



JCL Coding Requirements

By proceeding with this courseware you agree with [these terms and conditions](#). Interskill Learning Pty. Ltd. © 2019





Objectives

JCL Coding Requirements

Before beginning to write any JCL code, you need to become familiar with the formatting of JCL statements, and the consequences should these not be followed. In this module you will look at the syntax, and format of JCL statements generally and will need to combine these items with your own organizational JCL standards.

At the end of this module, you should be able to:

- Describe the Formatting Requirements of a JCL Statement
- Identify Common JCL Coding Errors

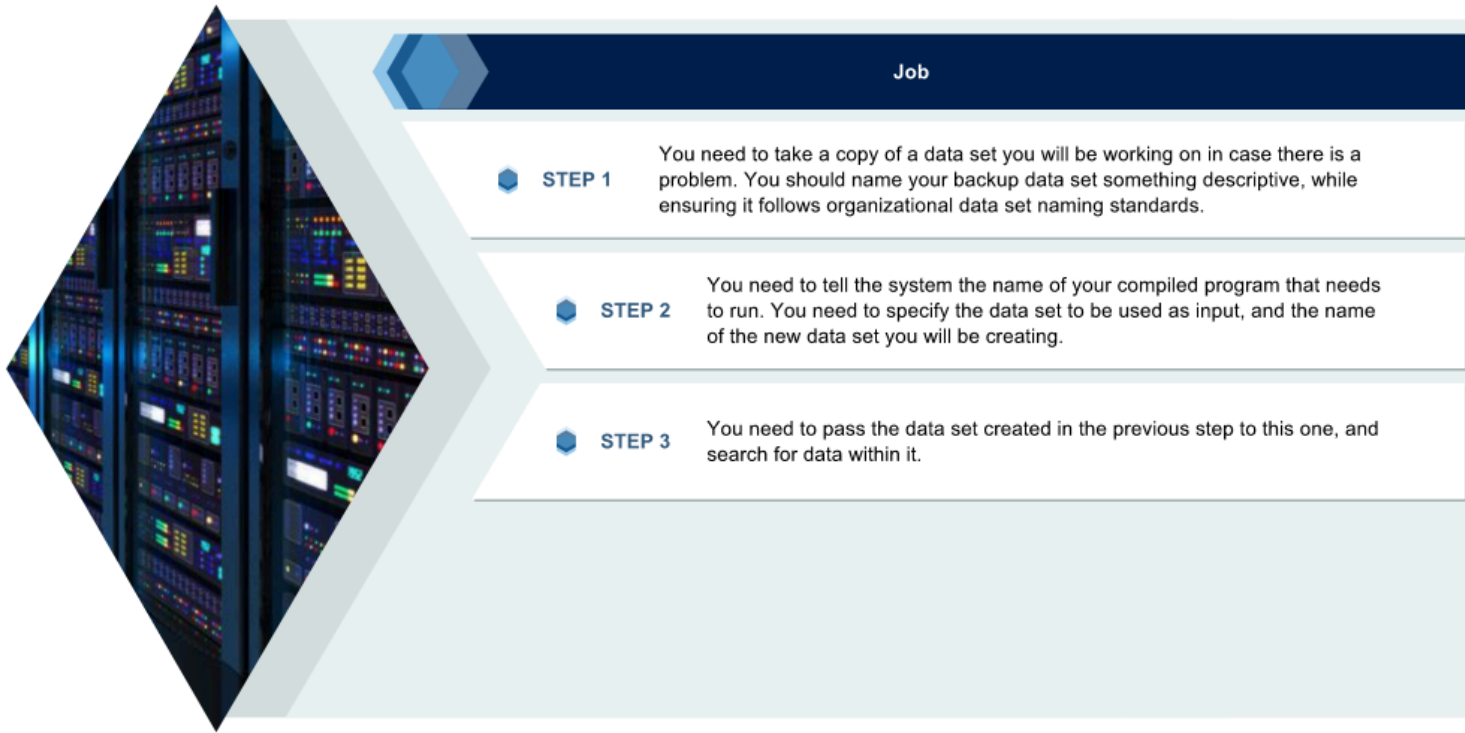


- You know where your JCL needs to be stored.
- You are aware of your organization's JCL coding standards.
- You know the software products that you need to access to create, and manage, your batch jobs.
- You know that batch processing can be performed manually, or autonomously, using workload scheduling software.

In the previous module you saw that there were several items you needed to think about before even laying your eyes on JCL.

Now that these have been addressed, it is time to take a high-level look at the building blocks that form the structure of JCL.





Before writing any JCL you need to have a plan of the batch processing you need to perform. For example, you may need to do the following:

- Take a backup of a data set before you work on it
- Run a COBOL program you have been working on, against the data set you copied
- Create a report that identifies discrepancies between the result of your program, and another set of data

All these tasks can exist within a single batch job and are broken up into steps.



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.00      Member MASTCOP copied
Command ==>                               scroll ==> CSR
***** Top of Data *****
000100 //FAMXTGH JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER
000200 //*
000300 //STEP1 EXEC PGM=ICEGENER
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=GHMAST.FAMAPS.D256,DISP=OLD
000600 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP,
000700 // DISP=(,CATLG)
000800 //SYSIN DD DUMMY
***** Bottom of Data *****
```

As this is only the first step, you would need to copy some JCL from another job, or write it yourself, to invoke other programs you need to run against the FAMAPS data set.

◀ Previous Restart

While you could write all the JCL required for the job on the previous page from scratch, the reality is that you are more likely to copy JCL from a similar job, or part of one, into a new PDS member and make general changes before submitting it. Your vendor may also supply sample libraries containing basic JCL for tasks you may need to perform.

Why do you need to know all this information about JCL code? The answer is that you need to understand the instructions you are passing to the system, especially when jobs fail, and you need to diagnose system messages.

Click Next to see how JCL is copied into this empty PDS member and modified.



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT          IBMUSER.JCL(FAMXTGH) - 01.00          Columns 00001 00072
Command ==>                                     Scroll ==> CSR
***** Top of Data *****
000100 //FAMXTGH JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER
000200 //*
000300 //STEP1 EXEC PGM=ICEGENER
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=GHMAST.FAMAPS.D256,DISP=OLD
000600 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP,
000700 //          DISP=(,CATLG)
000800 //SYSIN DD DUMMY
***** Bottom of Data *****
```

The JCL from the previous page is a simple example, where there is only a single step running a copy program.

While this JCL may look confusing to someone who has never worked with this code before, by the end of these JCL courses you will be able to confidently identify the purpose of these statem

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.00                Columns 00001 00072
Command ==> COLS                                     Scroll ==> CSR
=COLS>
*****
***** Top of Data *****
000100 //FAMXTGH JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER
000200 *
000300 //STEP1 EXEC PGM=ICEGENER
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=GHMAST.FAMAPS.D256,DISP=OLD
000600 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP,
000700 //DISP=(,CATLG)
000800 //SYSIN DD DUMMY
*****
***** Bottom of Data *****
```

These characters correspond to columns 1 and 2, when looking at the COLS display line above.

The RESET or COLS OFF command can be used to remove the COLS line from the display.

Now you will begin dissecting the JCL to see what is required.

JCL statements begin with double slash (//) characters appearing in columns one and two. This is how the system is able to interpret the data that you submit to the system is JCL. There are a 1 JCL rules that dictate the column in which information can be placed, therefore it is usually a good idea to enter the ISPF edit command COLS in the command line, to display this information at top of your data.

Note that having said that JCL statements must begin with double slash (//) characters, there are some exceptions, but these are discussed in later courses.


```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(AGENER) - 01.05                Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000100 //AGENER JOB MSGCLASS=C,MSGLEVEL=(1,1)
000200 //* STEP 1 - TEST GDG BASED CONCATENATION
000300 //STEP1 EXEC PGM=IEBGENER
000400 //SYSPRINT DD SYSOUT=*
000500 SYSUT1 DD DSN=IBMUSER.PEN.DAILY.TRANS(0),DISP=SHR
000600 //SYSUT2 DD SYSOUT=*
000700 //SYSIN DD DUMMY
000800 // IF STEP1.ABEND=TRUE THEN

Display Filter View Print Options Search Help
-----
SDSF OUTPUT DISPLAY AGENER JOB09308 DSID 2 LINE 19 COLUMNS 02- 81
COMMAND INPUT ==> SCROLL ==> CSR
      12 SYSOUT SPOOL KBYTES
      0.00 MINUTES EXECUTION TIME
1 //AGENER JOB MSGCLASS=C,MSGLEVEL=(1,1)
  //* STEP 1 - TEST GDG BASED CONCATENATION
2 //STEP1 EXEC PGM=IEBGENER
3 //SYSPRINT DD SYSOUT=*
4 //SYSIN DD * GENERATED STATEMENT
5 //SYSUT2 DD SYSOUT=*
6 //SYSIN DD DUMMY
7 // IF STEP1.ABEND=TRUE THEN
```

The message you receive as a result of forgetting to code // will be slightly different depending on where it is located. In this example, there are no // characters on line 000500, and when this is submitted, the system thinks that you are attempting to pass raw data to the system, and it has generated a //SYSIN DD * statement.

◀ Previous Restart

Writing JCL for the first time, it can be easy to forget to code the double slash (//) characters. In the example here, // is missing from the start of line 000100.

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(COBCOMP) - 01.99                Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000100 //COBCOMP JOB 123,'COBOL COMPILE',CLASS=A,MSGCLASS=X,
000200 //          NOTIFY=IBMUSER
000300 //          *
000400 //          SET MEM=COMPUTE2
000500 //          *
000600 //COMP1 EXEC PGM=IGYCRCTL,REGION=0M,
000700 //          PARM='LIST,XREF'
000800 //          *
000900 //          SCHEDULE STARTBY='+00:05'
001000 //SYSIN DD DISP=SHR,DSN=IBMUSER.COBOL.SRC(&MEM)
001100 //SYSLIB DD DISP=SHR,DSN=IBMUSER.COBOL.SRC
001200 //          DD DISP=SHR,DSN=CEE.ACEESRC1
001300 //SYSPRINT DD SYSOUT=B
001400 //SYSLIN DD DISP=(MOD,PASS),DSN=&&LOADSET,SPACE=(80,(10,10)),
001500 //          UNIT=SYSDA
001600 //SYSUT1 DD SPACE=(CYL,(1,1)),UNIT=SYSDA
001700 //SYSUT2 DD SPACE=(CYL,(1,1)),UNIT=SYSDA
001800 //SYSUT3 DD SPACE=(CYL,(1,1)),UNIT=SYSDA
001900 //SYSUT4 DD SPACE=(CYL,(1,1)),UNIT=SYSDA

```

A common mistake for new JCL users is to submit a job that contains a line where the only data is //. This type of statement is called a null statement and indicates that this is the end of your JCL

You can see in the example here that the user wanted to remove the content from a JCL statement, however left // by itself on line 000500. Even though JCL statements appear after this line, the system will not recognize them.

Click Play to see how the system interprets this code.



```
Display Filter View Print Options Search Help
-----
SDSF OUTPUT DISPLAY CSFSMFJ JOB09311 DSID 3 LINE 1 COLUMNS 02- 81
COMMAND INPUT ==> SCROLL ==> CSR
1 //CSFSMFJ JOB (), 'GMH RUN', CLASS=A, MSGCLASS=X, NOTIFY=gh8ibm
//*****
//* UNLOAD SMF RECORDS TO PRINT
//*
2 //SMFDMP EXEC PGM=IFASMFDP
3 //DUMPIN DD DISP=SHR, DSN=SYS1.S0W1.MAN1.DATA
4 //DUMPOUT DD SYSOUT=*
5 //SYSPRINT DD SYSOUT=*
6 //SYSIN DD *
STMT NO. MESSAGE
1 IEFC620I UNIDENTIFIABLE CHARACTER g ON THE JOB STATEMENT
1 IEFC620I UNIDENTIFIABLE CHARACTER h ON THE JOB STATEMENT
1 IEFC620I UNIDENTIFIABLE CHARACTER i ON THE JOB STATEMENT
1 IEFC620I UNIDENTIFIABLE CHARACTER b ON THE JOB STATEMENT
1 IEFC620I UNIDENTIFIABLE CHARACTER m ON THE JOB STATEMENT
*****
***** BOTTOM OF DATA *****
```

You can see the messages at the bottom of this job output, indicating that the system could not interpret the lowercase characters.

Another thing you will notice with JCL is that it is coded in uppercase characters. In fact, your edit profile - displayable using the PROF command in the command line - may show that CAPS is to ON, meaning that any data you enter is automatically converted to uppercase when you press the Enter key.

In the example shown here, the profile shows as CAPS OFF allowing lowercase characters to be accepted. The person updating this job, accidentally typed in their own user ID at the end of line 000001 in lowercase characters.

Click Play to see what happens when this job is submitted for processing.



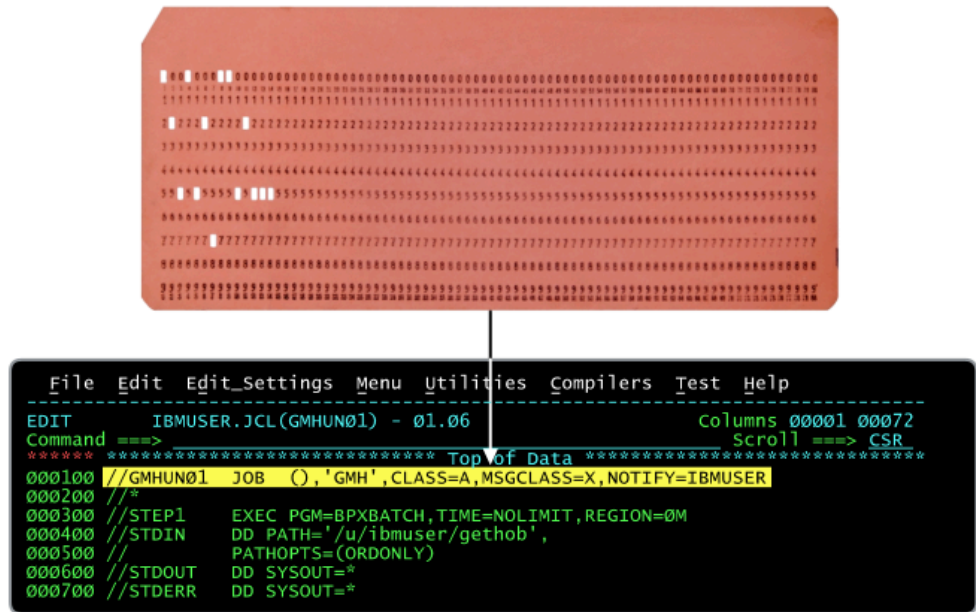


```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(A#UX01) - 01.01                Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000001 //A#UX01  JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER,REGION=0M
000002 //STEP1  EXEC PGM=IEFBR14
000003 //DDL    DD PATH='/u/ibmuser/account2',
000004 //          FILEDATA=BINARY,
000005 //          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH),
000006 //          PATHDISP=(KEEP,DELETE),
000007 //          PATHOPTS=(OCREAT,ORDWR)
```

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(SUPAPD05) - 01.11             Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000100 //SUPAPD05 JOB T112-345,'Eliza Brown',
000200 //          CLASS=A,MSGCLASS=X
000300 //          *
000400 //STEP1  EXEC PGM=IEBGENER
000500 //SYSUT1  DD DSN=PROD.DAILY.PAYMENTS,DISP=OLD
```

As you will see throughout these courses there will be some exceptions to many of the JCL rules. In relation to the use of case in JCL, there may be situations where lowercase characters are required. To facilitate this, you should change your profile to CAPS OFF and ensure that any lowercase characters are enclosed in single quotes.

In the top example shown here, a z/OS UNIX file is being referenced, and this needs to appear in lowercase. The example at the bottom of this page shows a name, using both uppercase and lowercase characters, that is enclosed in single quotes. This JOB statement parameter as you will see later allows you to specify a programmer's name to be linked to the job.



You may remember in the previous module that the partitioned data set you are using to store your JCL should have a record length of 80. This originates from punch cards that were used in the early days of computing, where these cards contained 80 columns.

Columns 73-80 of the punch cards, and also these columns in your displayed job, are ignored when submitted. They were traditionally used for sequence numbers.

Click Play to see what happens if you code JCL in these columns.

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(APDSE2) - 01.01                      Columns 00001 00072
Command ==>                                     Scroll ==> CSR
=COLS> ---+---1---+---2---+---3---+---4---+---5---+---6---+---7---
***** Top of Data *****
000100 //APDSE2  JOB MSGCLASS=X,CLASS=C
000200 /*
000300 //CREATE  EXEC PGM=IEFB14
000400 //PDSE2   DD DSN=IBMUSER.PDSE.GENS,
000500          DSNTYPE=(LIBRARY,2),MAXGENS=10,
000600          RECFM=FB,LRECL=80,
000700          UNIT=SYSALLDA,SPACE=(CYL,(1,1,1)),
000800          DISP=(,CATLG,DELETE)
***** Bottom of Data *****

```

Commas appear at the end of each line indicating to the system that the line that follows contains continuation information for that statement.

No comma is required here, signifying the end of information for this statement.

In this example, the continuation appears in column 12.

Click Next to see what happens if you do not continue a statement successfully.

Next ►

Since you cannot extend your JCL statement past column 72, how do you handle a JCL statement that contains lots of information? To continue a statement, you code a comma at the end of the parameter being specified on that line, and on the following line the double slash (//) characters must be in columns 1 and 2, and your continued information can appear anywhere between column 4 and 16 (inclusive).

Often you will see for readability purposes that continued line data is aligned with previous lines.



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT          IBMUSER.JCL(AGENER) - 01.05          columns 00001 00072
Command ==> _____ scroll ==> CSR
***** Top of Data *****
000100 //AGENER JOB MSGCLASS=C,MSGLEVEL=(1,1)
000200 //* STEP 1 - TEST GDG BASED CONCATENATION
000300 //STEP1 EXEC PGM=IEBGEN
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=IBMUSER.PEN.DAILY.TRANS(0),DISP=SHR
000600 //SYSUT2 DD SYSOUT=*
000700 //SYSIN DD DUMMY
000800 // IF STEP1.ABEND=TRUE THEN
000900 //STEP2 EXEC PGM=IKJEFT01,REGION=0K
001000 //SYSTSPRT DD SYSOUT=*
001100 //SYSTSIN DD *
001200 SEND 'JOB A#PENXT HAS ABENDED' -
001300 USER(IBMUSER,GREG) NOW
001400 /*
001500 // ELSE
001600 //STEP2 EXEC PGM=IKJEFT01,REGION=0K
001700 //SYSTSPRT DD SYSOUT=*
001800 //SYSTSIN DD *
```

Step 2 of 2

The HILITE command has identified that the code is JCL and is able to provide coloring to differentiate components. For example, comments and raw data appear as turquoise, while statement types are shown in red. The remainder of this course will use this highlighting feature when working with JCL code.





JOB Attributes the system will use when processing the batch job.

EXEC To specify programs that need to be run.

DD For identifying input and output requirements for the program.



Now that you have much of the JCL formatting rules covered, it is time to take a quick look at the general syntax of information displayed in JCL statements. Do not worry, as all this information covered in more detail in later courses.



This name contains seven alpha characters.

These statements contain a number as part of their name.

```

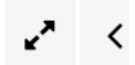
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.00          Columns 00001 00072
Command ==>                               Scroll ==> CSR
***** Top of Data *****
000100 //FAMXTGH JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER
000200 //
000300 //STEP1 EXEC PGM=ICEGENER
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=GHMAST.FAMAPS.D256,DISP=OLD
000600 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP,
000700 //          DISP=(,CATLG)
000800 //SYSIN DD DUMMY
***** Bottom of Data *****

```

Incorrect statement names		
	1STSTEP	Begins with a number ✗
	MYPROGSTEP	Too many characters ✗
	&BACKUP	Ampersand (&) is not a national or alpha character ✗

When creating JCL, you will need to tell the system the type of statement you are providing, and in most cases, provide a name for that statement. The name you provide for each statement is c to eight characters and appears immediately after the double slash (//) characters. This name must start with an alpha or national character (\$, #, @), while remaining characters can also contain numbers.

As you will see later, depending on the statement type, this name could be one of your choice, or may need to be specific.





Currently, this job contains a single JOB and EXEC statement, and several DD statements.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER,JCL(FAMXTGH) - 01.00                      Columns 00001 00072
Command ==>                                         Scroll ==> CSR
*****
***** Top of Data *****
000100 //FAMXTGH JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER
000200 //*
000300 //STEP1 EXEC PGM=ICEGENER
000400 //SYSPRIN DD SYSOUT=*
000500 //SYSUT1 DD DSN=GHMAST.FAMAPS.D256,DISP=OLD
000600 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP,
000700 //          DISP=(,CATLG)
000800 //SYSIN DD DUMMY
*****
***** Bottom of Data *****
```

Four blank spaces have been coded between this statement name and the statement type.

Associated with the name is the statement type. Common statements used include JOB, EXEC, and DD, which are discussed in detail in the courses that follow. At least one space must be coded following the name before this statement type is entered.

If you are working on existing JCL you will often see several spaces between the name and the statement type. This is for readability purposes as it aligns important information.



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.02                Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000100 //FAMXTGH JOB MSGCLASS=C,MSGLEVEL=(1,1),NOTIFY=IBMUSER
000200 //*
000300 //STEP1 EXEC PGM=ICEGENER
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=GIMAST.FAMAFS.D256,DISP=OLD
000600 //SYSUT2 DD DSN=GIMAST.FAMAFS.D256.BACKUP,
000700 //DISP=(,CATLG)
000800 //SYSIN DD DUMMY
***** Bottom of Data *****
```

There are three parameters shown on the JOB statement, each with a value, and separated by a comma. Where a parameter value consists of multiple items, like MSGLEVEL, they are usually enclosed in parenthesis.

Some organizational standards dictate that only a single parameter be displayed on a line, for readability purposes. In this example, the DISP parameter on line 000700 could have fitted onto the end of the text on line 000600 and would also be syntactically correct.

Click Next to see what happens if you do not spell a parameter name properly, or do not separate them with commas.

Next ▶

Every statement will have some parameters that describe requirements, or attributes, to be associated with that statement. There may be many associated with that statement, though in reality you are only likely to use a subset of them regularly. If parameters are specified, at least one space must follow the statement type - JOB, EXEC, or DD - before they are entered.

Where there are multiple parameters for a statement, they must be separated by commas, and if the parameter itself contains a space, it needs to be enclosed in single quotes.



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(AJOB) - 01.08                      Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000100 //IBMPRD1 JOB MSGCLASS=X,CLASS=C NOTIFY=IBMUSER
000200 //*
000300 //CLEANUP EXEC PGM=IDCAMS
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSIN DD *
000600 DELETE IBMUSER.JCL.EXEC(ACOP)
***** Bottom of Data *****
```

In this example, a space was left between the CLASS and NOTIFY parameters on line 000100, rather than a comma. The system interprets the information following the space as a comment. Note that in this instance the HILITE coloring assists you in identifying this issue, as NOTIFY and its value are displayed in turquoise.

◀ Previous Restart

Every statement will have some parameters that describe requirements, or attributes, to be associated with that statement. There may be many associated with that statement, though in reality you are only likely to use a subset of them regularly. If parameters are specified, at least one space must follow the statement type - JOB, EXEC, or DD - before they are entered.

Where there are multiple parameters for a statement, they must be separated by commas, and if the parameter itself contains a space, it needs to be enclosed in single quotes.



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.03                      Columns 00001 00072
Command ==>                                                Scroll ==> CSR
***** ***** Top of Data *****
000100 //FAMXTGH JOB DEPT-118,'GREG HAML'YN',
000200 //          CLASS=K,
000300 //          MSGCLASS=X,
000400 //          NOTIFY=IBMUSER
000500 //          *
000600 //STEP1 EXEC PGM=ICEGENER
000700 //SYSPRINT DD SYSOUT=*
000800 //SYSUT1 DD DSN=GMAST.FAMAPS.D256,DISP=OLD
000900 //SYSUT2 DD DSN=GMAST.FAMAPS.D256.BACKUP,
001000 //          DISP=(,CATLG)
001100 //SYSIN DD DUMMY
***** ***** Bottom of Data *****
```

If this accounting information was not required, but the programmers name was, then a comma would need to be coded to signify that the accounting information was being bypassed.

Even though this is a positional parameter, if it is not required and the accounting information is, then you do not need to code a comma to indicate its absence. Note that in this example, because the value contains a space, it needs to be encased in single quotes.

The parameters for each statement are separated into positional and keyword.

If used, a positional parameter must appear in a specific area of the code, and if it is not required, a comma is often used to indicate it being bypassed, although this is not always the case. In this example some parameters have been added to the JOB statement. These are discussed in detail in a later course, but for the purpose of this exercise the department accounting information and programmers name that appear on line 000100 are both positional parameters.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.04          Columns 00001 00072
Command ==>                               Scroll ==> CSR
***** Top of Data *****
000100 //FAMXTGH JOB 'GREG HAMLYN',
000200 //          CLASS=k,
000300 //          MSGCLASS=X,
000400 //          NOTIFY=IBMUSER
000500 //*
000600 //STEP1 EXEC PGM=ICEGENER
000700 //SYSPRINT DD SYSOUT=*
000800 //SYSUT1 DD DISP=OLD, DSN=GHMAST.FAMAPS.D256
000900 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP, DISP=(,CATLG)
001000 //SYSIN DD DUMMY
***** Bottom of Data *****
```

Keyword parameters are more common and can appear in any order within the statement, following the statement type. Their name is followed by an equals (=) sign and then the value assigned to that keyword parameter.

In this example, line 000800 shows a DISP parameter first, and then a DSN parameter. On the line after this, these two parameters appear in the opposite order. As DISP and DSN are keyword parameters this coding is acceptable.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      IBMUSER.JCL(FAMXTGH) - 01.04                Columns 00001 00072
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
000100 //FAMXTGH JOB 'GREG HAMLYN',
000200 //          CLASS=K,
000300 //          MSGCLASS=X,
000400 //          NOTIFY=IBMUSER
000500 //
000600 /* THE FIRST STEP IS USED FOR RECOVERY PURPOSES
000700 /* CHANGE THE SYSUT1 AND SYSUT2 DATA SETS AS REQUIRED
000800 /*
000900 //STEP1 EXEC PGM=ICEGENER
001000 //SYSPRINT DD SYSOUT=*
001100 //SYSUT1 DD DISP=OLD,DSN=GHMAST.FAMAPS.D256 * INPUT DATA SET *
001200 //SYSUT2 DD DSN=GHMAST.FAMAPS.D256.BACKUP,DISP=(,CATLG)
001300 //SYSIN DD DUMMY
***** Bottom of Data *****
```

Note that comments are not interpreted by the system but are simply accepted as they are coded.

As with any coding exercise, it is always best practice to include comments, so that if another person needs to use the JCL, they are made aware of what its purpose is, and what changes they will need to make before using it. Sometimes a comment is also used to break up the JCL code, so it is easier to read.

There are two ways of coding comments in JCL. The more common method is to code a `/*` statement such as on lines 000500 to 000800. Any text that then appears after this is considered a comment. Another method is to leave at least one space at the end of a line and type your comment, such as on the end of line 001100.



```
Menu  Functions  Confirm  Utilities  Help
-----
EDIT  IBMUSER.JCL                               Row 0000040 of 0000067
Command ==>                                     Scroll ==> CSR
SUB   Name      Prompt      Size  Created      Changed      ID
-----
SUB   FAMXTGH   HOBGET      13    2017/10/25   2017/10/27   00:56:44   IBMUSER
-----
SUB   IBMJOB    IBMJOB      35    2016/08/28   2017/10/26   08:26:42   IBMUSER
-----
SUB   IBMJOB2   IBMJOB2    36    2017/09/05   2017/09/05   00:09:32   IBMUSER
-----
SUB   IBMJOB3   IBMJOB3     9     2017/09/05   2017/10/26   08:27:51   IBMUSER
-----
SUB   IDCAMS1   IDCAMS1    13    2017/09/17   2017/10/26   08:35:18   IBMUSER
-----
SUB   IDCAMS2   IDCAMS2    18    2017/09/17   2017/09/17   23:50:22   IBMUSER
-----
SUB   IDCAMS3   IDCAMS3    18    2017/09/17   2017/09/17   22:08:30   IBMUSER
-----
SUB   IDCAMS4   IDCAMS4    14    2017/09/17   2017/09/18   00:33:30   IBMUSER
-----
SUB   IDCAMS5   IDCAMS5    17    2017/09/18   2017/09/18   03:11:18   IBMUSER
-----
SUB   IDCAMS6   IDCAMS6    11    2017/09/18   2017/09/20   23:59:14   IBMUSER
-----
SUB   IGGCSILC  IGGCSILC
-----
SUB   IGGCSIRX  IGGCSIRX
-----
SUB   IGGCSIVG  IGGCSIVG
-----
SUB   IGGCSIVS  IGGCSIVS
-----
SUB   MASTCOP   MASTCOP     8     2017/08/03   2017/10/25   18:27:48   IBMUSER
-----
SUB   OOCBKUP   OOCBKUP    54    2016/08/08   2016/08/08   20:14:12   IBMUSER
-----
SUB   PLICOMP   PLICOMP    32    2016/08/14   2017/10/04   23:26:01   IBMUSER
```

This screen is a list of PDS members accessed using the ISPF 3.4 option. If no editing is required to the JCL, then a SUB, or SUBMIT, command can be entered directly against the PDS member.

◀ Previous Next ▶

Although you have a long way to go before submitting your own batch job, it is an opportune time to look at the various methods used to perform this task.

The most common method is for you to type SUBMIT, or SUB, into the ISPF edit command line shown here.



• A maximum of eight characters can be specified for a statement name.

• A statement name needs to begin with an alpha or national character.

• There are two types of parameters that can be specified on a JCL statement – positional and keyword.

• There are two different ways of coding a comment in JCL.

• The SUBMIT command is used to send the JCL to the operating system.

In this section you have seen how specific you sometimes need to be when coding JCL. A simple missing comma or a misspelled parameter can prevent your job from running successfully or running at all. Having said that, many syntax-related errors can be quick to identify and resolve.

The courses that follow this will provide a more in-depth look at the major JCL statements, and offer you many opportunities to code your own JCL statements.



Summary

JCL Coding Requirements

Before beginning to write any JCL code, you need to become familiar with the formatting of JCL statements, and the consequences should these not be followed. In this module you looked at the general syntax and format of JCL statements.

You should now be able to:

- Describe the Formatting Requirements of a JCL Statement
- Identify Common JCL Coding Errors