Cognizant
Passion for making a difference

Coding Standards: COBOL

Version: COBOL/Coding Standards/0408/1.0
Date: 25-04-08

Cognizant
500 Glen Pointe Center West
Teaneck, NJ 07666
Ph: 201-801-0233
www.cognizant.com

TABLE OF CONTENTS

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Cognizant
Passion for making a difference

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

1. Scope of the Document

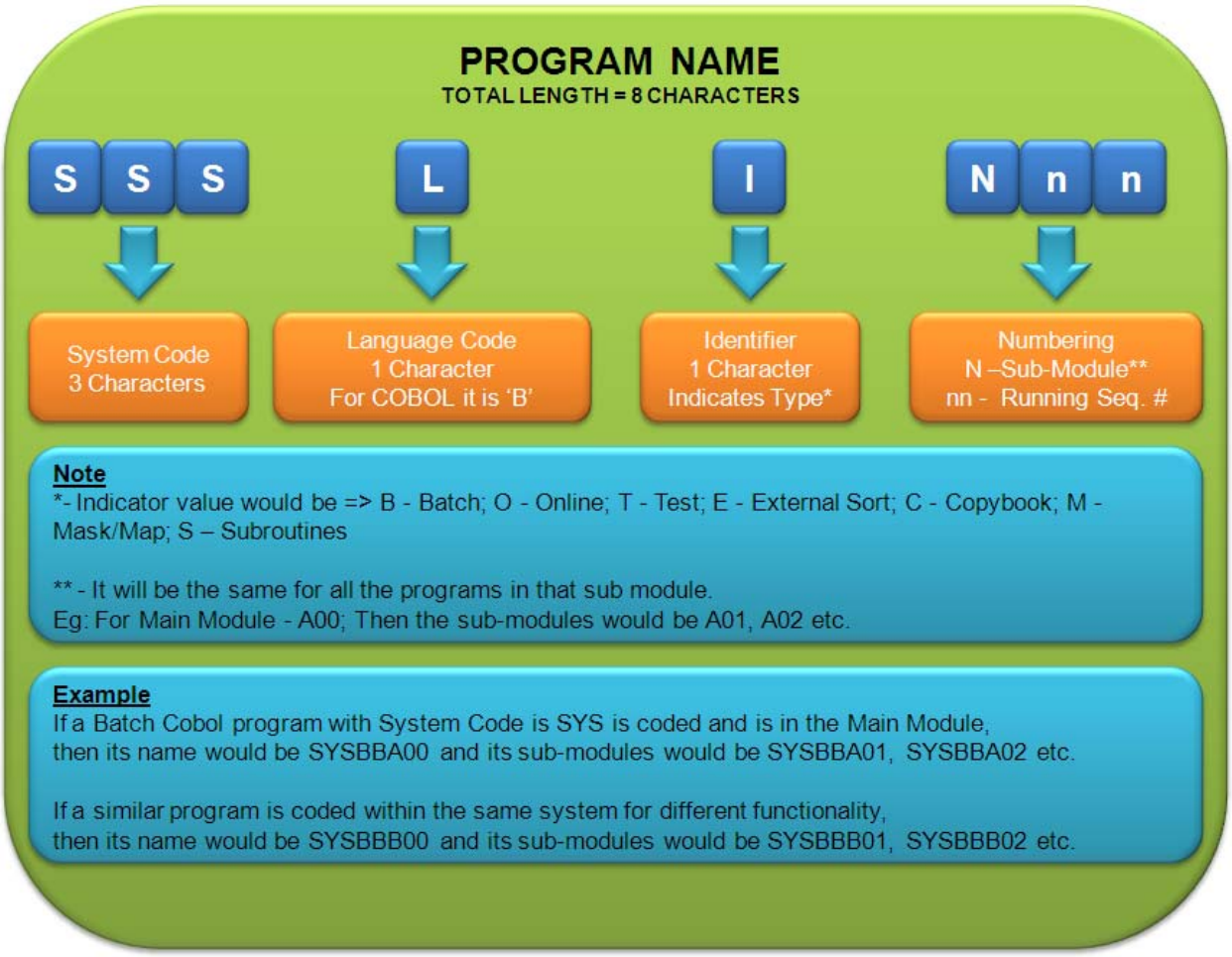
Overview

Standards are required for easy maintenance and debugging whatever may be the language. As COMMON BUSINESS ORIENTED LANGUAGE (COBOL) is used extensively by major mainframe applications, the document deals with the naming conventions of the modules, coding standards, indentations for better readability, performance conditions, good practices and the dos and don'ts involved.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

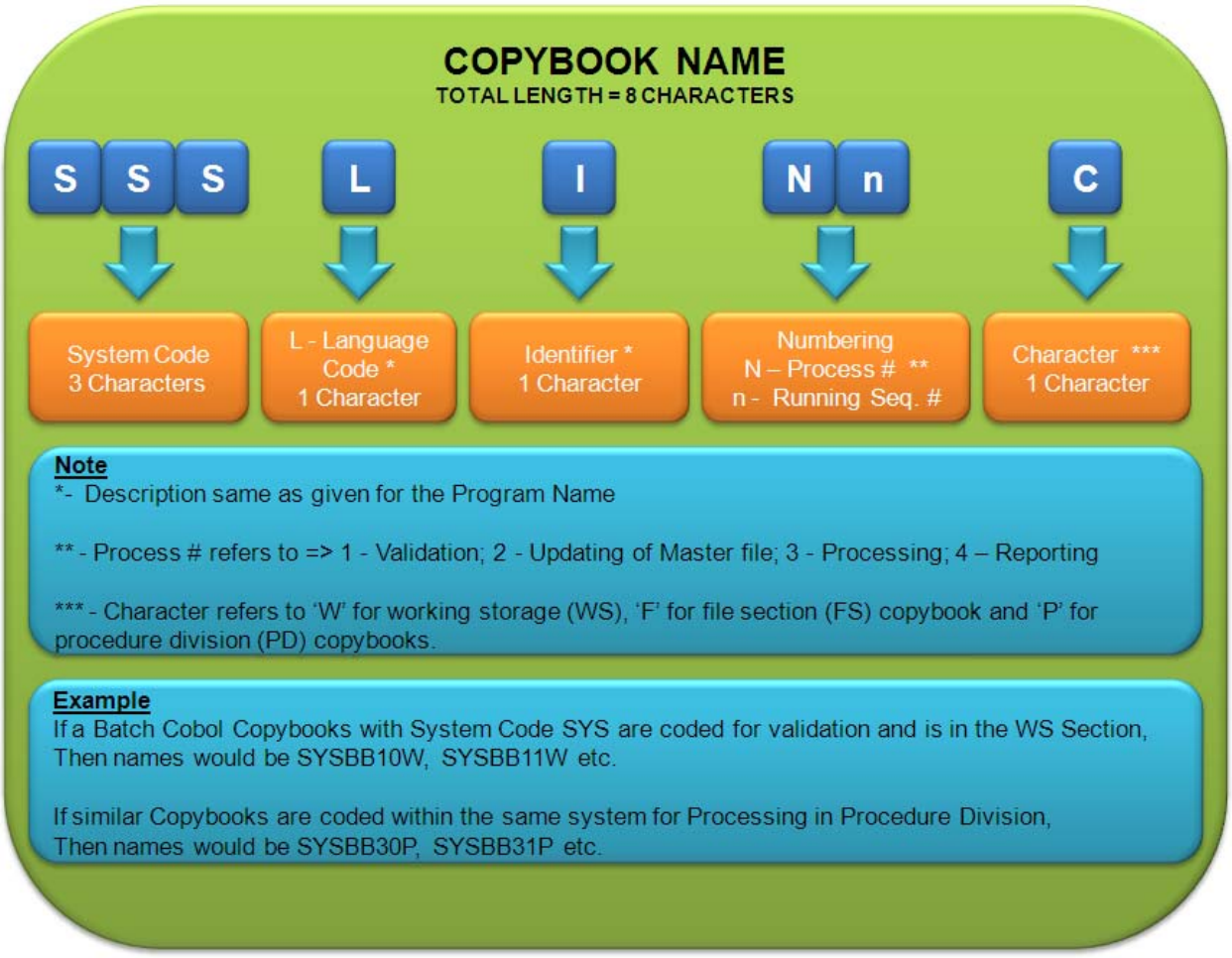
2. Naming Conventions

Program Naming Convention



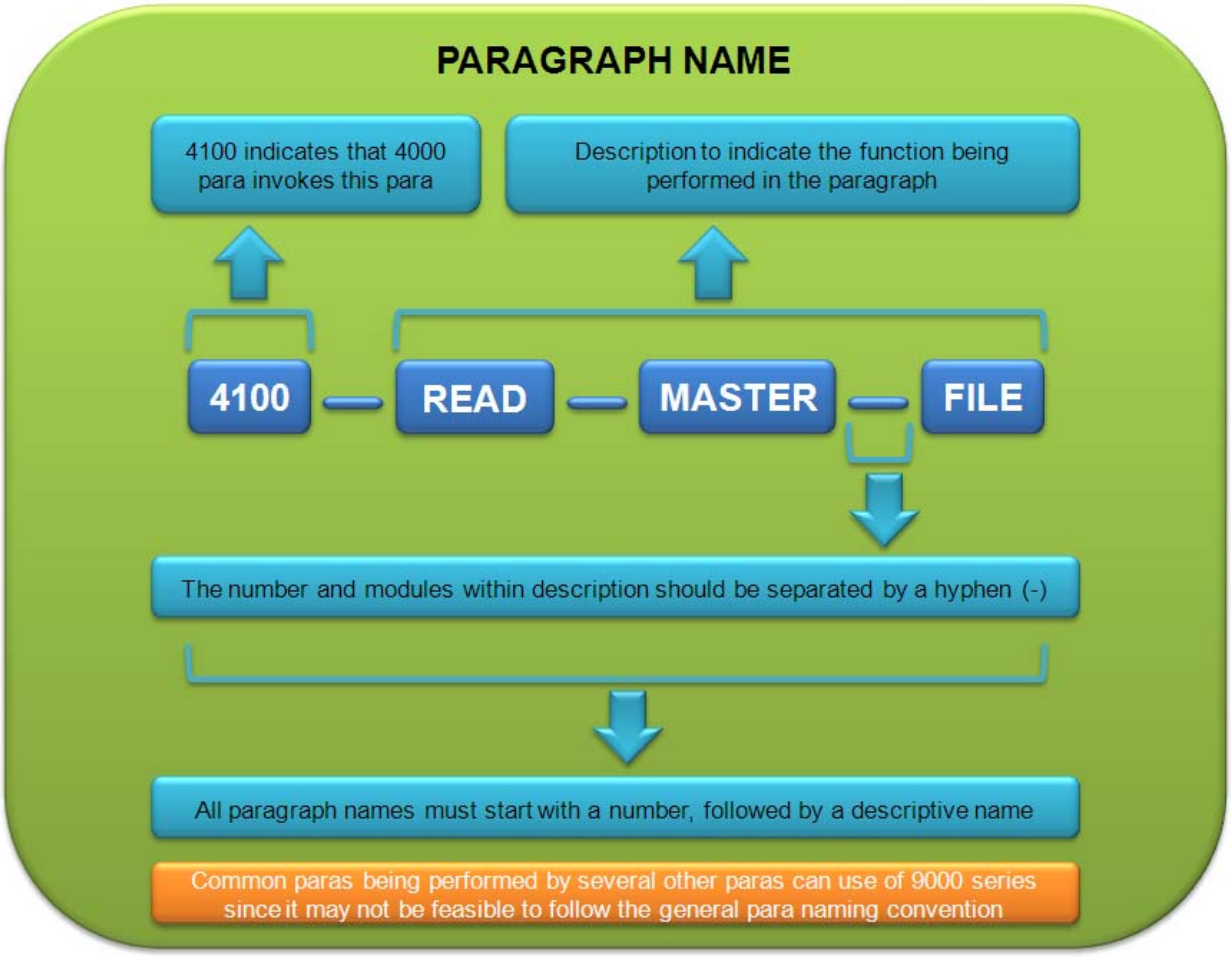
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Copybook Naming Convention



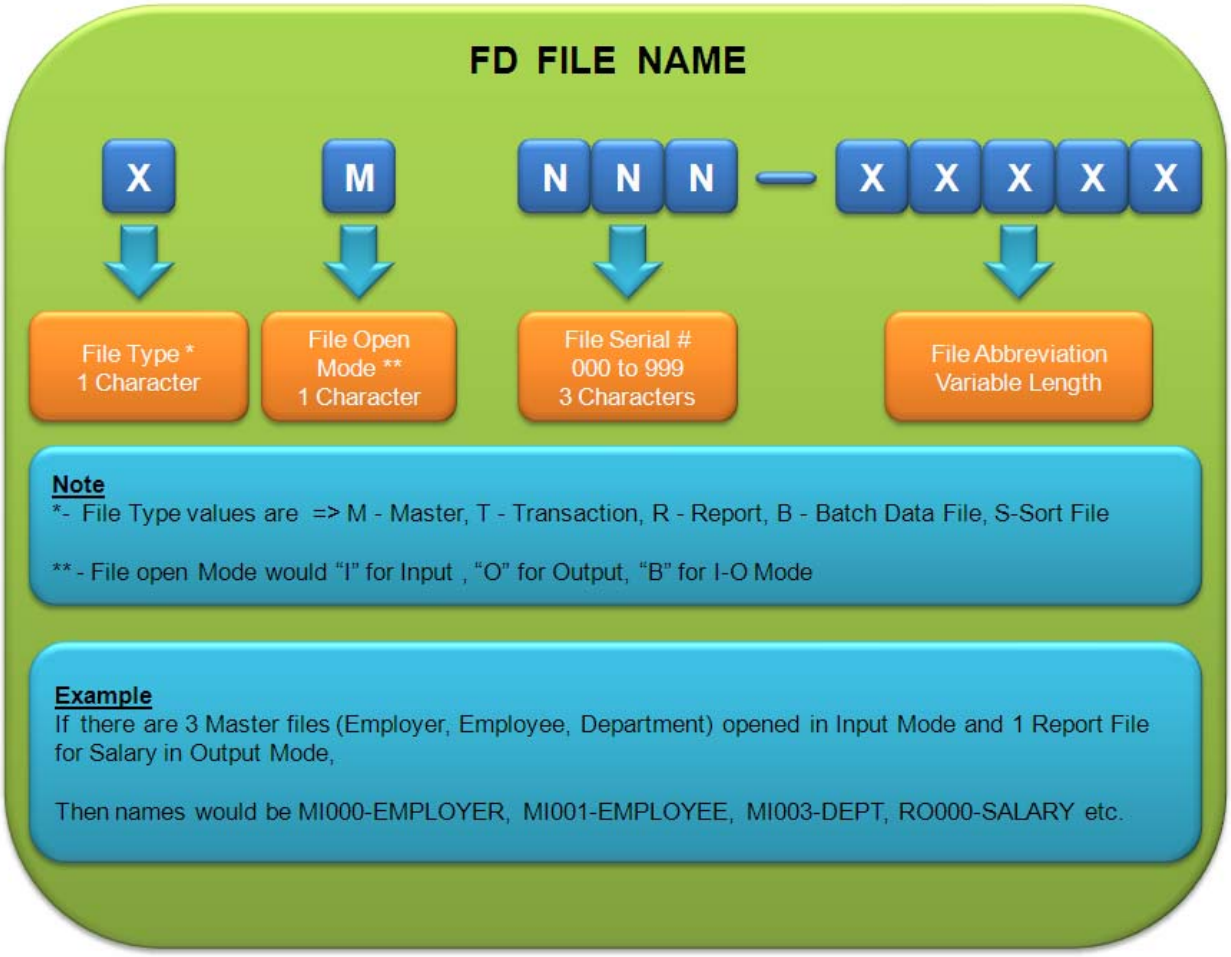
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Paragraph Naming Convention



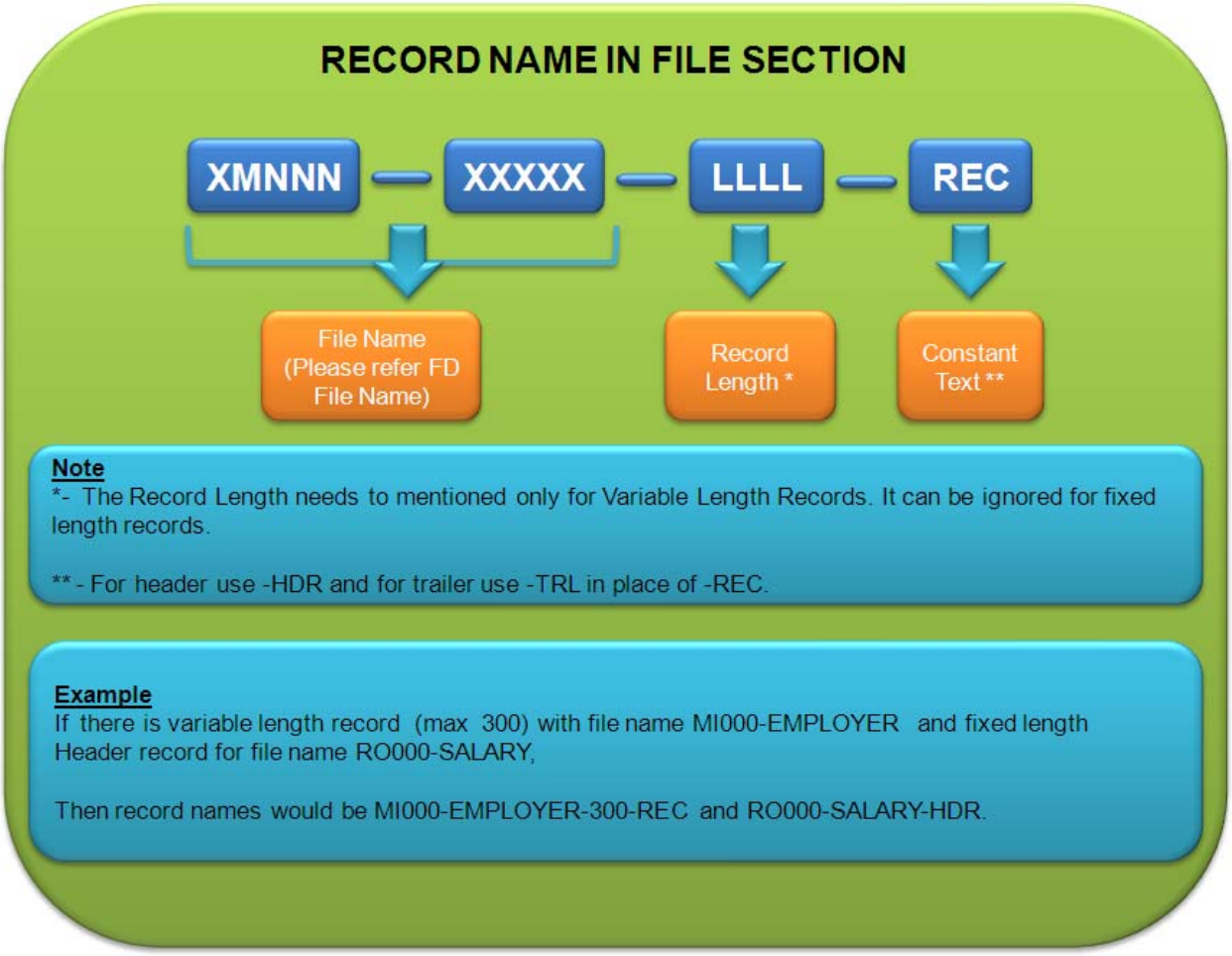
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

FD File Naming Convention



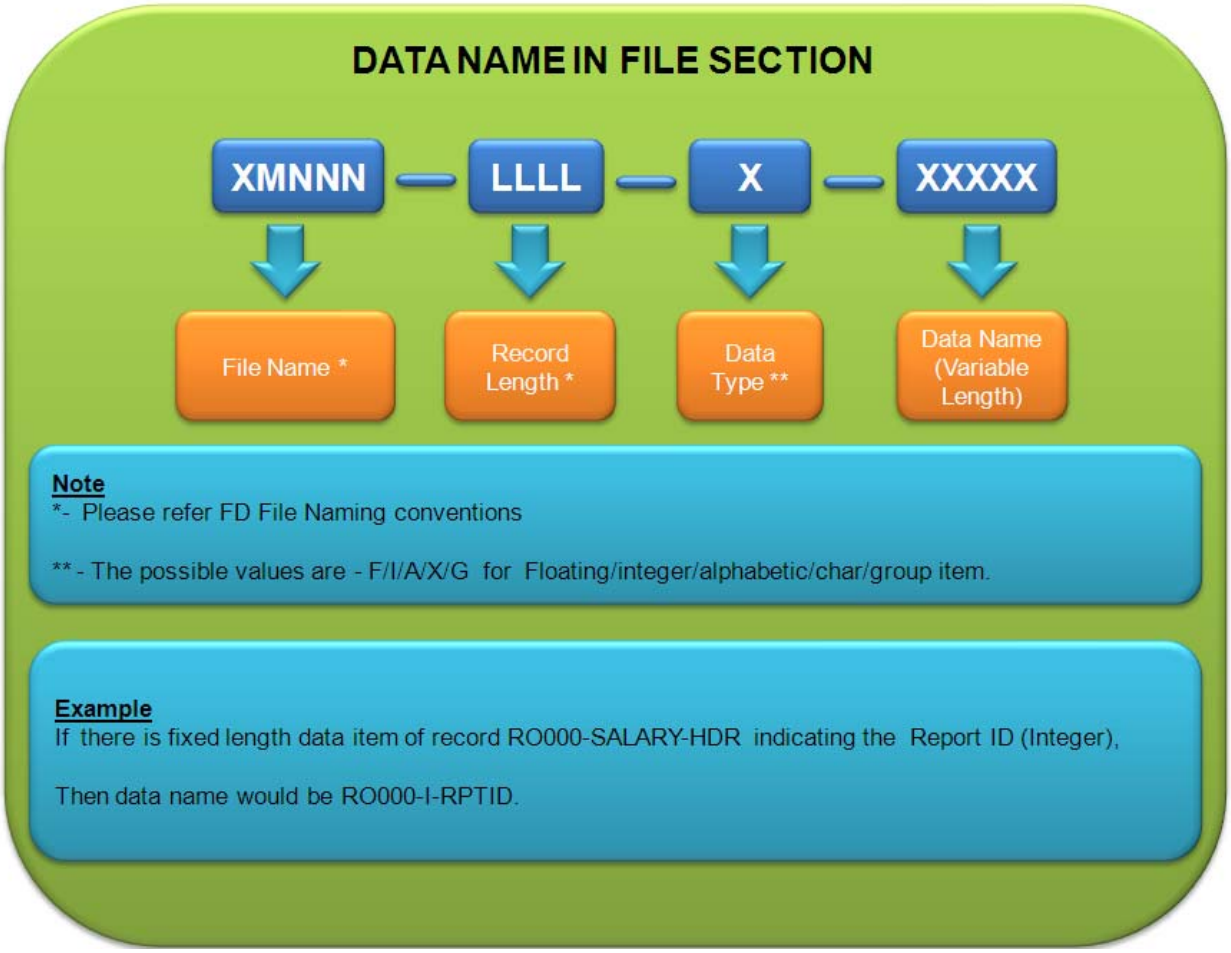
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Record Naming Convention in File Section



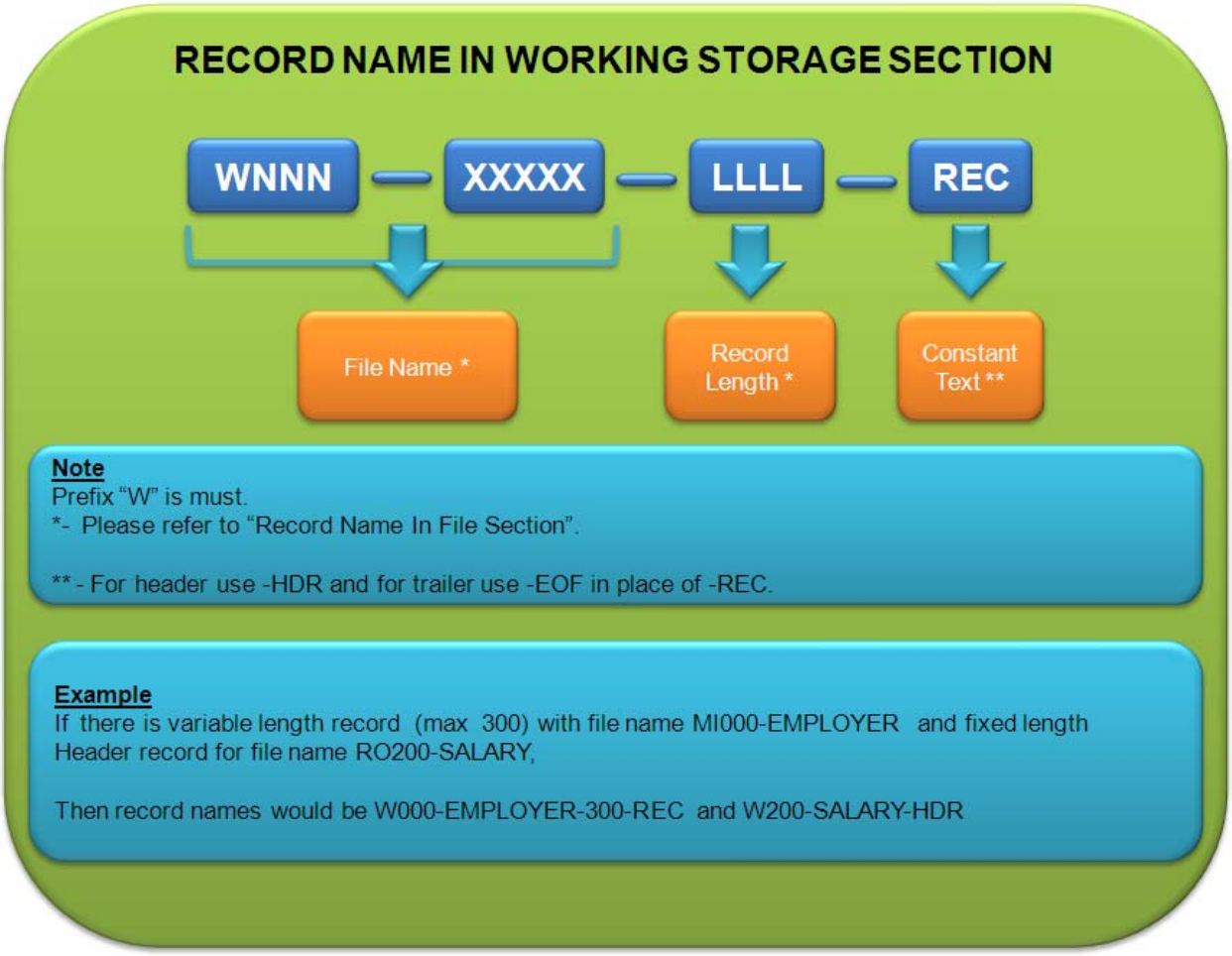
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Data Naming Convention in File Section



©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

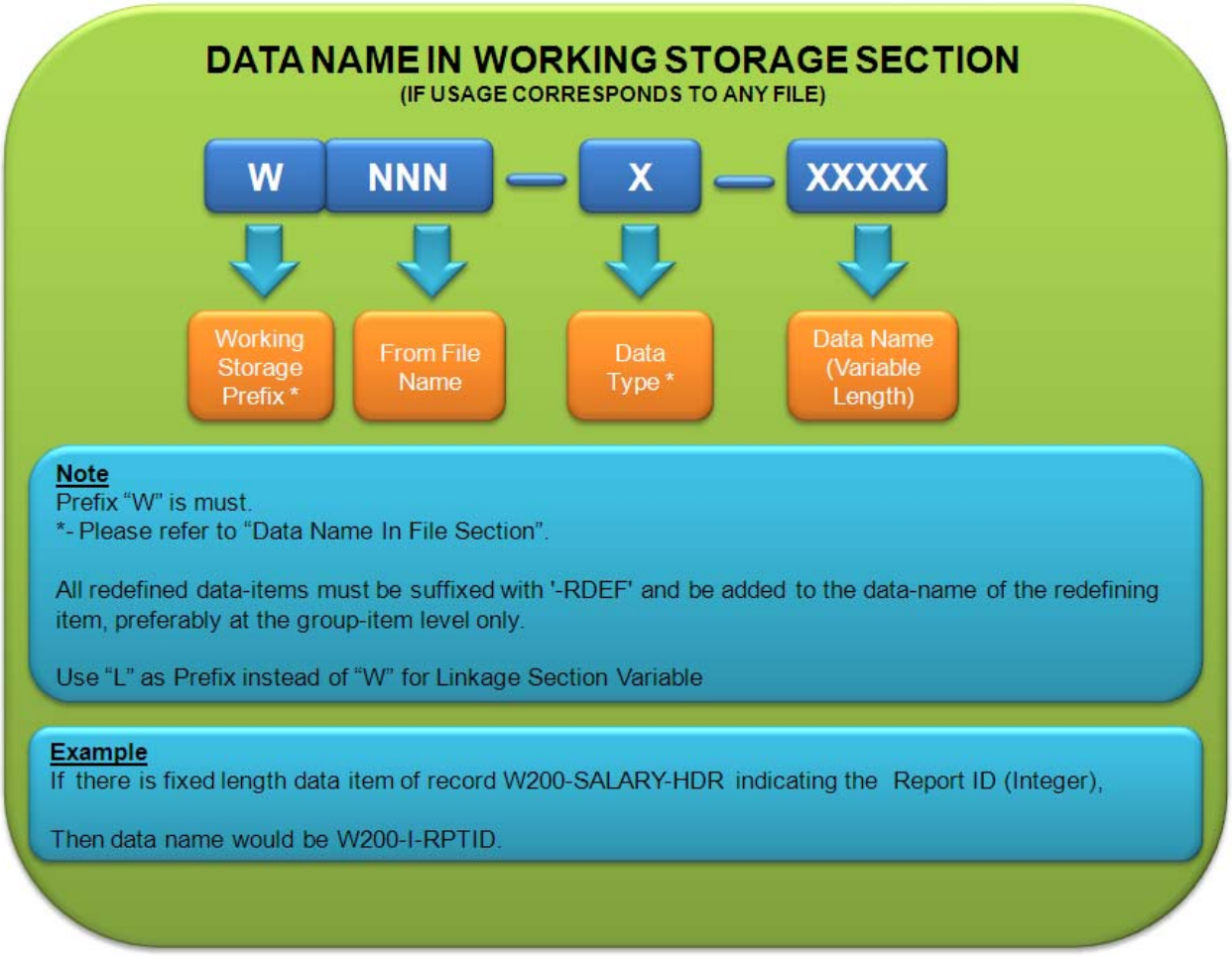
Record Naming Convention in Working Storage Section



©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

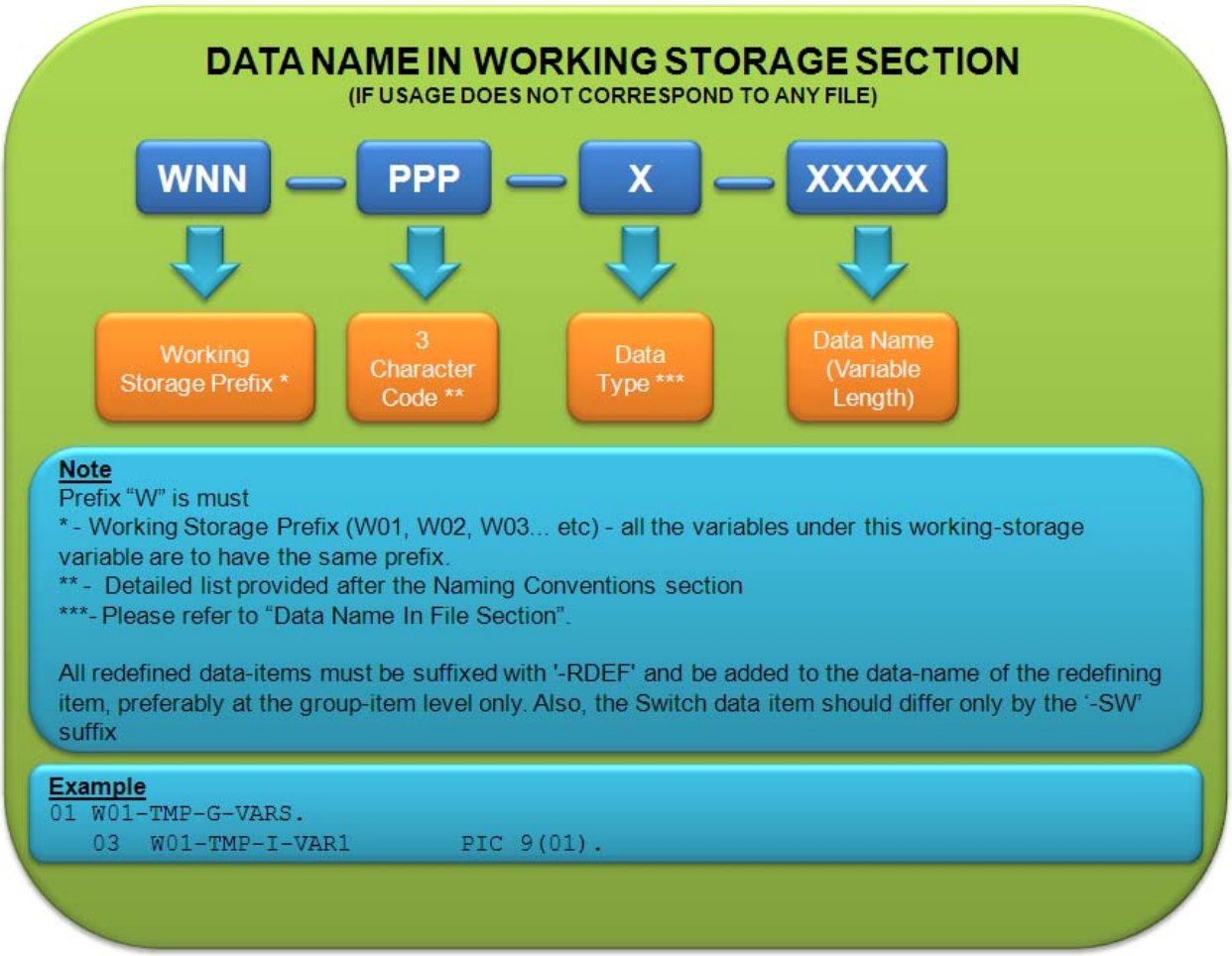
Data Naming Convention in Working Storage Section

If usage corresponds to any file



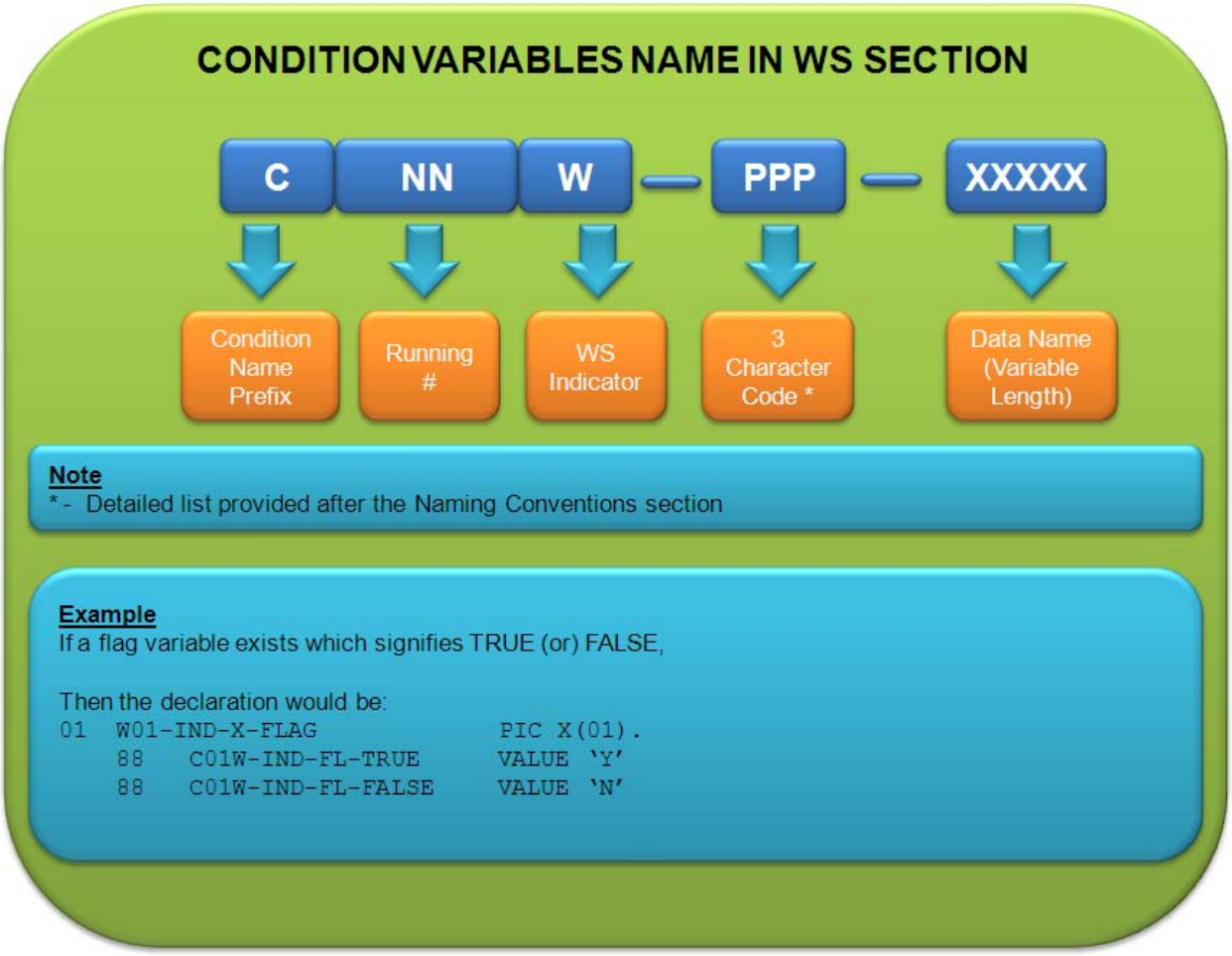
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

If usage does not correspond to any file



©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Condition Variables Naming Convention in WS Section



References to Naming Conventions

3 Character Codes

Description	3 Char Code	Description	3 Char Code	Description	3 Char Code	Description	3 Char Code
Abend Var.	ABD	Subscript	SUB	File Status	FST	Page counter	LPC
Vsam Code	VSM	Literal	LIT	Counters	CNT	Tables	TAB
Flags	FLG	Value of id	VID	Accumulators	ACC	Control point	CPT
Index	INX	Value literal	VAL	Line counter	LIC	Indicator	IND

Cognizant
Passion for making a difference
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

3. Coding Standards in Identification Division

Essential Pointers

Coding Standard

- The program -id and program name must be the same.

Indentation

- The identification division and program-id and all divisions must start at column 8

Good Practice

- A brief program description needs to be provided at the beginning of the program.

```
*********************************************************
*******         BRIEF PROGRAM DESCRIPTION          *******
*********************************************************
* AUTHOR            :
* DATE-WRITTEN      :
* PROGRAM OBJECTIVE :
* INPUT                  :
* OUTPUT            :
* ENTRY FROM        :
* REMARKS           :
*
* DEVELOPMENT / MODIFICATIONS HISTORY
* REVISION NO.  AUTHOR   DATE       COMMENTS
*********************************************************
*
/
```


©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

4. Coding Standards in Environment Division

Essential Pointers

Coding Standard

- The configuration section, input-output section, and file-control must be coded.
- Source computer name and the Object computer name must be coded.
- The file status code must be specified for all the files.
- For VSAM files, the ACCESS and RECORD-KEY must be specified.

Indentation

- All SELECT statements must start in column 12

Good Practice

- The SELECTs for a given type of file should be grouped together, starting with input files, followed by output files, and eventually input-output files. The orders for SELECT statements have to be followed.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

5. Coding Standards in Data Division

File Section

Indentation

- The optional statements should be aligned in column 12, each one appearing on a separate line.
- Some examples of optional statements are:

```
LABEL RECORDS OMITTED
LABEL RECORDS STANDARD
BLOCK CONTAINS 0 RECORDS
RECORDING MODE IS F
DATA RECORD IS <record name>
```

Good Practice

- The optional statements are coded such that each one should be appearing on a separate line.
- To allow the JCL (Job Control Language) to supply the block size, there is the 'BLOCK CONTAINS' clause coded with '0 RECORDS '.
- All the File Description (FD) and Sort Description (SD) entries in the file section appear in the same order as in the FILE CONTROL select statements.

Working Storage Section

Indentation

- The 01 level variables should begin in column 8, with the first data name in column 12.
- One or four spaces must appear between the level number and the data-name throughout. In the following example there are four spaces between '01' and 'W01-INPUT-RECORD'. Similar is the case with '05' and 'W01-INPUT-KEY'.

```
01    W01-INPUT-RECORD.
   05    W01-INPUT-KEY                              PIC X(10).
```

- A spacing of one line is to be maintained between the two working storage group variables.

Good Practice

- The dead variables declaration in working storage needs to be avoided.
- The level# must be odd numbers in increments of two, for example 01, 03, 05,07,09,11, and so on.
- The PICture clause must have minimum two digits in brackets for example code PIC 9(01) or PIC X (02) instead of PIC 9 or PIC XX.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

- Specify date variables in CCYYMMDD format. It is easy to sort the data based on this field if declared in the earlier format.
- If the maximum dimension of a table is more than three levels, then it should be avoided.
- When a program calls another program, all WORKING-STORAGE items referenced by the called program should be grouped under a single group level if possible, to avoid the confusion of passing many data names.

Performance

- Subscripts must be defined as COMP SYNC (Binary Synchronized) for efficiency and speed.
- Tables defined with an 'OCCURS DEPENDING ON' are slower than fixed length tables and should be avoided whenever possible.

Linkage Section

Standard

- There are three such sources, which are another COBOL program, which is calling this program, a JCL statement that is passing PARM (parameters) data, or PCB (Program Communication Block) from an Information Management System (IMS) main program.
- The data items in the LINKAGE SECTION must match the data items in the calling source, in terms of number, order, name, and definition.

To any Section in Data Division

Indentation

- In variable declaration, all the `PIC` clauses must be aligned to start in column 48 if possible.

Good Practice

- The usage of 01 level for elementary data items must be avoided.
- The usage of 66 and 77 level variables must be avoided.
- The COMP-3 fields defined with an odd number of digits before the decimal point in the 'PIC' clause for example PIC 9(07)V9(02) COMP-3.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

6. Coding Standards in Procedure Division

Essential Pointers

Coding Standards

- Coding statements must be coded within: Only section/ only paragraph/ Paragraphs under section.
- After each I-O operation the file status must be checked.
- The scope terminators must be used for the `IF`, `PERFORM`, and `EVALUATE` statements.
- The `'CLOSE'` statement has been coded once for each file. Also all the files that have been opened must be closed.
- 'Read Into' must be coded for all read operations.
- 'Write From' must be coded for all write operations.
- Standard subroutines must be used.
- The definition of the para must be followed by its exit para.
- `IF` clause should not be split by comments.
- The comments must have a reference back to the original design documentation wherever possible.
- Explicit Scope Terminators need to be used.
- The '`THRU`' option of the '`PERFORM`' should always be used. Exiting out of a `PERFORM` loop should always be through the corresponding exit paragraph.
- In an '`IF`' or '`PERFORM UNTIL`' with an '`OR`' in it, the check for exceeding the table limit should be made before all other checks.
- Also when actually accessing a table, its subscript or index should never have a value less than one or greater than the number of occurrences of the table.
- Possible overflow of the used tables must be checked.
- If an error is encountered while the program is in an `INPUT` or `OUTPUT PROCEDURE` and the program must end abnormally (called ABEND in IBM terminology), then care must be taken to first display the output before terminating the program. It should include `FILE-NAME`, `RECORD KEY`, `FILE STATUS` values, and other relevant fields.
- No values should be moved to the parameters '`SORT-CORE-SIZE`', '`SORT-FILE-SIZE`', or '`SORT-MORE-SIZE`'.
- When using arrays, subscripts cannot have arithmetic expressions in them but indexes can have. For example `W-TAB-I-ENTRY (W-SUB-I-ENTRY + 1)`. To avoid this situation avoid using arithmetic operations in the array elements.
- The '`EXAMINE`' statement should never be used. Instead of it you use '`INSPECT`'.
- Avoid using the 'TALLY' special register. Hence it should be replaced with an appropriate WORKING-STORAGE counter.
- Re-initialize WORKING STORAGE items for each new execution (Sub-programs) or input message (on-line IMS programs).
- Sections and paragraphs should have a single entry and exit point.
- Basic validations need to be performed on the data read from the input source.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

- The 'WITH TEST BEFORE' clause can be used to specify a condition test before executing the loop, but such coding is redundant and should not be used because this is the default option if no 'WITH TEST BEFORE/AFTER' clause is specified.

Indentation

- All section or paragraph names begin in column 8.
- Comments should be aligned to the left or indented with the code.
- MOVE and TO should be aligned where possible.
- Verb qualifier clauses (for example, 'UNTIL', 'THRU', 'TO' and so on) should be indented to the right in a separate line, under the verb with which they are connected. The exception to this rule is 'VARYING'.
- IF-ELSE should be coded such that:
  - IF is in alignment with the corresponding ELSE
  - ELSE appears on a line by itself
  - Statements constituting the IF and ELSE branches are indented to the right

Good Practice

- Each file should be opened only once.
- The 'GO BACK' (sub-programs) or 'STOP RUN' (MAIN programs) must be coded only once in the program.
- Commented code needs to be avoided. This helps in better readability of the code.
- Logically related process or actions should be kept in one paragraph.
- Each I-O activity on a file is to be performed from only one paragraph. Use only one READ and one WRITE paragraph for each file.
- Do not use hard coded literals in the procedure division:
  - When constants have to be used in the program (For example, PERFORM XYZ-SECTION 20 TIMES).
  - When checking for conditions (for example, IF W-INI-X-MARSTAT = 'S')
  - When displaying error or help messages.
- Use the EVALUATE clause instead of multiple 'IF' then 'ELSE' statements.
- "The following COBOL options needs to be avoided:
  - Alter
  - Go To
  - Move Corresponding
  - Level 77's
  - Level 66's
- The 'GO TO' is allowed only if it branches to the EXIT of a paragraph.
  - Backward branching, or branching between paragraphs is not recommended.
  - GOTO statements should always go to the EXIT PARA only"
- Use descriptive comments that will explain the purpose and function of the section or paragraph and complex logic, if any. Use embedded comments to further clarify logic.
- Do not use commas and semicolons, as they do not have any syntactic function and obscure readability.
- Using level-88 names for checking conditions
- Parentheses should be used to set off the operands and improve readability especially needed when mixed type of tests are done - Relational (>, <, =, Not), Conditional (88-Levels), Class (Numeric, Alphabetic), and Sign (Positive, Negative) Tests.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

- A maximum of three nested IF statements only should be coded.
- Using negative conditions must be avoided, like `IF NOT` condition.
- Switch variables must be implemented with 88 Level Variables
- `SECTIONS` should not be used.
- Before executing divide operation make sure that the denominator is not zero (provide a check for the denominator being non-zero).
- All unnecessary displays added for testing purposes should be removed from programs on completion of testing and prior to handover to Systems Assurance, with the exception of on-line programs.
- In this case all `DISPLAY` commands, which remain in the source code must have a 'D' in column 7 so that the `DEBUGGING MODE` can be turned off prior to handover.
- Move the appropriate value to `RETURN-CODE` before terminating the program
- There should be only one error routine within a program, preferably the standard for that particular system.
- Subscripts should be used instead of indexes, even though indexing is more efficient than subscripting. This is because subscripts are normal WORKING-STORAGE fields that can easily be found in a dump, while indexes are fields generated by compiler that are not so easily found.
- Repeat necessary edited picture characters rather than to use parentheses. For example, use ZZZ,ZZZ.99 rather than Z(3), Z(3).9(2) even though COBOL allows it.
- Always initialize numeric data, print records, and switches or flags.
- Use ascending numbers preceding all section and paragraph names. This provides easy visual reference to the location of the sections within the program. It also represents the flow of the program logic.
- Perform multiplication before division to reduce your truncation errors to maintain accuracy.
- If an indexed file has access mode dynamic, then sequential read should be done. `Start` statement must be coded to position the file pointer where required, followed by `read next` statement in a paragraph that executes in a loop until some specified condition.
- Inline comments are to be provided where ever necessary.
- Do not use comments as pseudo code.

Performance

- Use “Read into” or “Write from”.
- Ordering the code so that the most frequently occurring situations are first.
- Use a separate `INITIALIZATION-ROUTINE` to initialize data, open files, tables, read first record, and perform other operations required prior to processing the data.
- There should be no redundant or un-executable or `DEAD` codes in the program.
- Intervening section names are not allowed in a performed section. Each section may have only one entry and one exit.
- If a program has the same code in several places, then it is prudent to create a section containing this code, and executes it (through a ‘PERFORM’) whenever required.
- ‘`COMPUTE`’ generates more efficient code than separate arithmetic statements because the compiler can keep track of internal work areas and does not have to store the results of intermediate calculations.

Cognizant
Passion for making a difference
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

- Operations are more efficient when the data items involved have the same number of decimal places.
- It is more efficient to use a 'SEARCH' statement than a 'PERFORM...UNTIL' to search a table

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Glossary

Term	Description
ABEND	Abnormal end of task (ABEND) - The termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve during execution.
Batch	Pertaining to a group of jobs to be run on a computer sequentially with the same program with little or no operator action. A group of records or data processing jobs brought together for processing or transmission.
COBOL	Common Business Oriented Language
Division	A division is a block of code, usually containing one or more sections, that starts where the division name is encountered and ends with the beginning of the next division or with the end of the program text.
FD	File Description
File	A collection of related data that is stored and retrieved by an assigned name.
IMS	Information Management System - Any of several system environments available with a database manager and transaction processing that are capable of managing complex databases and terminal networks.
Index	A set of pointers that is logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness of the key values for the rows in the table.
Literal	A character string whose value is defined by the characters themselves. For example, the numeric constant 7 has the value 7, and the character constant 'CHARACTERS' has the value CHARACTERS.
Online	Pertaining to a user's access to a computer by way of a terminal. Pertaining to the operation of a functional unit or device that is under the control of the system or of a host.
Paragraph	A paragraph is a block of code made up of one or more sentences. A paragraph begins with the paragraph name and ends with the next paragraph or section name or the end of the program text.
SD	Sort Description
Section	A section is a block of code usually containing one or more paragraphs. A section begins with the section name and ends where the next section name is encountered or where the program text ends.
Sentence	A sentence consists of one or more statements and is terminated by a period
Statement	A statement consists of a COBOL verb and an operand or operands
Subroutine	A sequence of instructions within a larger program that performs a particular task. A subroutine can be accessed repeatedly, can be used in more than one program, and can be called at more than one point in a program.
Subscript	One or more expressions, each enclosed in brackets that follow an array name. A subscript refers to an element in an array. In COBOL, a positive number or variable whose value refers to a particular item in a table.
Table	In COBOL, a set of logically consecutive data items that are defined in the Data Division with the OCCURS clause.

Cognizant
Passion for making a difference
©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

Term	Description
VSAM	Virtual Storage Access Method - An access method for direct or sequential processing of fixed-length and variable-length records on disk devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative-record number.

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

References

Websites

- Cognizant Coding Standards / Guidelines can be accessed from the site:
- https://cognizantonline/qview/reference/standards.html

Books

- COBOL Programming by M K Roy, D Ghosh Dastida
- Structured COBOL Programming by Stern & Stern

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected

STUDENT NOTES:

©Copyright 2007, Cognizant Technology Solutions, All Rights Reserved
C3: Protected