br><br>OAPSSCRN, OMAP, OTABLE, OAPS, OAPSDB2, OAPSIMS, OASSEM, OCOB, OCOBDB2, OCOBIMS, XAPS, XAPSDB2, XAPSIMS, XASSEM, XCOB, XCOBDB2, XMQPARM, and Online PLANBIND.<br><br>This allows O packages to contain related types and even batch types.<br><br>An Online Planbind is any Plan with an owner of CICSAPP. |

**Table 4-17** Application Group I Package #

### Application Group I Package Type P or R

| Code | Description |
|---|---|
| M | Near Time related elements. Any element that is used in a Near Time cycle that would cause a problem to the cycle if it were moved to PROD when the Near Time Cycle was active. |
| P | Batch Planbinds and all Packbinds |
| Z | All other element types not in a # Package |

**Table 4-18** Application Group I Package Type P or R

**Package Names for Moves to FIXP1**

**Application Group I Package Type F**

| Code | Description |
|---|---|
| Any | All Element Types |

**Table 4-19** Application Group I Package Type F

**Package Names for Transfer of Prod Elements to Archive**

**Application Group I Package Type A**

| Code | Description |
|---|---|
| Any | All Element Types |

**Table 4-20** Application Group I Package Type A

### Uniqueness ID

The Uniqueness ID is given to each package to make it unique using any combination of the alpha numeric characters (e.g., A–Z, 0–9).

## Application Group M — Medicare

### Usage

All elements in JMEDCAR, JMEDVND, MEDICA0, and MEDICB0 System codes are required to use this procedure for all package types.

### CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

### Employee ID Validation

The Employee ID of the person creating the package is to be used.

### Application Group Defined Positions

- Position three is a constant value of "M."
- Positions 14–16 are defined as the **Sequence Number**.

### Sequence Number

Possible codes for Positions 14–16 are all numeric values 000–999.

# Application Group S — ICT and Technical Support Areas

## Usage

All elements in JPGTEST, JPGVEND, SYSTEMS and DASDOPA System codes are required to use this procedure for all package types.

## CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

## Employee ID Validation

The Employee ID of the person creating the package is to be used.

## Application Group Defined Positions

- Position three is a constant value of "S."
- Positions 14–16 are defined as the **Sequence Number**.

## Sequence Number

Possible codes for Positions 14–16 are all numeric values 000–999.

# Application Group T — TRICARE

## Usage

All elements in CHAMPUS and CHAPVND System codes are required to use this procedure for all package types.

## CCID Validation

The project, change sheet, or appropriate maintenance CCID the modules were changed for are to be used.

## Employee ID Validation

The Employee ID of the person creating the package is to be used.

## Application Group Defined Positions

- Position three is a constant value of "R."
- Positions 14–16 are defined as the Sequence Number.

## Sequence Number

Possible codes for Positions 14–16 are all numeric values 000–999.

# 4.4 GitHub Enterprise Standards

## 4.4.1 Repository Security & Compliance Requirements

Each repository in GitHub Enterprise must meet specific security requirements. If a requirement does not apply to a repository, a waiver must be submitted and maintained. Most of these requirements are enforced by GitHub Enterprise policies and bots set up by IT Business System (ITBS) and managed through the InformationSystems/InformationSystems-Repository-Settings repository.

For each repository, the primary branch (generally "master") will be used to represent the production state of the application. The primary branch will be used for inventory and vulnerability management.

Additional information on the requirements can be found in the latest documentation from Security, Risk and Compliance Assurance (SRCA) on the following topics as they pertain to source control management:

- Inventory Control
- Least Privilege
- Separation of Duties
- Second Set of Eyes
- Other Change Management
- Data Protection

Information on the current compliance checks (and their related corporate policies) performed on repositories in the Information Systems organization can be found at these links:

- Pull Request Requirements: https://git.bcbssc.com/InformationSystems/InformationSystems-Repository-Settings/blob/master/ORG_PULLREQUEST_YML.MD
- Consensus Requirements: https://git.bcbssc.com/InformationSystems/InformationSystems-Repository-Settings/blob/master/ORG_CONSENSUS_YML.MD
- Repository Settings Requirements: https://git.bcbssc.com/InformationSystems/InformationSystems-Repository-Settings/blob/master/ORG_SETTINGS_YML.MD
- Governance Waivers: https://git.bcbssc.com/InformationSystems/InformationSystems-Repository-Settings/blob/master/ORG_GOVERNANCE_YML.MD

## 4.4.2 Personal Repository Usage

Developers may create personal repositories in GitHub Enterprise for limited use projects, including:

- Proofs of Concept/Proofs of Technology
- Personal scripts or utilities
- Forks of other repositories in order to provide contribution via a Pull Request

Applications cannot be deployed to any environment (test or production) from a personal repository. If an application is started in a personal repository, it must be transferred to a managed organization in GitHub Enterprise (for example, the InformationSystems organization) and meet all security and compliance requirements before deployments are initiated.

## 4.4.3 Sensitive Data Handling

Any test data or configuration files that are committed to GitHub Enterprise for the purposes of running test cases MUST be scrubbed of any sensitive data prior to committing. Sensitive data includes:

- ACH Bank Account data
- Protected Health Information (PHI) data
- Personally Identifiable Information (PII) data
- Payment Card Industry (PCI) data
- Credentials
- Keys and access tokens

## 4.4.4 Branch Protections

All repositories must enable protections on the primary branch (typically master). This prevents anyone from pushing and changes directly to that branch. Changes must be submitted as a Pull Request from a development or test branch into the primary branch and be approved by at least one reviewer.

It is highly recommended that other types of intermediate branches also enable protection rules. For example, these recommended protected branches may include:

- Project integration branches
- Release integration branches

## 4.4.5 Repository Contents

Every repository must contain the following two files:

- README.md — This file describes the project in relation to its SMI identity, and gives basic information on usage and architecture.
- CONTRIBUTING.md — This file communicates guidelines, expectations, and standards for anyone contributing to the repository as well as the process for submitting changes. This is especially important for repositories that are maintained by multiple teams or areas. A sample CONTRIBUTING document can be found here: https://git.bcbssc.com/InformationSystems/application-technical-standards/blob/master/examples/CONTRIBUTING.md.

# 4.5 Open Source Inventory Standards

At minimum, all BlueCross BlueShield of South Carolina (BlueCross)-developed code within managed repositories should be scanned by the software composition analysis tool for open source dependencies for the production version. A current bill of materials generated from the software composition analysis tool is required for each internally developed code base and configuration files in the code management tool in order to establish an inventory of Open Source Software and manage risk associated with that software.

## 4.5.1 Open Source Libraries

Any Open Source Libraries used in internally developed code bases must adhere to the rules (policies) established by the Open Source Review Board (OSRB) in the software composition analysis tool. The current policies can be found at https://bluesc.sharepoint.com/sites/AppSecOneStopShop/SitePages/Open-Source-Policies.aspx.

## 4.5.2 Software Composition Analysis

The source code repository name should be an exact match to the software composition analysis project name in order to ensure data alignment of the inventory of code bases.

The software composition analysis tool project version containing the production code should be marked as Released.

The software composition analysis must indicate the appropriate distribution level:

- External — The software is published in either source or binary file format outside of BlueCross. For example, in a third-party mobile app store.
- SAAS — The software is hosted as a running service and can be accessed (over a network) by non-employees.
- Internal — The software is only made available to employees.
- Open Source — Code that is released as Open Source Code requires CIO approval through the OSRB. For details refer to *Systems Architecture > Free and Open Source Software*.

## 4.5.3 Repositories

Repositories should contain either all vendor code or all internally developed code and should be tagged with the appropriate SMI IDs.

- Any repositories with a requirement for co-mingled vendor code and configuration should work with the OSRB for guidance on maintenance and bill of materials.
- Internally developed code and configuration files written in support of a vendor product in SMI should have a child SMI ID with the vendor product as the parent SMI ID. The repository with the support code or configuration files should be tagged with the appropriate child SMI. This includes repositories that consist of files to assemble Docker container images for vendor products to run within the BlueCross environment. Refer to *Systems Architecture > General Architectural Standards > System Master Index* and *Systems Architecture > General Architectural Standards > Vendor Software Usage* for additional details.

# 4.6 Content Manager OnDemand Folder and Report ID Naming Standards

All searches in Content Manager OnDemand (CMOD) are performed at the Folder level. Because of this, the only search interface a customer will see is the Folder interface. The Folder search screen and the Folder search hit list display folder index labels, not Report ID index labels. The folder search screen fields and display fields can be displayed in any order by defining key sequences for the search (query order) and display (display order) in the CMOD administration system. The Folder results set can be ordered by defining a display sequence.

Since the order in which the fields are stored in the database is not significant, developing standards for specific fields to be defined in specific index key numbers (positions) is not needed. However, it is important to develop naming conventions for certain types of data (Social Security Number (SSN), etc.), and to develop folder search order and display order standards where practical.

The Enterprise Content Management (ECM) team recommends naming standards for Index/Key labels for new data that is created.

## 4.6.1 Report ID (Application) Naming Standards

A Report ID is defined to CMOD by the ECM CMOD Administrator based on the characteristics of the data or documents that will be stored to it. Note that CMOD uses the terms **Application** and **Application Description**. The terms Report ID and Report ID Description will continue to be used at BlueCross BlueShield of South Carolina (BlueCross) because of customer familiarity with these terms and because Resource Access Control Facility (RACF) continues to refer to this resource as a Report ID.

The CMOD Report ID naming standard was developed to **simulate** Report ID versioning and to allow the existing RACF Report ID definitions to be supported with minimal impact.

CMOD Report IDs are defined with a six-, seven-, or eight-character **base** and a four-character **suffix** consisting of **-Vxx** where xx represents the equivalent of a version (Table 4-21). For example, Report ID PVTCLAIM version 01 is CMOD Report ID PVTCLAIM-V01.

**Report ID Naming**

| Character Position | Description | Examples |
|---|---|---|
| 1-3 | LOB Prefix | CBA — Companion Benefits Alternatives<br>PVT — Private Business<br>TXS — TRICARE South |
| 4-8 | Description | ABILL, RGSUB, SDAKC |
| 9-12 | Version | -V01, -V02, -V03 |

**Table 4-21** Report ID Naming

For Non-Host PC documents loaded to CMOD such as PDF, TIFF, JPG, XLS, and TXT, the last three characters of the **base** should be the file type (e.g., HNFADTIF, MTXALTXT, MTXDODOC, and CPCAUPDF). Note that this is not a CMOD requirement but is done for informational purposes only.

The Report ID Description should contain a description of the LOB that will view the data, the type of data, and the four-character SMI System ID of the area that loads the data to this Report ID (Table 4-22). The SMI System ID has not been added to the Description in the past, but it will be used with new CMOD Report IDs. This will allow ECM to identify the I/S area that loaded the data years after the load job is no longer active.

**Examples of Report IDs and Report ID Descriptions**

| Report ID Name | Report ID Description — SMI System ID |
|---|---|
| AZBDFXGT-V02 | ARIZONA VRU LIBRARY — ALGS |
| MCDFXTIF-V01 | MEDICAID FAXES — MCCS |
| PVTMICST-V02 | MICHIGAN MED ADVANTAGE CUSTOMER SERVICE LETTERS — ALGS |

**Table 4-22** Examples of Report IDs and Report ID Descriptions

The ECM CMOD Administrator maintains a listing of Report ID prefixes (e.g., HRM, MCD, and PVT) and will work with I/S Areas or LOBs requesting new Report IDs to ensure the correct prefix is assigned.

The Report ID Name cannot include the following special characters:

’ “ % [ ] ! @ # $ & * ?

Additional Report ID characteristics are listed below:

- Report ID Name — 12 characters (six-, seven-, or eight-character base + four-character version)
- Report ID Description — 60 characters

## 4.6.2 Folder Naming Standards

All folders in CMOD are defined with a Folder Name and a Folder Description.

Since all searches in CMOD are performed at the Folder level, all Report IDs must be mapped to at least one Folder. The ECM CMOD Administrator will always define a folder of the same base name as the Report ID, referred to as a **Primary** Folder. In addition, other folders may be defined that have multiple Report IDs mapped to them.

## 4.6.2.1 Primary Folders

The ECM CMOD Administrator will always define a Folder with the same name as the Report ID **base** name and map the Report ID to that Folder (Table 4-23). In addition, the Primary Folder index keys will always be defined to match the Report ID index keys.

**Examples of Report ID Names and Matching Primary Folder Names**

| Report ID Name | Primary Folder Name to which Report ID is Mapped |
|---|---|
| AZBDFXGT-V02 | AZBDFXGT |
| MCDFXTIF-V01 | MCDFXTIF |
| PVTMICST-V02 | PVTMICST |
| PVTNOTPD-V01 | PVTNOTPD |

**Table 4-23** Examples of Report ID Names and Matching Primary Folder Names

Primary Folders will only have one Report ID mapped to the folder. This will be the standard going forward in CMOD.

If requested, a new Primary Folder can be defined with a different name than the Report ID. For example, a new PVTBBILL.PRIMARY.FOLDER could be defined and the PVTBBILL Report ID mapped to it. This would be in addition to the existing PVTBBILL folder that was migrated to CMOD.

## 4.6.2.2 Collection Folders with Multiple Report IDs

Additional Folders referred to as "Collection" Folders may be defined that contain multiple Report IDs. This allows multiple Report IDs to be searched at one time. The Report IDs must contain at least one search field that matches the Folder Fields so it can be mapped to the Folder. For example, the Report ID "Claim Number" search field should be mapped to the Folder "Claim Number" field and be defined with the same length. The DMS CMOD Administrator or a DMS Analyst will work with customers to ensure the Folder mappings are logical.

Folders will be named with periods in place of spaces for multiple words.

*Examples*:

```
HNF.ALL.TIFFS

HOSPITAL.REMITS

ITS.ABC.REPORTS
```

*Examples of Folders with multiple Report IDs*:

```
Folder BOP.COMM.ALL.CLAIMS contains the following Report IDs:

BOPAFTIF-V01 PGBA BOP FACILITY AUTHORIZATIONS
```

```
BOPCFTIF-V01 PGBA BOP CORRESPONDENCE FEDERAL BOP

BOPCPTIF-V01 PGBA BOP CORRESPONDENCE PROVIDER

BOP04TIF-V01 PGBA BOP UB 04 CLAIMS

BOP04TIF-V02 PGBA BOP UB 04 CLAIMS

Folder ACT.EOBS contains the following Report IDs:

FLBEOB-V01 BC/BS FL EXPLANATION OF BENEFITS

FLBEOB-V02 BC/BS FL EXPLANATION OF BENEFITS

HMBEOB-V01 HMO BLUE EOB

PVTEOB-V01 BC/BS EXPLANATION OF BENEFITS

PVTSTE-V01 STATE GROUP EXPLANATION OF BENEFITS
```

The Folder Name cannot include the apostrophe (‘), percent (%), left bracket ([), right bracket (]), or double quote (") character.

**Folder Characteristics:**

- Folder Name — may contain up to 60 characters, with periods in place of spaces
- Folder Description — may contain up to 120 characters
    - Primary Folder Description should match Report ID Description
    - Collection Folder Description should describe the type of documents included

## 4.6.3 Folder and Report ID Index Field Naming Standards

Although CMOD searches are all performed at the Folder lever, both Folder and Report ID index fields are included here, as the Report ID fields must match the Folder fields to which they are mapped. CMOD supports Folder/Report ID Search Fields and Folder/Report ID Display Fields. The CMOD Administrator or a DMS Analyst will work with areas that request new Folders and Report IDs to ensure consistent index field naming and index field order between similar Report IDs and Folders.

Primary Folder index fields will always be defined to match the Report ID index fields mapped to it. Report IDs mapped to a Collection Folder must have at least one matching key in order to be mapped to the folder.

### 4.6.3.1 Folder and Report ID Search Fields

All Folder Search Fields appear in the order defined by the CMOD Administrator. The Posting Date will always be the last search field and will provide a search data range for the folder search (from date and to date).

## 4.6.3.2 Folder Display Fields

The result of a folder search will be a document list (hit-list), which contains a list of document indexes that met the folder search criteria. Fields that are defined as display only will appear on the document list after the search is completed but will not appear as searchable fields.

The Posting Date will always be the first field on the left on the display list, and the Report ID will always be the second field on the left of this list. The remaining fields will be displayed in the order defined by the CMOD Administrator.

## 4.6.3.3 Folder Field Labels

Many BlueCross departments within several Lines of Businesses load data to and view data from CMOD. There are thousands of Folders and Report IDs defined in CMOD production with hundreds of different document types. Each department's data is different. Even standard document types such as Explanation of Benefits (EOB), Remits, and Claims do not always contain the same data, nor will every department need to search on the same index data. In addition, data in many reports may contain names with different variations for the same data (CLAIM NUMBER, CLAIM NO., CLAIM #), etc. It is therefore not practical to require naming standards across all Lines of Business (LOBs).

However, DMS does recommend that each department maintain consistency in the naming of Folder Field labels and the order in which they appear within the department for similar document types. This is especially important when multiple Report IDs are mapped to the same Folder.

*Example:*

```
EOB Folder              EOB Report ID

CLAIM NUMBER            CLAIM NUMBER

SUB-ID                  SUB-ID

PATIENT NAME            PATIENT NAME

NOTICE DATE             NOTICE DATE

ID CARD NUMBER          ID CARD NUMBER
```

**Folder Field Characteristics:**

- Folder Field Name, or Label — may contain up to 60 characters
- Folder Field Value for String Fields — may contain up to 254 characters
- Folders and Report IDs currently may be defined with up to 16 index keys.

# 4.6.4 Electronic Data Archive Implications

Documents uploaded to CMOD from a Non-Host server via Electronic Data Archive (EDA) or Message Queue (MQ) are limited to 16 Folder Keys, which each can be up to 64 characters in length.

# 4.7 HostBridge Development

## 4.7.1 General Discussion

HostBridge is integration software that provides specific functionality for accessing existing Host CICS transactions as web services using JavaScript. It is used for the new development of all real-time Host-based integration going inbound to a host system or outbound from a host system.

The HostBridge JavaScript Engine (HB.js) is a server-side JavaScript engine running inside CICS. It can execute an existing transaction by using HostBridge Base XML or call any CICS API. HB.js provides the ability to quickly develop, test, and deploy reusable web services integrated with CICS applications. Both a development facility and runtime engine, HB.js provides all the capabilities z Systems customers need to rapidly develop and deploy reusable web services and/or scripts.

## 4.7.2 HostBridge Compatibility

The purpose of this section is to note certain restrictions or idiosyncrasies associated with the current release of SOA Express. There are no restrictions at this time.

HostBridge uses a custom JavaScript engine originally based on the Firefox JavaScript engine, SpiderMonkey. As such, it is not always compatible with the latest ECMAScript standards. Documentation for the HB.js JavaScript Engine can be found on the official vendor Wiki page https://wiki.hostbridge.com/.

HostBridge interacts with CICS screens via Basic Mapping Support (BMS) using named fields. Any OAPSSCRN ENDEVOR element, which has not been generated with BMS field names pulled from the OAPSSCRN file based on programmer-assigned names (circa March 2019), will require a onetime recompile before it can be accessed by HostBridge web services.

# 4.8 IT Business Systems Host

The primary responsibility of IT Business Systems (ITBS) Host is to assist I/S staff members in the resolution of difficult or unusual technical problems. The problem need not involve an ABEND or a memory dump. Assistance is also available on a wide variety of other problem types. These include, but are not limited to:

1. JCL — Errors, techniques, design strategies.
2. Linkage Editor — Errors, operations, usage.
3. Cobol — Features and their use, coding efficiencies, compile time errors, etc.
4. Access Method — BSAM, QSAM, BDAM, VSAM, etc. Internal operation, selection of appropriate method, efficient use of method.
5. IBM Utilities — Capabilities and usage of each.

## 4.8.1 Limitations

Normally, problems should be referred to the ITBS Host team only after a reasonable effort on the part of the I/S staff member involved, by submitting an IT Business Systems Help form. The individual should also utilize the expertise available in his own section.

## 4.8.2 Exclusions

1. Problems involving "IMS" should be referred directly to Data Base Administration, using the DB-HELP form on the TSC Self-Service Portal. Control statement formats and usage, program design questions, etc. Persons in that area can handle involving these products best.
2. Questions concerning the Data Dictionary; control statements, meaning of data dictionary components, etc., should be referred to the Data Base Administration, using the DB-HELP form.
3. "Easytrieve" questions should be directed to the Data Base Administration using the DB-HELP form.
4. All applications support questions for AMB-related issues can be reported via the IT Business Systems Help form on the TSC Self-Service.

## 4.8.3 Specific Procedures

1. During prime shift the I/S staff member should spend at least 30 minutes to one hour on the problem. If this fails to yield significant progress or insight, then at least one other senior member of the individual's section should be consulted. Naturally the more experienced section members are preferred for this consultation phase. The problem then may be referred to ITBS Host. This procedure applies for both test and production problems.
2. During non-prime shift hours, one additional step is required. The individual's section manager must be notified. The manager then may request ITBS Host assistance.
3. If at any time the appropriate ITBS Host individual is not available and immediate problem resolution is required, then any manager may contact Systems Support for assistance.
4. Programming personnel must not request assistance from Systems Support other than as outlined above. Requests must be made using the IT Business Systems Help form.
5. The IT Business Systems Help form must be used to request assistance. The Terminal-ID field is required.

# Chapter 5 Business Systems Considerations

This chapter contains standards that are applicable to technical considerations needed for Business System Applications used at BlueCross BlueShield of South Carolina.

# 5.1 Data Warehouse and Reporting Standards

## 5.1.1 Informational Reporting

### 5.1.1.1 Purpose

The Informational Reporting section of the ISSM documents the standards that govern the development, use, and deployment of informational reporting applications. This includes the components, processes, tools, and cost impacts of Informational Reporting as they relate to BlueCross BlueShield of SC.

These guidelines and policies affect both end customer reporting initiatives and the role of Information Systems.

There are two major categories of Application Systems that pertain to reporting:

- Operational Systems: Those systems that directly support the transactional business functions of the organization by providing automation for those functions and provide operational reporting in support of those functions.
- Informational Systems: Those systems that create specific informational reporting as well as the creation of Informational Data Repositories for ad hoc reporting.

These guidelines apply to the Informational Data Repositories created by Information Systems.

### 5.1.1.2 Types of Information Data Repositories

#### Raw Data Warehouse

The Raw Data Warehouse is comprised of all DB2 informational tables and the accumulation of transactional and non-DB2 informational files (i.e. IMS, VSAM, QSAM, flat files, Access). The Raw Data Warehouse contains the unchanged data from each major transactional operational master file. Each of these files is individually accessible.

External data, such as pharmacy data and external data warehouse data, is considered part of the Raw Data Warehouse.

#### Informational Data Warehouse

The Informational Data Warehouse contains data that is based on business drivers. Informational Data Warehouses are built to contain decoded, logically grouped and summarized data extracted from the Raw Data Warehouse. To minimize the depth of data knowledge needed to make use of the data, derived data elements can be added to the extracted raw data for loading into the Informational Data Warehouse during the ETL (Extract, Transform, and Load) process. The informational files are not updated by sources outside of the operational systems.

Data quality checks and/or referential integrity checks (validating data against a set of valid values) will be performed on the data being extracted before loading the Informational Data Warehouse.

Ultimately all Informational Reporting will use Informational Databases as the data source.

### DataMart

DataMarts are datasets created to address the specific needs of a small customer base and are accessible by specific customers or specific areas.

Specialized DataMarts are created on a limited basis and are based on business needs for specific reporting and presentation requirements that cannot be met by the Raw Data Warehouse or the Informational Data Warehouse. Data in DataMarts is extracted from Informational Data Warehouses or from the Raw Data Warehouse.

Data quality checks and/or referential integrity checks will be performed on the data being extracted before loading the DataMarts.

### Data Garages

Data garages are data repositories created, owned, and maintained by each individual customer. As customers extract and massage data for their own use, the data can be stored on the customer's desktop or server.

## 5.1.1.3 Data Modeling

When developing or creating a new informational data warehouse or data mart, data modeling of the structure is necessary to understand the data relationships between the differing data elements. Data modeling is a process where the relationships between entities and their attributes are discovered and documented. Diagrams that show the relationship between a set of Data Structures express the model.

The logical data model is developed and maintained by the responsible application development areas. Database Administration is the point of review and approval of the logical model before the physical tables area created.

Multiple data sources may be involved in developing a data warehouse solution. The data must be modeled for determining and documenting the data relationships between the sources.

## 5.1.1.4 Data Sources for Informational Reporting

Production operational files are the data "source" for informational reporting. All Operational System Master Files will have a corresponding Informational File (preferably relational) that contains all operational data elements. Since reports will be run from Informational Files, Informational Reporting activities will not degrade production processes.

The operational systems areas own the operational data extract processes. The informational and reporting area(s) will use the outputs from these processes.

The Informational Reporting systems area owns the data translation and load processes for the informational structures.

Database utility loads and unloads of data are an acceptable means for data extraction and for use in loading the data to database structures. If business requirements cannot be met by the use of the database utilities, program coding is required. The exact manner for extracting and loading databases is determined on a case-by-case basis.

Unloads of production data and "change data capture" are two methods for obtaining production operational data for informational use. The term "change data capture" or "CDC" is a data warehouse term that refers to collecting data as it is changed in the operational system. Change data capture defines the process of capturing additions, modifications and deletes that occur against a production operational/transaction file.

Data quality checks and/or referential integrity checks will be performed on the data being extracted before loading the DataMarts.

To preserve scalability, data warehouses must be designed to be linear scalable. This means that a Data Warehouse will have the ability to grow without deleting and reloading the repositories. The timing of data in the data warehouse is related to business need. The data warehouse design allows the migration of data off the data warehouse.

## 5.1.1.5 Data Dictionaries

Data dictionaries detail the data element attributes (i.e. record position, length, etc.). A customer centric description of each element will be created for each file or table structure. The descriptions must be customer friendly formats stored for long-term support, and maintained each and every time a change is made to the operational system that has impact on the data dictionary. For a more complete description refer to *Procedures & Tools > Host Tools > Developer Tools > Application Interfaces & Utilities*.

## 5.1.1.6 Data Access

The RACF Administrator maintains security for data access. Table level access and data view access is supported.

Informational Middleware is the software class that allows a reporting tool to connect to the data structure. Currently six standard middleware components are recognized for data access of reporting information. They are:

- ODBC
- DRDA
- STAR SQL
- ORACLE NET
- DB2 Connect
- MQ

## 5.1.1.7 Presentation of Data

The following tools are recognized standard tools for report preparation and presentation in the production environment:

- Easytrieve Plus

- QMF/SQL
- SAS
- Crystal Enterprise
- Business Objects
- DocuDirect
- DocuAnalyzer
- ACL (Audit Control Language)
- AMB Report Writer

## 5.1.1.8 Implementation of Informational Solutions

Single source report feeds and common report formats are to be used whenever possible to reduce risk and to allow for ease of maintenance and growth.

After production reports are created, a file can then be used to produce the final INFOPAC report. This allows for ease of re-creation when necessary as well as allowing staff with report file access to use the raw report data without having to use products like Mobius DocuAnalyzer to retrieve raw statistics from the reports.

Existing corporate end user computing assets will be used where practical. All new end-user reporting tools must be evaluated through the Information Management (IM) Committee and cost modeled for impacts to include infrastructure, security and licensing impacts. For end user reports to be scheduled into production, Information Systems will review, migrate and support those reports at a Work Request level.

For Internet deployment or distribution of information to external audiences via browser technology, Business Objects is the standard tool.

Report inventory audits by LOB are to be performed annually. Any unused or outdated reports must be removed from the production cycles.

# Chapter 6 File Management Standards and Guidelines

The purpose of this section is to provide standards and guidelines concerning file management, which encompasses the efficient utilization of Direct Access Storage Device (DASD) and the management of tape utilization.

# 6.1 DASD Enforcement Standards

**DASD Enforcement Standards**

| Action: | All File Conditions: | |
|---|---|---|
| Delete | A. File with invalid naming conventions. | |
| Delete | B. File with invalid dsorg. | |
| Report | C. File using poor block size. | |
| Fix-w | D. File with write verify active. | |
| Release | E. File over allocated. | |
| **Action:** | **Test File Conditions:** | |
| Delete | A. Test file not on test packs (e.g., B93T?? and T?????) | |
| Delete | B. Test file not accessed 186 days. | |
| Delete | C. Test file not cataloged. (SMS requires all files be cataloged.) | |
| **Storage Group Archive Action** | **After Days Not Used** | **Days Retention** |
| Test | 30 | 186 |
| DB2T | 45 | 365 |
| **Action:** | **Production File Conditions:** | |
| Delete | A. Production file not used 100 days. | |
| Delete | B. Production file not cataloged. (SMS requires all files cataloged.) | |
| Archive | C. Non-current GDGs using UNIT=SYSDG, retention 100 days. | |
| Archive | D. UNIT=SYSDG file non-used in 10 days, retention 90 days. | |
| Archive | E. UNIT=SYSDA, CREDT+1, retention 100 days. | |
| Archive | F. Private Production Storage Groups. | |
| **Storage Group (s)** | **After Days Not Used** | **Days Retention** |
| DB2-Production 120 1825 | DB2-Production 120 1825 | DB2-Production 120 1825 |
| Private Production | 45 | 55 |
| **Storage Group (s)** | **After Days Not Used** | **Days Retention** |
| Operating System | 30 | 1825 |
| DB2P BB30???S | -- | 100 |
| TSO.emp/grp ID.PDS | **** See below. | 365 |

**** TSO personal and group libraries or Partitioned Datasets (PDSs) will be allocated by the Technical Support Staff and will not be greater than 185 tracks. These PDSs are not to be re-allocated larger than 185 tracks or with secondary allocation due to problems encountered in the backup and recovery of PDSs over 185 tracks in size.

**Table 6-1** DASD Enforcement Standards

Enforcement follows (Table 6-2):

**Enforcement**

| Action: | TSO Personal and group PDS File Conditions: |
|---|---|
| Delete | Not allocated by Tech Support |
| Archive | Owner no longer with BlueCross BlueShield of South Carolina |
| Not restored at hot site | Allocation greater than 185 tracks |

**Table 6-2** Enforcement

## 6.1.1 Major Points Concerning DASD Management

Enforcement of the DASD standards will be accomplished by both regularly scheduled jobs and impromptu jobs as deemed necessary by the Storage management team.

Do not specify a retention period or expiration date on any DASD file.

Pre-allocation on storage and/or public volume to hold or reserve space for future use is not allowed. Space should be requested when needed, used, and deleted as soon as possible. This will help ensure available space for all production systems.

Few exceptions will be made for the non-use standard of 186 days for Test and 100 days for Production. Use tape for long term storage.

It is strongly suggested that JCL space parameters be symbolic. The TEST Storage Group is limited in size, and without the ability to down size your allocations for validation, allocation problems will result.

The Project Managers/Application owners need to inform Storage Admin 30 days or earlier in advance of large DASD requirements. A "DASD REQUIREMENT" email is mailed monthly as a reminder and as a statement; THIS IS WHAT WE KNOW ABOUT! If your project is not listed, you can get on the list by filling out the I/S Portal form "DASD-ESTIMATES." DASD is ordered by this list. If you are not on the list, then your DASD is not on the floor and shipment delays may impact your project's due dates. DASD requirements in excess of one (1) GB (1200 cylinders) need to be on the list.

## 6.1.2 Major Points Concerning Production and Test Files

### 6.1.2.1 Updating Production Files

The only jobs allowed to update production files are regularly scheduled production jobs and approved "P" jobs. Production files and databases are only updated from production online applications. (Refer to

*Technical Standards — Applications* > *Application Coding* > *Enterprise Server – JCL* > *Design Considerations* for additional information.)

## 6.1.2.2 Copies of Production Files for Validation

Full vs. Partial copies — factors for consideration:

- File size and/or usage should be kept to a minimum acceptable for effective validation.
- Full production copies are allowed under the following conditions:

    - Extract criteria cannot be determined in advance (e.g., Benchmark, stress testing, etc.)
    - Validation will be random in nature and cannot be confined to a range of records or file keys.
    - The cost of extraction versus duplication cannot be justified.

Within QA cycles and QA CICS regions, the need for full copies is understood due to the requirements to duplicate or mimic Production. If the criteria for a full copy cannot be met, then a partial copy must be created. The copy will be created with records extracted in the most efficient and cost effective method available (e.g., Easytrieve Plus, COBOL, Access/DB2, etc.).

# 6.2 DASD File Allocation Information

## 6.2.1 BlueCross DASD Structure

Datasets reside on DASD packs accordingly:

| | |
|---|---|
| A. Cycle Storage — "Unit=SYSDA" | Datasets used for an entire cycle and deleted by the end of the cycle. The maximum size should be no more than 600 cylinders. |
| B. Cycle Dependent — "Unit=SYSDG" | Datasets needed for another cycle. Does not have to be a GDG file. If GDG, just the current (0) generation, is kept on DASD unless an exclusion table update has been requested. The maximum size should be no more than 250 cylinders. |
| C. Sortwk/public — "Unit=SYSDA" | Temporary sortwork files and datasets denoted by DSN=&&dsname, which are deleted at end of job. The maximum size should be no more than 60,000 cylinders total sortwork space, and 600 cylinders primarily for each sortwork DD statement. |
| D. Test — "Unit=SYSDA" or "UNIT=SYSDG" | Files used in Software Developer validation. All files on test packs are enforced as test files. Re-naming a test file to a production name will not protect it from test enforcements. Files are placed by the first node of the dataset name. The UNIT= field is ignored. SYSDA or SYSDG both work under SMS. |
| E. Private | VSAM, PDS, and system type files. |

**NOTE** If pre-allocated, refer to the section *DASD Enforcement Standards* above for a description and general information.

## 6.2.2 General Information on DASD File Allocation

### 6.2.2.1 Process for Requesting DASD

Application areas or customers request DASD from Storage Administrators within ICT Host Operations. The Storage Administrators team assesses the capacity requirements, needs, and availability prior to approving a capacity increase to maintain the environment. Requests must be submitted via email, or a Service Request ticket, to CDC.OPS.STORAGE for processing. The various DASD request types are outlined below. Since there are no volume-level backups, files must be backed up before requests are made.

- DASD-ESTIMATES — Capacity planning requires early analysis of projected DASD requirements, as reserves may have to be increased before the DASD due date. Requests must be submitted 30 days prior to the required date to allow for ordering, shipping, and installation of additional DASD.

- DASDPROD — Required for production type requests like new production VSAM-, PDS-, IMS-, and DB2-type files. Requests must be submitted within one week of the DASD due date. For large requests in excess of 1200 cylinders (or one [1] GB), allow at least four weeks.
- DASDTEST — While test file size has not been mandated, reviews of abuse will be brought to the attention of management. Downsizing test files and correct blocking will improve the validation experience.

## 6.2.2.2 Approval Process

Approval must be obtained from the Storage manager in the following situations:

- To place any datasets on the storage SYSDA over 600 cylinders. (Files exist for cycle only, UNIT=SYSDA)
- To place any datasets on the storage SYSDG over 250 cylinders. (Files needed in another cycle, UNIT=SYSDG)
- Sortwork space over 60,000 cylinders. For more information on large sorts contact Storage.Admin.
- Temporary datasets over 1,000 cylinders (e.g., `DSN=&&dsname, UNIT=SYSDA`).
- Allocation of non-registered datasets on the private packs.

## 6.2.2.3 Retention

Do not specify a retention period on any DASD dataset.

## 6.2.2.4 Approval Process

Do not specify "VOL=SER=" for any files, in production JCL or PROC.

## 6.2.2.5 Generation Datasets

Generation datasets that reside on DASD should utilize the following table (Table 6-3) to determine the number of generations to be kept for a given file based on creation frequency. Abuse will be questioned on a case-by-case basis.

**Generation Datasets**

| Frequency | Number of Generations |
|---|---|
| Daily | Eight (8) |
| Weekly | Six (6) |
| Monthly (tape-only)* | Four (4) |
| Quarterly (tape-only)* | Five (5) |
| Intermediate work file | Four (4) |

**Table 6-3** Generation Datasets

*** Reminder: Non-access production DASD files have a 100-day lifespan. This may affect these DASD generations.**

## 6.2.2.6 SB37 ABENDS

To minimize "SB37" ABENDs occurring within the production cycles, these procedures should be followed for disk space allocations:

- Primary space should be allocated to handle 100% of the normal load, or 90% of the maximum load expected.
- Secondary allocations should not exceed 20% of the primary request. The secondary should only be used as a safety net.

Operations routes PRDSPOOL reports showing SB37 ABENDs. These can be used to maintain production systems by increasing the primary request in order to prevent repetitive ABENDs.

## 6.2.2.7 DASD Manager Recommendation

The DASD manager recommends that the best DISK block size is half-track blocking. See the example allocation below.

The MAX TAPE block size is 32,760. Current shop DASD is of 3390 types. The architecture is 56,664 bytes to a track, and there are 15 tracks to a cylinder. A 3390 model three (3) has 2.835 GBs.

An example allocation follows:

```
Given that a file has:

150,000 Records
LRECL=2075
DSORG=PS
RECFM=FB
Logic:
For half-track blocking take 27986;
divided by 2075 LRECL = 13.49
13 * 2075 = 32760 Blksize.
13 * 2 = 26 Records per track
26 * 15 = 390 REC/CYL
For 150,000 records that would be:
DSORG=PS,SPACE=(0,(11550,1155),RLSE),
or SPACE=(32760,(11550,1155),RLSE),
or SPACE=(TRK,(5775,578),RLSE),
or SPACE=(CYL,(385,39),RLSE)
```

**NOTE** DSORG=PS,BLKSIZE=0 is available; however, the user needs to be aware that in some applications like IMS log files, Easytrieve output and COBOL files outputting to a device, the BLKSIZE=LRECL. BLKSIZE=0 will automatically get the best blocksize no matter the device

architecture. Normally this is half-track blocking on DASD and 32k for tape. Watch for BLKSIZE=LRECL as this is very poor blocking and will drive up everything from I/O to storage used. Validate it before making this change.

# 6.3 Backup Information

## 6.3.1 Procedure Name — File Backups

It is the responsibility of Systems and Programming to back up their own files. Direct any questions to the Storage manager.

Each application owner is responsible for ensuring that databases or data stores in Production are receiving the appropriate levels of backup. The backup method and frequency must support the stated Recovery Time Objective (RTO) and Recovery Point Objective (RPO) in the Disaster Recovery Plan for that application.

When setting up a new Enterprise Server application, DR Manager is used to continuously track critical datasets required for recovery of an application. Reports will be generated on a daily, weekly, quarterly, or annual basis depending on how frequently the application systems run their report jobs. These reports will give the current recovery status of a given application's critical datasets. All applications being monitored by DR Manager must use the dataset listing generated by DR Manager to determine if all critical datasets are backed up and recoverable.

# 6.4 DASD Archiving

## 6.4.1 Procedure Name — DASD File Restoration

### 6.4.1.1 Blue Cross Blue Shield DASD File Restoration

DASD files will be archived as deemed necessary by the Storage manager. This process will be transparent to the file user. Archived files will be automatically restored to DASD when requested by the executing job. Files that are archived will be scratched according to the standards covered earlier. For details on these standards, refer to the section *DASD Enforcement Standards* above.

It should be noted that the archive index is purged each Sunday of all GDG entries that are not cataloged.

# 6.5 File Access Methods

## 6.5.1 Procedure Name — Access Methods

No new VSAM or TOTL files are allowed to be created for applications created by BlueCross. All new database structures must be either IMS/DB or DB2 for all internally developed applications.

All new files used by the online systems (e.g., CICS) must be IMS/DB or DB2 datasets.

Support and maintenance of existing VSAM files is allowed. For maintenance and support of VSAM files, all existing standards found in *Technical Standards — Applications* > *Business Systems Considerations*, *File Management Standards and Guidelines*, *File Design*, and *Other Considerations* apply.

For Alternate index VSAM files, refer to the section *VSAM Standards and Practices* below.

# 6.6 VSAM Standards and Practices

When defining any Non-IMS VSAM dataset, the following standards will be used:

- Use of the access method services password protection feature for datasets is strictly forbidden.
- The uses of "stepcat" or "jobcat dd" statements are strictly forbidden.
- All VSAM datasets will follow standard naming conventions for base clusters, using the following qualifiers to identify them as VSAM datasets:

    - For TRICARE production datasets — VS as the third qualifier.
    - For all other production datasets — VSAM as the high-level qualifier.
    - For test datasets — TESTVS as the high-level qualifier.

- Each component of a VSAM cluster will be named. Default names will not be used. Data and index components will be named by suffixing the cluster name with**.DATA** and **.INDEX** respectively.
- There will be only one dataset define per VSAMLIB member. This will eliminate the problem of keeping multiple defines in sync. VSAMLIB members can then be concatenated in JCL to define multiple datasets in a single IDCAM execution. Repros should not be concatenated to execute in the same step as the define. This will fail under systems managed storage causing a "repro output error" and give an IDCAMS return code of 12. Condition code checking should always be used to confirm IDCAMS processing. (Refer to *Procedures & Tools* > *Host Tools* > *Developer Tools* > *Host Application Analysis Tools* for additional information.)
- The owner option will always be used: owner(xxxxxxxx)

    - Where the xxxxxxxx for the data portion is the job that creates (defines) the file.
    - Where xxxxxxxx for the index portion is the VSAMLIB member name of the IDCAM define. In the case of non-indexed files, this will reside in the cluster portion of the define.

- Control interval size will be specified for the data and index components separately. The CI size for the index should be sufficiently large to contain a single key and a three-byte pointer for each control interval in one data control area. It should be noted here that line files perform better when the CISZ for the data and index components are not the same size. This makes them go to different buffer pools.
- The use of share options (3,3) for Non-IMS VSAM files is strongly discouraged. Files requiring shared update will be IMS DB due to the integrity features of IMS. You will need the approval of the Senior Director of Systems Support to use share options (3,3). You will also need approval of the VSAM coordinator to use the share option of (3,3). The IBM manual states that IBM cannot ensure the integrity to the data with share options (3,3).
- The use of alternate indexes is strongly discouraged. File structures requiring alternate indexes will be IMS databases. Consideration for an alternate index file should be submitted in writing to the Senior Directors of Systems Support. A write-up should explain in detail the use of the file and whether it will be updated or totally replaced. Upon completion of the review, the requestor will receive from the Senior Director of Systems Support written approval.
- VSAM repro control statements can be maintained on BC.NDVR.PROD1.CARDLIB. However, members exist on BC.NDVR.PROD1.VSAMLIB to cut down on the duplication. These are:

```
>BC.NDVR.PROD1.VSAMLIB(repro1)  <
01 input/output single repro load looks like this:


repro -

infile(sysut1) -
outfile(sysut2)


>BC.NDVR.PROD1.VSAMLIB(repro2)  <
02 input/output multi-repros for two to ten are like this:


repro ifile(indd1) ofile(outdd1)
repro ifile(indd2) ofile(outdd2)


>BC.NDVR.PROD1.VSAMLIB(repro5)   <05 input/output


>BC.NDVR.PROD1.VSAMLIB(repro6)   <03 input/output
>BC.NDVR.PROD1.VSAMLIB(repro6)   <04 input/output
>BC.NDVR.PROD1.VSAMLIB(repro6)   <05 input/output
>BC.NDVR.PROD1.VSAMLIB(repro6)   <06 input/output


>BC.NDVR.PROD1.VSAMLIB(repro7)   <07 input/output


>BC.NDVR.PROD1.VSAMLIB(repro8)   <08 input/output


>BC.NDVR.PROD1.VSAMLIB(repro9)   <09 input/output


>BC.NDVR.PROD1.VSAMLIB(repro10)  <10 input/output
```

- When creating VSAM repro statements, specify the dataset to be accessed via the "DD name" rather than the dataset name. Use of the DSN parameter results in VSAM generating a disposition of old for that dataset. This disposition remains in effect for the duration of the job.

Another reason for referring to the file by JCL "DD name" is that the filename shows up in the cross-reference report of dataset names.

- When using the VSAM verify facility, specify the dataset to be accessed via the DD name rather than the dataset name. Use of the DSN parameter results in VSAM generating a disposition of old for that dataset. This disposition remains in effect for the duration of the job.
- While constructing a DD necessitates card, use a DD name that does not tie up the dataset and points to the dataset allowing specific disposition of SHR.

Example of the wrong way:

```
//xxxxs010 EXEC PGM=IDCAMS
//*
//SYSPRINT   DD  SYSOUT=*
//SYSIN    DD *
Verify dataset(VSAM.XXXX.HISTORY)
/*
```

Example of right way:

```
//xxxxs010 EXEC PGM=IDCAMS
//*
//SYSPRINT   DD SYSOUT=*
//SYSUT1   DD DSN=VSAM.XXXX.HISTORY,DISP=SHR

//SYSIN    DD *
Verify file (SYSUT1)
/*
```

- To reduce the number of VSAMLIB members, it is requested that the following VSAMLIB members be used for verifies:
  - To verify one file, use VSAMLIB member VERIFYL1, where the DD name of the file is SYSUT1.
  - To verify two files, use VSAMLIB member VERIFY2, where the DD names will be SYSUT1 and SYSUT2.
  - To verify from three to up to a maximum of 10 files, use VSAMLIB member VERIFY#, where # indicates the number of files (e.g., VERIFY10).

- All IDCAMS control statements defining VSAM datasets will reside in the protected library below. The VSAM coordinator, located in OPA, will have the final responsibility for the production VSAM defines as they are stored and/or updated on:

  - BC.NDVR.PROD1.VSAMLIB
  - BC.NDVR.PROD1.DMERVSAM
  - BC. SYSTEMS.VSAMLIB (no additions here — old file)

- Maximum file allocation will be no more than the present number of records plus three-months' growth. This will require quarterly purges and reorganizations to limit the size of the files except where purge dates are federally mandated.

# 6.7 Tape Standards

## 6.7.1 Tape Management System

### 6.7.1.1 Procedure Discussion

Tapes are specified as an output medium by specifying the UNIT parameter in the JCL:

- UNIT=TAPEC for 36 track 3490 square tape (BlueCross' output tape medium standard)
- UNIT=TAPES for 18 track 3480 square tape (BlueCross' mailout tape medium)
- UNIT=TAPEJ (3592JA high density tape, encryption capabilities)

**Tape Setup Guidelines**

1. All tape datasets must have a block size that is the largest multiple of the file's record length and as close to 32K (+ or -) as possible.
2. Tape expiration dates are determined by specifying LABEL=RETPD= # of days or LABEL=EXPDT=JULIAN date in the JCL.
3. If LABEL=RETPD=# of days is used, the Tape Management System (TMS) adds the # of days specified to the tape creation date to determine the expiration date. For example, today is 2001/088 (3/30/01) and specified in the JCL as LABEL=RETPD=3. This means the tape will be scratched in three days.
4. If LABEL=EXPDT=56 is used, TMS takes the Julian date you specify as the date you want the tape scratched. In this case, the tape would expire on February the 25th.
5. If the RETPD and EXPDT sub-parameters are both omitted from the label parameter, then the tape is kept three days, the default retention period for permanent tapes.
6. Generation datasets on tape should use the following table (Table 6-4) to determine the number of generations to be kept for a given file based on creation frequency. The manager responsible must approve exceptions.

**Generation Datasets**

| Frequency | Number of Generations |
|---|---|
| Daily | Eight (8) |
| Weekly | Five (5) |
| Monthly | Five (5) |
| Quarterly | Four (4) |
| Intermediate Workfile | Three (3) |

**Table 6-4** Generation Datasets

7. These datasets should use LABEL=EXPDT=99000 to specify retention periods to use the catalog system.
8. Test tape files created in test jobs and not used in six months (186 days) are returned to scratch status by the Storage Management staff.

## 6.7.1.2 Special Keywords for TMS

The four special keywords listed below are used for TMS to signify the retention period of a tape and its data. In order for a tape with a special keyword to be scratched, the expiration date must be manually changed. This is not allowed without the director's approval. Requests for expiration date changes should be submitted on a TMS REQUEST FORM via email to CDC.OPS.STORAGE.

- LABEL=EXPDT=98000 indicates to TMS that this tape is from an external source. TMS processing is bypassed by writing a record to the audit dataset, which keys a record of all exceptional tapes. TMS does ask the operator to verify that the tape is from an external source.
- LABEL=EXPDT=99000 indicates that the dataset is under catalog control and to be scratched when it becomes uncatalogued from the system catalog.
- LABEL=EXPDT=99CCC indicates a tape which is to have cycle control. (CCC is the number of cycles desired and can be from one to 364.) The highest cycle governs only those tapes for the same dataset name with cycle control specified in their expiration date. One cycle of a dataset is defined as all volumes with the same dataset name and creation date. This ensures that reruns of a particular program do not result in the premature loss of copies of the dataset. This is not used for generation datasets.
- LABEL=EXPDT=99365 indicates a dataset totally protected by TMS. This is not allowed at BlueCross without approval from the area director.

    - DMS/OS DASD management uses the EXPDT=99365. The software package interfaces with TMS via archive index maintenance. Volumes are then released to the scratch pool automatically. These datasets follow the naming convention below:

      ```
      DSN=DMS.ARCHPRIM.DMSN.CXXXXYYY.TZZZZZZ
      DSN=DMS.ARCHCOPY.DMSN.CXXXXYYY.TZZZZZZ
      DSN=DMS.MERGPRI1.DMSN.CXXXXYYY.TZZZZZZ
      ```
      Where: XXXX is the creation year
      YYY is the creation Julian day
      ZZZZZZ is the creation time, hhmmss

**NOTE** The default does not apply to temporary datasets.

## 6.7.1.3 Temporary Datasets

1. Temporary Datasets are those created by the system because no dataset name was specified in the JCL and no RETPD/EXPDT parameter was used.
2. Temporary Datasets that have the first two characters of a dataset name being && and no RETPD/EXPDT parameter.
3. Temporary Datasets that have LABEL=RETPD=0 with or without a dataset name.
4. Temporary Datasets have DISP=(NEW, DELETE) and no RETPD/EXPDT parameter.
   These tapes are ready for immediate use after the job finishes.
   If a tape needs to be kept in the vault any length of time, this should be specified on your tape retention sheet.

## 6.7.1.4 Mailout Tape Guidelines

1. The default standard is to compact the output data for Mailout tapes. Any time we write a mailout tape, the decision must be made if the end user can read the square tape in compaction format. If the end user cannot read the compact format, the DCB sub-parameter must be used: TRTCH=NOCOMP
2. Do not code the TRTCH sub-parameter with DCB sub-parameters **KEYEN**, **MODE**, **PRTSP**, **STACK** or **DD** parameters **DDNAME**, **KEYEN**, **QNAME**.
3. Tapes needed for mailout or plan/net on a regular basis — daily, weekly or monthly must be updated in TMS.
4. An email request must be submitted with the dataset name and job that creates the tape and given to the tape librarian for the librarian to update in TMS. This is a one-time update, which enables all tapes to be retrieved and delivered to the correct area.
5. A report is run at 6:00 a.m. daily allowing all tapes created before this time to be retrieved and delivered to the correct area. All tapes created after 6:00 a.m. are processed the following morning.

## 6.7.1.5 Mailout Scratch Subpool

TMS allows you to create separate scratch subpools to fill specific needs. Computer operators have the need to identify mailout tapes based on media type. This allows the dataset to be created in the correct scratch pool.

When using this feature, you name the various subpools. This name appears in the mount message informing the operator which media type to mount.

## 6.7.1.6 Mailout Dataset Names

1. When creating any mailout DSN, the third node must specify the word **MAILOUTC** (e.g., BS.CHAP.MAILOUTC.XXXX.XXXX).
2. To control and track the mailing of data from our company to other companies, a table has been established to control this process. All current production jobs that create "mailout" files are listed in this table. For files that need to be mailed out and are not in the production table, the Software Developer can apply **MAILOUTC** (an indicator) to the third node such as shown in the example below.
   `BC.WORK.MAILOUTC`
   Most applications have been added to the table. If an application is not in the table, send a request to Tape.Library to verify.
3. Some systems rarely or never mail tapes out of house; the table does not include all system prefixes. Ultimately the ability to freely create any DSN for mailout purposes may require Software Developers to request via email any new production mailout DSNs to the tape library that need to be added to the table. Tech Support must install updates at open weekends, so communication about additions to the mailout table is essential. Mailouts incorrectly created on cartridge (because DSN is not on table) will be recreated on virtual tapes.
4. In the below example, any DSN that begins with BC.AMMS.MAILOUTC and was created by a job whose name began with AMMS would be assigned to subpool MAILOUT.TAPEC.
   DSN=BC.AMMS.MAILOUTC*, JOB AMMS*, MAILOUT.TAPEC
5. The description is composed of the subpool name and volser range.
   Scrpool=mailout.Tapec,range=990000-999999

6. In the above example, tapes in the volser range 996000 thru 999999 are in scratch pool MAILOUT.TAPEC. The subpool names can be up to 13 nonblank characters. The names and volser ranges initially suggested are:

   c. MAILOUT.TAPEC 990000–999999

7. The DSNs for mailouts need to be standardized. The initial suggestions are:
   BC.XXXX.MAILOUTC
   where XXXX is the four (4)-digit system code

   XXX.XXXX.@ML(for Trailblazer)
   where XXX.XXXX is the eight (8)-digit system code for Trailblazer.

   Those datasets whose names are specified will be added to the table. It must be remembered that this table will be searched for every non-specific tape mount request. UCC1 informs operations that this search will be a nit.

## 6.7.2 Tape Management Standards — External Tape Labels

In order to provide a more accurate means of identifying tapes used by I/S, it has become necessary to standardize the external labels that are prepared by I/S Operations. Printed labels are available from the tape library. Software Developers should notify the tape library when a label needs to be created, and the tape library will create the label and affix it to the tape volume.

## 6.7.3 Tape Drive Allocation Standards

Only three tape drives can be used by any one job step without written permission. In cases where more than three tape drives must be used, the Director of I/S Operations and the Director of I/S Systems must issue written authorization for the exception.

## 6.7.4 Multiple Datasets on Tape

The multiple dataset option on one volume should be used when several small files are to be written to tape.

## 6.7.5 Tape Vault Storage Information

The storage and vaulting process for rotating physical tapes is performed daily by a member of the CDC Operations staff, and exclusively applies to Non-Host tape rotation to the off-site vault:

- Vault tapes are pulled at 9:00 a.m., 1:00 p.m., and 3:00 p.m.
- Tapes are returned to the vault at 5:00 p.m. each day. If a tape is needed for a longer period of time, you must notify the tape librarian.
- Tapes designated as vault tapes must be approved by a manager to be kept out of the vault overnight.
- Tapes are pulled from the off-sight vault only with the approval of a manager or director.

The following steps should be taken when tapes are to be sent to an off-site storage vault:

1. All datasets to be vaulted need to be updated on the vault authorization table. All areas requesting datasets to be added to this table must first get approval from their management and then send their request to: **CDC.OPS.STORAGE**.
2. DR, VLT1, VLT, or VAULT must be in either the 3rd or 4th NODE of the dataset name.
3. The following steps should be taken when requesting tapes from an off-site tape storage vault:

    a. Check the status of the tape using the "TMS" lookup feature of TSO. The "TMS" lookup on TSO can be accessed by way of the "Storage Management Products" selection from the TSO main menu.
    b. Select "Tape Management" and then "Online Inquiry," where the password "TMCINQ" must be entered. This will bring you to the "Tape Inquiry" panel where the tape number may be entered.
    c. Check for the "out code" and "slot" number on the "VOLSER INQUIRY DISPLAY" screen. There are four out codes:

        i. LIBR — This tape is being held in the on-site tape library.
        ii. VLT1 — This tape is in the off-site Medicare vault.
        iii. VLT2 — This tape is in the off-site vault.
        iv. VMS — This tape is returning from the vault.

    d. Complete a Service Request through the TSC Self-Service or by sending an email to **CDC.OPS.STORAGE** for CDC or CDS.
    e. Submit the completed Service Request to the tape librarian. Tapes will be pulled at 9:00 a.m., 12:00 p.m., 3:00 p.m., and as required.

## 6.7.5.1 Vaulting Vendor Non-Host CD Procedure

The process for vaulting Vendor Non-Host Application CDs is conducted by the Operations area through email confirmation and a chain of custody process to ensure the integrity of all assets:

1. Notify the CDC Storage Team via an email to **CDC.OPS.STORAGE** for the Vendor Non-Host Application CD (CD) to be vaulted. The request must include the person responsible and two backup personnel, along with phone extensions, and a brief description of the CD contents.
2. The responsible individual will be required to hand deliver the CD to be vaulted to the 2nd floor of the Data Center. Once at the first set of glass doors going into the Data Center facility pick up the phone, which will automatically dial directly into the Data Center. The Operations team will then take receipt of the CD to be vaulted. Along with the CD, attach a printed copy of the email request.
3. The CDC Storage Team will be responsible for the placement and organization of the CDs located in the vault. When the CDs are vaulted, the CDC Storage Team will reply to the request with the location of the CDs.

## 6.7.5.2 Retrieving Vaulted Non-Host CD Procedure

The process for retrieving Vendor Non-Host Application CDs is conducted by the Operations area through email confirmation and a chain of custody process to ensure the integrity of all assets:

1. Notify the CDC Storage Team via an email to **CDC.OPS.STORAGE** of the vaulted CD required. The request must include the responsible person, a phone extension, a brief description of the contents, and the location of the CD provided by the CDC Storage Team at the time of vaulting.

2. The CDC Storage Team will then retrieve the CD from the vault and notify the requestor when it is available for pickup.
3. The requestor will be responsible for making arrangements to receive the requested CD from the CDC Storage Team.
4. Once the requestor has completed using the CD, this individual will be responsible for requesting that the CD be placed back in the vault. The individual can follow the guidelines for vaulting CDs.

# 6.8 RISC/6000 System

All development and validation for DirectTalk/6000 voice response applications must be done on the designated RISC/6000 **development** machine. No debug operations will be done on the production machine without authorization from the Voice Team.

The Voice Team is responsible for doing a weekly incremental backup of the DirectTalk/6000 directories on the RISC/6000 production system for voice response applications.

Tactical Services is responsible for doing a full system backup of the RISC/6000 production system once a month.

The two most recent weekly incremental backups and the two most recent monthly system backups will be stored in the vault.

# Chapter 7 File Design

## 7.1 Design Considerations

For all production operational files, an informational version of the production file is required to exist. This will prevent informational reporting activities from degrading production processes.

# 7.2 Enterprise Server – IMS

## 7.2.1 Programming Guidelines

### 7.2.1.1 File Documentation - IMS/DB2 File Design Documentation

File documentation describes the physical composition of a given computer file and in the case of IMS/DB database files, the logical composition of a given computer file is also provided.

#### Non-IMS/DB2 File Design Documentation

All system design manuals must provide file documentation for all primary files that are affected by a given development effort. A primary file is defined as a transaction file or any file which will be backed-up, retained either by retention period or generation, or will permanently reside on disk. Physical file documentation is created during the Design process by completing a dataset definition form (see forms section) for a given file. After a copy member has been created, physical documentation can be created using an automated documentation package (ref).

#### IMS/DB2 File Design Documentation

##### Logical File Design Documentation

All system design documents must include general logical IMS/DB2 database file documentation. This documentation consists of a hierarchical structure chart of the IMS/DB2 database file and a list of data elements that will comprise each segment identified in the structure chart. This documentation is created as a result of the combined efforts of the Application Development Team and the Database Administrator(s) assigned to a given project.

##### Physical File Design Documentation

All physical IMS/DB2 database file design documentation will be provided in the database design document (ref).

#### Non-IMS/DB2 Operational File Documentation

##### Logical File Documentation

Logical operational non-IMS/DB2 file documentation can be retrieved on an as needed basis from the Data Dictionary and that is where this type of documentation is maintained by the Database Administration area.

##### Physical File Documentation

Operational file documentation can be created on an as needed basis using an automated documentation package (ref).

## Operational Dataset Name Cross-Reference

Any Work Request Team member needing this information should submit a Service Request (search on "Development Center").

# 7.2.1.2 File Design Considerations - Database Files

The purpose of this section of the Information Systems Standards Manual is to provide an explanation of the policies and guidelines to be followed when designing Enterprise Server computer files.

## General - Applies to All Types of Files

All permanent master files will have file descriptions beginning with an 01 level. In the case of files used by CICS only or shared by CICS and batch, they must begin with 02. For AMB programs, all database, VSAM file layouts, and IMS segment layouts must begin with 03.

The file description will be defined to the most detailed element used in any program accessing the file. Due consideration must be given to editing (e.g., a payment field should be set up as a group level and defined in the elementary level. This allows the checking of the field for numeric or spaces and for performing arithmetic operations.) The file description must be checked and approved by the manager.

EXAMPLE:

```
04          MSTR-PAYMENT-X

04          MSTR-DATE OF BIRTH

08          MSTR-PAYMENT              PIC S9(4) V99

08          MSTR-DOB-MM               PIC 99

08          MSTR-DOB-DD               PIC 99

08          MSTR-DOB-CC               PIC 99

08          MSTR-DOB-YY               PIC 99
```

## Sequentially Processed Files

All master files or important transaction files will contain a header or trailer record with appropriate low or high value key. This record will contain a count of the logical records in the file plus a total of some

significant fields such as dollar amounts or an appropriate hash total. These totals will be checked and balanced by the program during each run. A failure to balance internally should abort the run.

These totals and program internal balancing processes are to be done in addition to UNITECH balancing where appropriate. Refer to *Technical Standards — Applications > Application Coding > RULEs Standards and Guidelines > Balancing and Reconciliation Processing Requirements* for further details.

# Database Files

## General Procedures

All files developed or converted to use online update will be developed using IMS or DB2 databases.

Any changes made to an IMS or DB2 database will be coordinated through Database Administration.

## Physical Organization

The physical organization and design of IMS and DB2 databases is the responsibility of Database Administration. The Software Developer will access databases in the same manner (DL/I calls or DB-CALLS or SQL exec calls) no matter what physical organization is used.

VSAM commands for defining clusters for IMS databases physically organized as Database Administration will maintain VSAM files.

## Database File Design Procedures

The development of new databases will always be a joint effort between Database Administration and the Software Designer. Under normal conditions, certain steps will be followed to collect data elements and develop a flexible database structure. Based on project constraints, deviations from steps may be required. Any deviations must have prior management approval. These steps are based on the assumption that a non-IMS automated system already exists. In any case, all documentation and dictionary entries will be completed no later than the Post Go-Live Phase of the project.

1. The Software Designer and the DBA assigned to the project will collect all INCLUDES and CICS screens or GUI interfaces for the current system.
2. All data elements collected will be placed in the Data Dictionary or if the element already exists in the dictionary, the element will be related to the new database. Also, the elements are assigned a COBOL or equivalent name (COBOL DD name on the dictionary) that will be used in developing the segment INCLUDE.
3. Any new data elements will be defined per Database Design Manual Considerations found in this chapter. Data elements no longer in use will be deleted.
4. Steps A-C will be repeated until all areas are in agreement that the collection process is complete and the user signs off on the data elements.
5. At this point, the logical design of the data begins. The Software Designer and DBA must develop if IMS or DB2, charts as a joint effort. The DBA and Software Designer will study the required accesses to the data and a logical design will be completed.
6. The DBA collects volume statistics related to transaction/GUI interface requests and data updates. Based on the information gathered in step E and the usage statistics, the DBA will design a physical database structure. This physical database design should be completed before any program design is begun.

7. The DBA and the Software Designer will develop the copybooks for each database segment. Once the copybooks are completed database the test DBD is generated, and the test database is loaded.
8. Any changes to the test database during the validation phase of the project will be made after consultation with the DBA and may involve repeating some of the above steps.
9. After validation is complete, the DBA should be notified of production implementation as specified in *Technical Standards — Infrastructure > Operating Systems > Production Update Schedule*.

### Modifications to Existing Production IMS Database Files

#### Major Modifications

Major modifications to existing databases are those which involve changes to the entire structure of the database. In this case the new structure should be assigned a new name and should be treated as a new database, following the procedures for new database development. This is to allow the old test database to remain unchanged for validation modifications of old programs concurrent with the development of new programs.

#### Minor Modifications

Minor changes to a database Copybook are adding/deleting a few fields from a segment without changing the physical length of the segment, adding a search field, or anything that would not necessarily affect all programs accessing the DB.

If required, the DBA All INCLUDE members will change the test DBD will be updated and/or the Copybook for IMS segments are RACF protected and can only be changed by the DBA. Any changes to program logic should follow procedures established in.

After validation is complete, the DBA should be notified of production implementation.

## 7.2.1.3 Backup and Recovery Policies for Production IMS and DB2 for z/OS Databases

All Production Operational IMS and DB2 mainframe databases will have a production backup and recovery process based on business requirements as defined by the supporting I/S application programming area. These backup and recovery processes will be defined in the Database Administration Disaster Recovery Plan. (DRP 800-501_DB2 and DRP 800-500_IMS)

All Production Informational DB2 mainframe databases will have backup and recovery processes based on business requirements as defined by the supporting I/S application programming area and the following criteria.

1. If the Informational Database is loaded with data provided by the application via flat files then the supporting I/S application area will maintain the data required to reload the Informational copy in the event that a recovery is required.

2. If the Informational Database is loaded from a file unloaded from an Operational Database then the recovery of the Informational would be to unload the Operational and load the unloaded file to the Informational.

3. If the Informational contains historic data not retained in the Operational, or is updated rather than simply loaded, then a backup process would then be required which would be the same as the policy for an Operational database.

4. Business requirements (such as a required recovery time SLA for instance) may require a backup and recovery process which would be the same as the policy for Operational databases. As a failsafe, criteria number 1 should be followed also for Informational Databases loaded with data provided by the application via flat files.

#### 7.2.1.4 Backup and Recovery Policies for Test IMS and DB2 for z/OS Databases

The DBA area will develop backup and recovery JCL for the IMS databases for all validation environments when the databases are initially created. This JCL is turned over to the application development team or System Support group responsible for the system. The application development team or System Support Group will schedule the backup jobs to run routinely. The recovery JCL will only recover a full copy of the test database with no forward recovery. No test IMS logs will be applied.

DB2 databases for validation environments are not backed up by the DBA area unless requested by the application development team or by the System Support group. Upon request, the DBA area will create the processes necessary for backup. If a recovery is necessary, the application development team or the SSG should request a recovery by the DBA. Only a full copy of the test DB2 database will be recovered. No test DB2 logs will be applied.

## 7.2.2 Design Considerations

### 7.2.2.1 IMS DBD Names

The purpose of this standard is to describe the naming format of IMS Database (DBD) names. The Database Administration Area assigns these names.

DBD Name Format:

1. Maximum length is eight characters

2. Format (Figure 7-1):

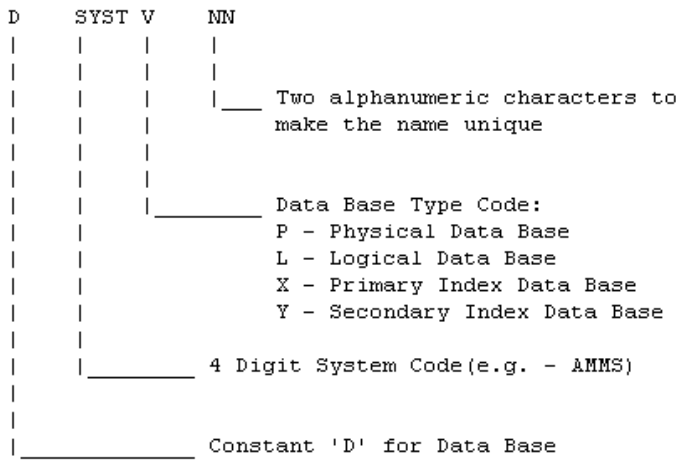



**Figure 7-1** DBD Format

## 7.2.2.2 IMS PCB Names

The purpose of this standard is to describe the name format for IMS/DB Program Control Blocks (PCB). The Database Administration Area assigns these names. The PCB name format is the same as the DBD name format. Refer to the section *IMS DBD Names* above.

## 7.2.2.3 IMS PSB Names

The purpose of this standard is to describe the name format for an IMS/DB "PSB". The Database Administration Area assigns these names. The PSB name format is the same as Program Name Format. For CICS programs, each CICS transaction will have an associated PSB that will be named in the following format (Figure 7-2):

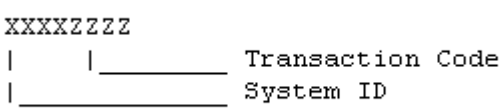



**Figure 7-2** IMS PSB Names

## 7.2.2.4 IMS Segment Names

The purpose of this standard is to describe the name format for IMS/DB "Segments". The Database Administration Area assigns these names.

Segment Name Format:

Maximum length is eight characters.

Format (Figure 7-3):

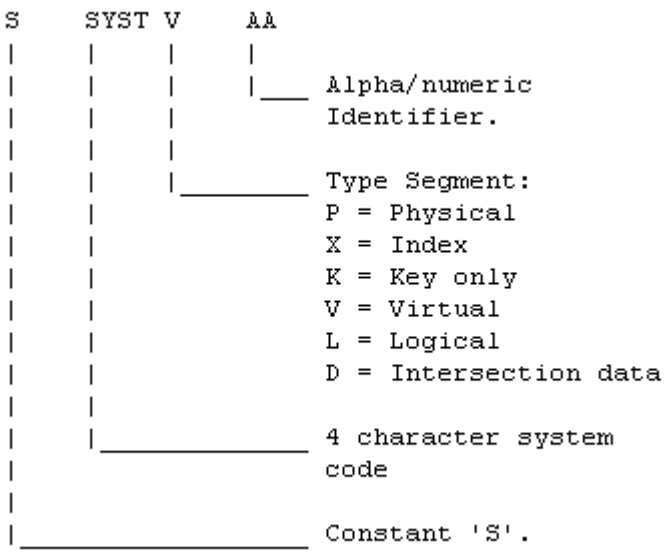



**Figure 7-3** Segment Name Format

# 7.2.3 Other Considerations or Guidelines

## 7.2.3.1 Database Design Manual Considerations

This section provides an explanation of IMS and DB2 database design documentation as a component of design and operational documentation. The IMS and DB2 database design documentation provides a basis on which database files are created.

This documentation also provides operational documentation of the logical and physical designs of production database files. The responsibility for the creation of this documentation rests with the DBA area, the application area responsible for the system needing the database file, and the CICS area when online processing is required for the file.

### Database Design Documentation

The following Data Dictionary components will be created or updated with the appropriate information to support a given development or maintenance effort. For a detailed description of DBA and Software Designer responsibilities for the Data Dictionary, refer *Procedures & Tools > Host Tools > Developer Tools > Application Interfaces & Utilities*.

- Data element definitions (DD)
- Record (segment) definitions (RD)
- File definitions (FD)

A series of relational or hierarchical charts will be created, which graphically describe the physical and logical structures of a given database file. For large database projects involving new databases, logical database design documents are produced by Accelerator (for IMS database structures), or another suitable data-modeling tool. Once PSBs and DBDs are created, these charts can be produced by an automated documentation package called IMS/MAP.

IMS segment Copybook documentation will be created using an automated documentation package (Refer to the subsection *Operational IMS Database Documentation* below.)

## Database Design Manual

- The following information will be included in a given database design manual for a given development or maintenance effort.
  Logical/physical hierarchical or relational charts
- For each database file:
  - FD documentation
  - RD documentation for each logical view
  - Data definitions
  - IMS segment INCLUDE documentation or DB2 table documentation
- Once the logical design of a database is completed, there is no effort to maintain the logical model. The logical model is used to develop a physical database, and all database documentation becomes operational database documentation.

## Operational IMS Database Documentation

All operational database file documentation is stored in the computer system and can be generated by requesting the computer system to produce the documentation. The following documentation is maintained (Table 7-1):

**Operational IMS Database Documentation**

| Documentation | Documentation Software |
|---|---|
| IMS file documentation | Data Dictionary (FD) |
| IMS segment documentation | Data Dictionary (RD) |
| IMS segment include document | DCDII |
| Data element definitions | Data Dictionary (DD) |
| Hierarchical charts | IMS/MAP |
| DB2 Database document | Data Dictionary (FD) |
| DB2 Table Documentation | Data Dictionary (RD) |
| Data element definitions | Data Dictionary (DD) |

**Operational IMS Database Documentation**

| Documentation | Documentation Software |
|---|---|
| DB2 Table Layouts/Referential (Available through DBA) | Integrity |

**Table 7-1** Operational IMS Database Documentation

## 7.2.4 IMS Database Validation - Batch and Online

There will be a maximum of four (4) tests IMS databases established for each production database. Database Administration will work with each area in developing comprehensive test databases and procedures required to refresh these databases. Once the test databases are turned over to the application area, it is their responsibility to do image copies and recoveries as required. Any modifications required to the JCL or delete/defines will be coordinated through Database Administration.

# 7.3 Enterprise Server – DB2

## 7.3.1 Programming Guidelines

### 7.3.1.1 DB2 Stored Procedures

All DB2 stored procedures must comply with these standards:

- Use of stored procedures requires approval from the DBA group and from the MQ Oversight Committee (this will be a cursory review if the user-defined MQ functions are not used).
- Stored procedures must be managed in ENDEVOR. The approvers for these processor groups will include the DBA area, the MQ Oversight Committee, and Tech Support to enforce the required approvals.
- Each stored procedure must be a native SQL stored procedure, not an external procedure.

DB2 stored procedures that are used for DBMS linking of any sort between test and Production subsystems must comply with these additional requirements:

- PGBA applications require a waiver due to Department of Defense regulations. This waiver requires the approval of the VP of ICT Host Operations and the VP or AVP of the application area developing the stored procedure.
- Versioning should be used to distinguish procedures that are used in multiple DB2 subsystems.

## 7.3.2 Design Considerations

Refer to the following sections for additional application guidelines:

- *Technical Standards — Applications > Application Coding > DB2 Design Considerations*
- *Technical Standards — Applications > Application Coding > DB2 Program/Query Design*
- *Technical Standards — Applications > Application Coding > Online Design Considerations*
- *Technical Standards — Applications > Application Coding > Online System Design Considerations*

### 7.3.2.1 DB2 Triggers

Use of DB2 triggers requires approval from the DBA group.

### 7.3.2.2 Writing to MQ

- If a DB2 stored procedure writes to MQ, it must use the DB2-supplied user function.
- If a DB2 trigger writes to MQ, it must use the DB2-supplied user function.
- Either process requires approval from the DBA group and from the MQ Oversight Committee.
- These are the only approved MQ write methods that do not use the MQ API.

### 7.3.2.3 Data Migration

If data migration from a test DB2 subsystem to Production DB2 must be executed in a direct real-time manner, the following considerations apply:

- DB2 stored procedures are the only approved mechanism for this. These procedures must comply with the standards in *Technical Standards — Applications > Application Coding > Online System Design Considerations*.
- The application is required to comply with the security and audit standards described in *Technical Standards — Applications > Application Coding > Online System Design Considerations > Data Migration*.
- Limitations on volume of data must be by approval on a case-by-case basis, based on performance tuning and load testing.

## 7.3.3 DB2 Validation and Performance Consideration

A task force that evaluated PIMS system performance after it went into Production developed the following general standards. These are presented only as a general checklist; other recommendations should be added as they evolve with other systems.

### 7.3.3.1 Design Review Checklist

**Design Review Checklist**

| Primary Responsibility | Tasks |
|---|---|
| Appl* | Explain Output for all SQL and review for potential problems. (Should reflect valid table statistical values). |
| Appl* | Estimated number of rows returned for each SQL statement. |
| Appl* | Expected number of rows to be sorted for each order by Call path for SQL statements. (How many times each SQL statement will be executed). |
| Appl* | Test Application traces for high exposure transactions. |
| Appl* | Data Volumes both immediate and future. |
| Appl* | Customer expectations by individual transaction. |

*The Applications Group collects the information; the DBA Group reviews and approves the information.

**Table 7-2** Design Review Checklist

## Post Production Performance Problem

**Post Production Performance Problem**

| Primary Responsibility | Tasks |
|---|---|
| ---------- | ----- |
| SQL Expert | Initial categorization of problem |

**Table 7-3** Post Production Performance Problem

### SQL Expert

As the query environment becomes more widespread, it becomes more important to have a SQL expert to provide detailed assistance to both the users and to the Software Developers as questions for code efficiencies arise. This person could be the initial contact for performance questions and should be proficient in application tuning.

## 7.3.3.2 Unknown or System-Related Problem

- Tech — Check Omegamon exceptions for problem indicator
- Tech — Check any abnormal error condition messages
- Tech — Check system statistics against model for abnormal conditions
- Tech — Use Omegamon to monitor the transaction as it runs
- Tech — Create application trace as needed

## 7.3.3.3 Problem Appears Application Related

- High CPU
  - Look at SORTS and JOINS
- High I/O
  - Look at access paths and DASD
- DBA and Application Area
  - Examine recent explain for potential problems
  - Compare to any previous explains for changes
  - Look at types of accesses — index vrs relational scan
  - Look at sort requirements — sorts instead of index usage
  - Look at types of indexes — clustered/unclustered
  - Look at match cols for high filtering
  - Determine if better indexing scheme is practical
  - Examine applications trace to determine time consuming function
  - Look at amount of data accessed
  - Look at amount of data sorted

- Look at number of times SQL calls are executed
- Look at List Prefetch usage
- Check related tables and indexes for reorg needs

- Tech
  - Check for DASD contention

Take corrective actions as determination is made.

## 7.3.3.4 Other Considerations or Guidelines

- All DB2 package binds are required to have EXPLAIN(YES).
- The Endevor processor group for package binds will automatically insert an EXPLAIN(YES) statement into all DB2 packages being bound through Endevor.

# 7.4 Enterprise Server – Sequential

Refer to *Technical Standards — Applications > Application Coding > AMB/COBOL.*

# 7.5 Enterprise Server – GTM

Refer to *Technical Standards — Applications > Application Coding > AMB/COBOL.*

# Chapter 8 Other Considerations

## 8.1 Corporate Mail Considerations

Prior to the completion of the Design, Development and Validation Phase of project development, new Production jobs and changes to the current production jobs must be tested through InfoPrint Workflow (IPW)/File Based Insertion (FBI) and approved by Print Operations and Corporate Mail Services. Corporate Mail Services must be involved in all phases of any system changes or development affecting documents mailed from Corporate Mail Services.

**NOTE** In order to allow sufficient time for the validation of any applicable deliverables, the Print Operations and Corporate Mail Services must be included in project plans from the start of a given project.

Finalist will not be used to obtain mail discounts. Instead, Finalist will be used by the I/S application area to validate, format, and store address information on legacy databases. The approved uses for the Finalist application can be found in *Procedures & Tools > Host Tools > Developer Tools > Application Interfaces & Utilities*.

# 8.2 Address Validation Standards

The following Address Validation standards apply to all new development after June of 2008 and must be retrofitted into any system or subsystem that is undergoing a major revision.

Standardization and validation of address information must be accomplished by using the third-party vendor product Finalist. Refer to the section *Corporate Mail Considerations* above for additional information related to utilizing the Finalist tool.

These standards are designed to meet the United States Postal Service requirements documented in the United States Postal Service (USPS) publication 28.

## 8.2.1 General

The Delivery Address Line and the Last Line of addresses output to the mail piece must be complete, standardized, and validated with the ZIP+4 file and City State file, respectively.

The definition of a complete address is one that has all the address elements necessary to allow an exact match with the current Postal Service ZIP+4 and City State files to obtain the finest level of ZIP+4 and delivery point codes for the delivery address.

A standardized address is one that is fully spelled out, or abbreviated by using the Postal Service standard abbreviations, or as shown in the current Postal Service ZIP+4 file.

It is required that delivery address information be stored in a minimum of 30 bytes or spaces in your computer systems. The optimum recommendation is 64 bytes or spaces to be compatible with the Postal Service National ZIP+4 databases.

**Example:**

| | | |
|---|---|---|
| ABC MOVERS | → | Recipient Line |
| 1500 E MAIN AVE STE 201 | → | Delivery Address Line |
| SPRINGFIELD VA 22162-1010 | → | Last Line |

### 8.2.1.1 Format

Format all lines of the address with a uniform left margin. Uppercase letters are preferred on all lines of the address block.

## 8.2.2 Secondary Address Unit Designators

Secondary address unit designators, such as APARTMENT or SUITE, are preferred to be printed on the mail piece for address locations containing secondary unit designators. The preferred location is at the end of the Delivery Address Line. The pound sign (#) must not be used as a secondary unit designator if the correct designation, such as APT or STE, is known or is shown in the ZIP+4 file.

**Example:**

1500 E MAIN AVE STE 201

### 8.2.2.1 Pound Sign (#)

If the designation, such as APT or STE is unknown the pound sign (#) can be used. If the pound sign is used, there must be a space between the pound sign and the secondary number.

**Example:**

425 FLOWER BLVD # 72

### 8.2.2.2 Alternate Location

If all Delivery Address Line information cannot be continued in the Delivery Address Line above the city, state, and ZIP Code, place secondary address information on the line immediately above the Delivery Address Line.

**Example:**

MR M MURRAY
APT C
5800 SPRINGFIELD GARDENS CIR
SPRINGFIELD VA 22162-1058

## 8.2.3 Attention Line

The Attention Line is placed above the Recipient Line, that is, above the name of the firm to which the mail piece is directed.

**Example:**

ATTN JOHN DOE
ABC COMPANY
1401 MAIN ST
FALLS CHURCH VA 22042-1441

## 8.2.4 Dual Addresses

Eliminate dual addresses on the output mail piece, if possible, although mailer files may maintain both mailing and physical addresses. However, if dual addresses are used, place the intended delivery address on the line immediately above the city, state, and ZIP+4 Code. This normally is the Post Office Box address. The other address must be placed on a separate line above the Delivery Address Line. The ZIP+4 Code used must be the correct code for the delivery address on the line directly above the city, state, and ZIP Code.

**Example:**

1201 BROAD ST E
PO BOX 1001
FALLS CHURCH VA 22062-1001

## 8.2.5 Last Line of the Address

### 8.2.5.1 Punctuation

With the exception of the hyphen in the ZIP+4 Code, punctuation must be omitted in the last line of address.

**Example:**

PLUMMERS LANDING KY 41081-1411

### 8.2.5.2 Format

Format the Last Line with at least one space between the city name, two-character state abbreviation, and ZIP+4 Code.

**Example:**

TAMPA FL 33630-9998

### 8.2.5.3 Spelling of City Names

Spell city names in their entirety.

**Example:**

Newberry Springs

### 8.2.5.4 Military Address

Overseas military addresses must contain the APO or FPO designation along with a two-character "state" abbreviation of AE, AP, or AA, and the ZIP Code or ZIP+4 Code.

**Example:**

APO AE 09001-5275

## 8.2.6 Delivery Address Line

The Delivery Address Line must be broken down into its individual components on the mailpiece with one space between address elements.

These components are the primary address number, predirectional, street name, suffix, postdirectional, secondary address identifier, and secondary address.

**Example:**

101 W MAIN ST S APT 12

### 8.2.6.1 Street Name

Punctuation must be limited to periods, slashes, and hyphens:

- Periods: 39.2 RD
- Slashes (fractional addresses): 101 1/2 MAIN ST
- Hyphens (hyphenated addresses): 289-01 MONTGOMERY AVE

## 8.2.7 Rural Route Addresses

### 8.2.7.1 Format

Print rural route addresses on mail pieces as: RR N BOX NN. Do not use the words RURAL, NUMBER, NO., or the pound sign (#).

**Example:**

RR 2 BOX 152

### 8.2.7.2 Leading Zero

A leading zero must not be included.

**Examples:**

RR 02 BOX 152
Incorrect

RR 2 BOX 152
Correct

### 8.2.7.3 Designations RFD and RD

Change the designations RFD and RD (as a meaning for rural or rural free delivery) to RR.

**Examples:**

RFD ROUTE 4 #87A
Incorrect

RR 4 BOX 87A

Correct

### 8.2.7.4 Additional Designations

There must be no additional designations, such as town for street names, on the Delivery Address Line of a rural route address. If an additional designation is used, place it above the Delivery Address Line.

## 8.2.8 Highway Contract Route Addresses

### 8.2.8.1 Format

Print highway contract route addresses on a mail piece as: HC N BOX NN. Do not use the words HIGHWAY CONTRACT, ROUTE NUMBER, NO., STAR ROUTE, or the pound sign (#).

**Examples:**

HIGHWAY CONTRACT ROUTE 68 BOX 23A
Incorrect

HC 68 BOX 23A
Correct

### 8.2.8.2 Leading Zero

A leading zero before the highway contract number must not be included.

**Examples:**

HC068 BOX 98D
Incorrect

HC 68 BOX 98D
Correct

### 8.2.8.3 Hyphens

Print hyphens as part of the box number.

**Example:**

HC 68 BOX 19-28

### 8.2.8.4 Additional Designations

There must be no additional designations, such as town for street names, on the Delivery Address Line of a rural route address. If an additional designation is used, place it above the Delivery Address Line.

# 8.2.9 General Delivery Addresses

## 8.2.9.1 Format

Use the word GENERAL DELIVERY, uppercase is preferred, spelled out (no abbreviation) as the Delivery Address on the mail piece.

**Example:**

MR JOHN ADAMS
GENERAL DELIVERY
TAMPA FL 33602-9999

# 8.2.10 Post Office Box Addresses

## 8.2.10.1 Format

Post Office Box addresses are output as PO BOX NN on the mailpiece.

**Examples:**

PO BOX 11890

PO BOX G

## 8.2.10.2 Designations

PO Box addresses often appear with the word CALLER, FIRM CALLER, BIN, LOCKBOX OR DRAWER. Change these to PO BOX as output on the mailpiece.

**Examples:**

ABC COMPANY
DRAWER L
Incorrect

ABC COMPANY
PO BOX L
Correct

# 8.3 Browser-Based Tools

## 8.3.1 Quickr Standards

### 8.3.1.1 Quickr Application

Quickr is a browser-based application used to publish, share and track information with a community of individuals. For collaborative projects for which Quickr is not suitable, contact Presentation Systems for a list of other collaborative tools.

Quickr is **ONLY** to be used for:

- Collaboration (e.g., document manipulation/viewing/commenting among registered users) Information sharing (e.g., uploading of files to a common location for viewing only)
- Quickr may **NOT** be used for:
- Workflow (e.g., hierarchical levels of editing/approving/routing of information)
- I/S Project Management (e.g., task-based activities for users, tracking)
- Projects that need immediate interaction (e.g., real-time communication and collaboration like virtual meetings)
- Highly structured methodologies (e.g., Joint Application Design, phase-based reporting)
- Client Management ensures that the customer receives an overview of Quickr, including the I/S standards for Quickr site creation and use. This will determine if Quickr is the best product to use to meet the business need. If it is determined that Quickr is the application tool to use, these requirements must be met:

    - Use a standard template that contains only these elements:
    - Corporate logo, to be located in the upper left of the Quickr site
    - Welcome (a page that describes the purpose of the Quickr site)
    - Library (a folder that stores pages of your choice)
    - Index (a folder that stores all pages in the Quickr site)
    - Manager Tools (a room where managers can customize certain aspects of the Quickr site and manage members)
    - Technical Help (a page that contains technical contact information)
    - An area for external members to change their passwords or profile information
    - Once the template is created, managers and authors of a Quickr site are required to attend technical training.

- Roles:

| | |
|---|---|
| Managers | Have total control of a Quickr site. They can read, author, set security, decorate and make custom forms. External users will not be authorized for this role. |
| Authors | Have the ability to both read and create new pages within a Quickr site. External users will be allowed to be authors. Technical training, however, will be required. |
| Readers | Have read only access to a Quickr site. External users are authorized for this role with no training required. |

- Naming conventions must be established for each Quickr site. Pages within a Quickr site will use "TopicCategory_SpecificSubject" as the standard.
- File type considerations:

  - No video or audio files are allowed.
  - For any file type that is attached, all members of the Quickr site must have the software or necessary viewer in order to access the file. Otherwise, the file must be converted to some other format (e.g., Microsoft Word, Visio, Microsoft Excel).

## 8.3.1.2 Quickr Site Structure

1. A customer must request Quickr via a general TSC Self-Service Request, which is forwarded to Client Management.
2. The customer provides the business requirements for the Quickr site, along with the list of Quickr site managers, to Client Management. At the direction of Client Management, the Presentation Systems Website Systems and Systems Support teams will create the Quickr site and add the initial group of managers.
3. Presentation Systems using the standard I/S Work Request process will handle customization that requires programming.
4. Administrators of all Quickr sites will be the Presentation Systems Website Systems and Systems Support teams.
5. The standardized Quickr site template is initially set up by Presentation Systems and includes the components shown in Figure 8-1 Quickr Site.

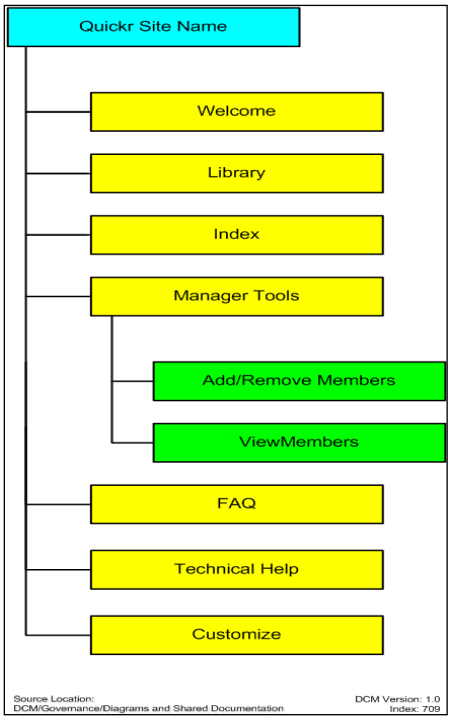



**Figure 8-1** Quickr Site

6. A Managers Tools room, accessible by all managers of the Quickr site. In this room will be an inner room with Manager FAQs.
7. Presentation Systems Website Systems and Systems Support teams will be automatically listed as managers of every Quickr site.
8. A Line of Business logo will be shown in the top right corner of every Quickr site.
9. Quickr site managers will be able to create rooms. The Presentation Systems Website Systems and Systems Support teams, however, must be added as managers of every room created.
10. The Quickr site calendar may be made visible to members at the discretion of the Quickr site managers; the Quickr site calendar, however, will not be synchronized with the Outlook calendar.

## 8.3.1.3 Infrastructure/Application

Presentation Systems is the owner of the Quickr software.

### Storage Requirements

The production Quickr servers are currently backed up daily and held for six (6) revisions until I/S Network Services revises or improves the back-up process.

### Capacity Plan

- Each production server is 98 Gigabytes.
- Individual Quickr sites will store no more than 300 MB.
- Individual documents will not be larger than 12 MB.
- All initial customization will be done in a test environment in order to minimize production problems.

Please refer to the Quickr System Documentation Manual for more information on infrastructure.

### Security

Virus scanning is enabled for all the production Quickr servers.

### Purge Criteria

Documents older than 6 months that have not been read or modified will be permanently purged. The purge process is dynamic and sends notification to document owners 1 week in advance.

### Authentication/Authorization

Users register themselves in a Quickr registration database, choosing their own usernames and passwords. Users may have the same ID and password for every Quickr site to which they have access.

### Encryption

Encryption for Quickr is an all or nothing feature. Therefore, for security reasons, encryption will always be turned on. Also, port 443 is the only port open for Quickr use.

## Authorization Revocation

A manager of a Quickr site can remove members, as can the Presentation Systems Website Systems and Systems Support teams.

## Pages/Attachments

Versions of pages that are older than 6 months must be deleted (Refer to the subsection *Purge Criteria* above.).

# 8.4 SharePoint Online Content Storage

The application software for ***SharePoint Online (SPO)*** is Microsoft SharePoint. Vendor-related standards refer to those set forth by Microsoft for any given version of SharePoint.

## 8.4.1 SPO Standards for Work Request Life Cycle Documentation

### 8.4.1.1 File Naming

#### Documentation Originating in I/S

The following format is recommended for documentation that originates in I/S.

*<work request ID>_<Document type>_<optional descriptor><.ext>*

**<Work Request ID>**

For projects use the project code. This is the ***WRMS*** project code, not the ***ETKS*** "P-number." For Change Sheets, use the Change Sheet number.

**<Documentation Type>**

The type of documentation as it relates to a Work Request. Table 8-1 is a list of example values. It is important to maintain consistency in the use of document types.

**Work Request Documentation**

| List of Example Values | | |
|---|---|---|
| WDD | Context | Issues |
| HLE | Concept | Agenda |
| Discovery and Delivery Strategy | Bridge | Recap |
| Design | Project Plan | Requirements |
| Implementation/Go-Live | Test Matrix | CC (Change Control) |
| Post Go-Live | Test Plan | CCnn (nn = Change Control Number) |

**Table 8-1** Work Request Documentation

**<ext>**

This value is automatically assigned by the application used to create the document.

Refer to Table 8-2 below for examples of file naming conventions.

**Examples of File Naming Conventions**

| Work Request Number | Documentation Type | Description (optional) | Ext | Example File Names |
|---|---|---|---|---|
| GS1070 | Agenda | 20080123 (YYYYMMDD) | .docx | GS1070_Agenda_20080123.docx |
| GS1070 | Design | Human Resources | .docx | GS1070_Design_Human_Resources.docx |
| | | | | GS1070_Design_HR.docx |
| GS1070 | CC01 | Add deferral | .docx | GS1070_CC01.docx |
| | | | | GS1070_CC01_Add_deferral.docx |
| GS1070 | Requirements | AMMS rewrite | .xlsx | GS1070_Rqmts.xlsx |
| GS1070 | Email | Approval | .txt | GS1070_CC01_Approval_e-mail.txt |

**Table 8-2** Examples of File Naming Conventions

## Documentation Originating Outside of I/S

### Official Documentation

Documents originating outside of I/S can retain their original file name. Examples of documentation with external origin include TMA policy changes, standards, and contracts.

### Customer Correspondence

A brief description to clarify the documentation's purpose includes the date when there is a need to identify multiple documents of the same type (e.g., CC01 approval 20080112.txt).

# 8.4.1.2 Control

All Work Request documentation for a given Work Request is stored, versioned, and maintained within its Work Request site.

Work Request sites are automatically generated by an interface with the ***Work Request Management System***. A site Owner is assigned when the site is created. ***SPO*** Administrators or ***SPO*** Champions are responsible for adhoc Work Request site creation.

Access to the Work Request site is managed by the site Owner — the person with primary responsibility for a Work Request at any point within the Work Request Process Workflow. Refer to the section *Security* below for more information on the levels of access that may be granted.

## Declaring Records

When an ***SPO*** site is unused for a declared period of time, it may be associated with a retention/purge policy that manages the Work Request life span within ***SPO***.

## Retention/Purge Policies

The term **Purge/Delete** applies to retention and purge standards for ***SPO*** and means that documentation is physically removed from ***SPO*** and cannot be recovered..

The ***SPO*** retention and purge policies are based on contractual or legal requirements. Retention and purge policies are further defined by the Project Management Office (PMO) by line of business and can differ by PMO.

---

**NOTE** Retention and purge policies only apply to ***SPO*** Work Request sites that are NOT under a legal hold.

---

Work Request sites are purged and deleted from a ***SharePoint On-Premises*** Archive when the following conditions are true:

- The Work Request site is not on legal hold
- The Work Request site closure date is greater than 3 years

The data owner must indicate in the Closed Work Request list which work request sites are to be excluded from the Purge/Delete process.

When the Purge/Delete process has been completed for a specific Work Request site, all associated documentation cannot be recovered.

# 8.4.1.3 Security

The table below (Table 8-3) details the custom access for Work Request Life Cycle Documentation.

**R = Read**
**W = Write**
**D = Delete**
**Restricted Read = Read rights limited to specific documents or folders**

**Custom Access for Work Request Life Cycle Documentation**

| Assigned Role | Work Request Management | Task Management | Life Cycle | System Doc | Other | Customer Documentation | Manage Access |
|---|---|---|---|---|---|---|---|
| Site Owner | R, W, D | R, W, D | R, W, D | R, W, D | R, W, D | R, W, D | Yes |
| Member | R, W, D | R, W, D | R, W, D | R, W, D | R, W, D | R, W, D | No |
| Viewer | R | R | R | R | R | R | No |

**Table 8-3** Custom Access for Work Request Life Cycle Documentation

### 8.4.1.4 Search

Search results are security-trimmed and only return items where a user has access rights to the document.

# Chapter 9 Standards for Web-based Applications

## 9.1 Web Single Sign-on Standards

### 9.1.1 Single Sign-on Technology

BlueCross has adopted the Security Assertion Markup Language **(SAML) 2.0** specification to support single sign-on to and from partner websites. Within the SAML specification, BlueCross has adopted the **Web Browser SSO Profile** and **POST Binding**.

SAML is an open standard, enabling Web-based authentication and authorization. This standard provides specific support for single sign-on scenarios, which prompts users to log in once to access multiple Web-based software systems.

### 9.1.2 Confidentiality

Encryption protects the confidentiality of data during transmission.

All messages containing sensitive information such as PHI, PII, financial or HR information (e.g., Social Security number) MUST use transport-layer encryption. That is, such messages MUST be transmitted via HTTPS URLs both on the BlueCross application system as well as the partner or target application. No SAML messages containing sensitive information should ever be transmitted to or from the browser over unencrypted HTTP.

Additionally, all messages SHOULD use message-layer encryption. Confidentiality risk exists with the SAML 2.0 Web Browser SSO Profile. Using HTTPS exclusively could still allow the user to unknowingly pass data to an insecure third-party (e.g., via "man-in-the-middle" style attack). Message-layer encryption protects against this risk. If a partner site cannot technologically accommodate message-layer encryption and the agreed-upon message format contains no PHI, only then is it acceptable to forego message-layer encryption. Validation scenarios — in particular, early development and integration of an SSO interchange with a new partner site — may refrain from using message-layer encryption to ease troubleshooting.

### 9.1.3 Data Integrity/Non-repudiation

Digital signatures protect the integrity of the message during transmission. If a message is edited during transmission, the digital signature will be invalidated. Digital signatures also assure non-repudiation. Because the digital signature can only be produced by the sender, the sender cannot deny having sent the message.

Digital signatures MUST be applied to the following fields:

- All data related to the sender's identity
- All Personally Identifiable Information that identifies the user
- Creation timestamps

# 9.1.4 Message Expiration

Message expiration protects the system from replay attacks. Message lifetimes are enforced by the receiving system. To prevent messages with long periods of validity, the BlueCross system will derive the message expiration date from the creation timestamp in the message, even though the SAML specification allows the sender to pass an expiration date in the message itself. This maximizes system control over expiration.

All messages MUST contain creation and expiration timestamps.

- Expiration of messages MUST be derived from the creation time.
- The internally derived message expiration SHOULD have a short, but reasonable expiration, such as 60 seconds.
- Expiration timestamps generated by partners SHOULD be ignored.

# Chapter 10 Print Operations

This chapter contains standards that are applicable to the Print Operations Center.

# 10.1 ICT Print Operations and Corporate Mail Services

## 10.1.1 AFP Resource Change Control Process

Any new or existing Advanced Function Presentation (AFP) resource requiring creation or change, whether it is a PageDef, FormDef, Overlay, or PSEG, must be requested using the **Print Operations – Modify Print Resources** form via the TSC Self-Service. This form must be filled out in its entirety, and clearly stating all required information. If the form is incomplete, the request will be rejected and a new form requested with the missing required fields completed, which will result in a delay in the change control process.

Upon completion of an Overlay or PSEG, a signature is required from the requesters and from Corporate Communications prior to the resources being compiled to any production or test library. Once a resource has been validated and compiled to Production, additional first-run production verification is required as stated in the following ISSM sections:

- Refer to *Technical Standards — Applications > Application Development Support Tools > Package Naming Conventions for Endevor > Non Production Maintenance Packages.*
- Refer to the section *StreamWeaver JES Monitor Procedures* below.
- Refer to the section *Print Priority Procedures* below.
- Refer to the sections *InfoPrint Workflow Validation Procedure* and *InfoPrint Workflow Add/Delete Request Procedure* below.

## 10.1.2 StreamWeaver JES Monitor Procedures

The StreamWeaver Job Entry Subsystem (JES) Monitor is a started task that monitors the JES queue destination for PRT9900 and PRT9901 in Production and STRW1 in Test. All jobs received in these destinations are checked against the StreamWeaver JES Job Table to determine if the secondary StreamWeaver job should be executed.

This StreamWeaver Job Table contains the following fields that are specified in the I/S application area's job:

- JOBNAME
- SYSOUT
- FORMS

ICT Print Operations will need to be notified with what the Production, rerun and Test job names will be for all output using StreamWeaver. ICT Print Operations will then update the StreamWeaver Job Table with the required information.

If an output job is received in one of the PRT destinations and the job fields match an entry on the StreamWeaver Job Table, then the output job executes the StreamWeaver secondary job process.

If an output job is received in one of the PRT destinations and it doesn't match an entry on the StreamWeaver Job Table, then it will pass through to InfoPrint Workflow (IPW) with no execution of the secondary StreamWeaver job process.

## 10.1.3 Print Priority Procedures

If a print job needs to be given priority, ICT Print Operations will need to be notified via a **Print Operations – Priority Print** form located on the TSC Self-Service. Instructions for filling out the request can be found on the TSC Self-Service.

## 10.1.4 INFOPAC Procedures

All change requests for INFOPAC reports must be submitted through the TSC Self-Service. NO free-form emails will be accepted for changes.

### 10.1.4.1 INFOPAC Add Requests

**The INFOPAC Add request is to ADD a NEW report, NOT to add users to existing reports.** In order to define a new report to INFOPAC, an **INFOPAC Add** request form will need to be submitted via the TSC Self-Service. Instructions for filling out the request can be found on the TSC Self-Service. Any requests to add users using this form will be rejected.

### 10.1.4.2 INFOPAC Update Requests

**The INFOPAC Update request is for ANY changes to existing reports including adding user access.** In order to update an existing report on INFOPAC, an **INFOPAC Update** request form will need to be submitted via the TSC Self-Service. Instructions for filling out the request can be found on the TSC Self-Service.

This request is secured and can only be submitted by mangers and above.

### 10.1.4.3 INFOPAC Print Validation

INFOPAC reports that are to be printed as test jobs need to be routed to a corresponding test SYSOUT class. INFOPAC will do periodic sweeps of the test SYSOUT classes, and the resulting output will have a z/OS job name of *DCTL--$*. The DCTL--$ job will force the test Overlay at print time. An example of this would be a Software Developer wanting to validate a hardcopy of report XXXXXXXX. Instead of routing the report to the Production SYSOUT class of J, the report should be routed to **F**, which will be picked up by the test INFOPAC sweep job. Resulting TEST INFOPAC output will carry the job name of DCTL--$, which will then be printed with the test Overlay.

For each of the INFOPAC Production databases there is an equivalent test SYSOUT class (Table 10-1).

**Production and Test SYSOUT Classes**

| | Production | Test |
|---|---|---|
| INFOPAC | J | F |
| MCINFOPK | O | 9 |

**Table 10-1** Production and Test SYSOUT Classes

If an INFOPAC test print is required to be inserted, then a **Print Operations - Willowby IPW Test** form must be submitted via the TSC Self-Service.

## 10.1.5 InfoPrint Workflow Validation Procedure

InfoPrint Workflow (IPW) test print requests are to be submitted to the IPW.ADMIN, INSERTER.OPERATIONS and PRINT.AFP email accounts using the **Print Operations - Willowby IPW Test** form via the TSC Self-Service. IPW requests must include:

- Production job name.
- Test job name.
- SYSOUT class.
- Unique Identifier from the Automated Document Generation System (ADGS).
- PageDef (Forms Control Block (FCB)).
- Proclib used.
- INFOPAC report ID being validated.
- Mail piece counts of test work.
- Any other instruction pertinent to printing, additional processing and/or routing of these test materials.

Instructions must be specific to ICT Print Operations and must use pink paper, form 2440, whether you are validating manual or automated inserter mail. All test material that is to be either manually or automatically inserter validated must be preceded by a **Print Operations - Willowby IPW Test** form.

IPW ADMIN will check to see if validation is for a new or an existing job in IPW and submit a Banner Page Update Request to Inserter Operations (copying IPW ADMIN, Data Control, and Print AFP). The response from Inserter Operations will be one of the following statements:

- No changes needed for existing jobs.
- Listed the attribute changes for existing jobs.
- Provided the attributes needed for configuring and validating new jobs to IPW.

If changes are needed, the form will be sent back to be completed with attribute changes requested.

Once Inserter validation has been completed, Inserter Operations will convey the results back to the requestor via the **Inserter — Test Results** form. Test output must be provided a minimum of two weeks prior to the Production print phase.

## 10.1.6 Inserter/Manual Mail Validation Procedure

Inserter/Manual Mail test requests are to be submitted to the IPW.ADMIN, PRINT.AFP and INSERTER.OPERATIONS email accounts using the **Print Operations – Willowby IPW Test** form via the TSC Self-Service.

Inserter/Manual Mail test requests must have all the required fields on the request completed. Instructions must be specific to Corporate Mail Services and all print must be on pink paper (form 2440) whether you are validating manual or automated inserter mail. All test material, once printed, will be scheduled (in Corporate Mail Services) and processed. Results for all validation will be reported back to the original sender and any designees on an **Inserter - Test Results** email form.

# 10.1.7 ID Card Creation and Change Procedures

## 10.1.7.1 Procedures for New/Existing Membership ID Cards

These procedures are for the creation of new and changes to existing membership ID Cards. This action will occur when new groups are added to the membership system or when groups request changes to be made to existing ID card types. The ID Card Production area and the Print AFP area both require 72 hours per card to complete these actions. Time needed may be reduced if the cards requested are similar. Procedures for creating and changing cards are as follows:

1. The customer will need to submit a request to add or change an ID card using the TSC Self-Service form **ID Card & ID Card Carrier - Add/Change** found under Printing/Reporting Services. They should provide an attached layout of how the card should look as well as identify what card type it is to ICT Print Operations. The layout must use the following standards:

    a. If there is a business logo, it will be the responsibility of the customer area to provide that. Some logos will need to be altered to print in our format (600 DPI, black and white or color, JPEG or PNG format, and sized to fit an ID card 1" width by ¾" height).
    b. The layout will contain all variable fields needed for the card type in GREEN.
    c. The layout will contain all hard-coded fields for the card type in BLUE.
    d. All Changes to the existing card will be in RED.

---

**NOTE** When sending a layout, customers are reminded to indicate what data is constant and what data is variable. Constant data is data such as the client's name, logo, etc., which appears on every ID Card and is hard coded. Variable data is data such as the patient's name, ID number, employer group name and is contained in the file.

---

2. ICT Print Operations will add the assigned card type to Option A in SEZI. This will allow the customers to assign groups to the card.
3. ICT Print Operations will create the AFP Overlays in the test PDS along with Overlay numbers. ICT Print Operations will place the Overlays in the proper mainframe libraries. ICT Print Operations will produce a PDF mock-up based on the layout. The customer and Corporate Communications need to approve the PDF mock-up.
4. Once the PDF mock-up has been approved, Options B, C and D in SEZI will be updated to reflect the business need. Additional details on this update process are as follows:

    a. Option B assigns to a file Line of Business (LOB) and lists the images used and the start/term date of the images.
    b. Option C activates a field.
    c. Option D allows the override of nonstandard tags.
    d. There is a list of available fields for override. If a needed field is not on the list, it will require an update by Programming to the MEMB_GENERIC_REF table. This will require a Child Ticket to be created by ID Card Production and forwarded to I/S Membership for completion. The two types of fields are listed below:

i. For taking an existing field and making it available for override, you will need to go to the MEMB_GENERIC_REF table with a REF TYPE of ID CARD TAG. Verify that the REF_CODE_DESC_SHRT field is blank and that it exists.
ii. For adding a new variable and override, you first create a new REF TYPE = to IDCARDVAR on the MEMB_GENERIC_REF table. The CARD TAG NAME should be under REF CODE as ID CARD TAG. The override should also have a significant, LONG DESCRIPTION field. The REF_CODE field needs a 10-character TAG NAME.

5. ICT Print Operations will move the AFP Overlay with the test watermark to the test libraries for validation.
6. ICT Print Operations will need to create/update the FormDef or PageDef in order to accommodate for printing the Overlay and provide the number to AFP Programming for use in their applications.
7. Upon validation approval and receipt of all approvals, ICT Print Operations will need to be given the go-ahead to remove the test watermark and move the AFP Overlay to the production libraries and production PDS.
8. The first production print will need to be reviewed by the customer before mailing.

# 10.1.8 ID Card Carrier (Jacket) Creation and Change Procedures

## 10.1.8.1 Procedure for New/Existing Carriers

These procedures are for the creation of new and changes to existing ID Card Carriers. This action will occur when new groups are added to the membership system or when groups request changes to be made to existing ID Card Carriers. The ID Card Production area and the Print AFP area both require two (2) business days for the first carrier and one (1) business day for each carrier thereafter. This time frame could change for more complex changes or mass amounts of carriers.

1. The customer will need to submit a request to add or change an ID Card Carrier using the TSC Self-Service ticket form **ID Card & ID Card Carrier – Add/Change** found under Printing/Reporting Services and attach a layout of how the carrier should look to ICT Print Operations. ID Card Carriers will be assigned by individual card type based on the customer need. Requirements for ID Card Carriers are as follows:

   a. If there is a business logo, it will be the responsibility of the customer area to provide that.
   b. Some logos will need to be altered to print in our format.
   c. The layout will contain all variable fields needed for the carrier in GREEN.
   d. The layout will contain all hard-coded fields needed for the carrier in a BLUE font.
   e. The customer will need to provide a return address.

2. ICT Print Operations will create the AFP Overlay in the test PDS along with the Overlay number. ICT Print Operations will place the Overlay in the proper mainframe libraries. ICT Print Operations will then create a PDF mock-up. The customer and Corporate Communications will then need to approve the mock-up.
3. ICT Print Operations will move the AFP Overlay with the test watermark to the test libraries for validation.
4. ICT Print Operations will need to create/update the FormDef or PageDef in order to accommodate for printing the Overlay and provide the number to AFP Programming for use in their applications.

5. Upon validation approval and receipt of all approvals, ICT Print Operations will need to be given the go-ahead to remove the test watermark and move the AFP Overlay to the production libraries and production PDS.
6. The first production print will need to be reviewed by the customer before mailing.

## 10.1.9 Existing Production Mailables

IPW Admin must be notified via IPW.ADMIN at least two weeks in advance of any changes to the data format of existing production output. Requests must be submitted on the **Print Operations - Willowby IPW Test** form. This is to allow for changes to the control file, bar code placement, printing and passing to Inserter Operations for insertion validation. The email request will also need to contain the following information:

- Test Job Name
- Production Job Name
- Sysout Class
- PageDef (FCB)
- Mail piece Count
- INFOPAC Report ID (if applicable)
- Unique Identifier from the Automated Document Generation System (ADGS)

The email form and the response from the IPW Admin staff must precede the running of any test job that produces output going to IPW. As part of the email request, the IPW Admin staff will need sample data. This needs to be in the form of held output on the Enterprise Server that uses the alternate job name logic for test jobs (Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > Print Operations Standards* for additional information.). Once the IPW Admin staff has configured the job to IPW, the job will then be released to IPW for printing.

## 10.1.10 InfoPrint Workflow Add/Delete Request Procedure

An individual working with a particular Line of Business will be added to the InfoPrint Workflow (IPW) system with an original user ID and a password that the individual will decide upon. In order to be granted access to IPW, the individual's manager will need to complete the **IPW Access** request form on the TSC Self-Service. No verbal access/removal requests will be accepted.

Access will only be granted for the appropriate Line of Business for that individual.