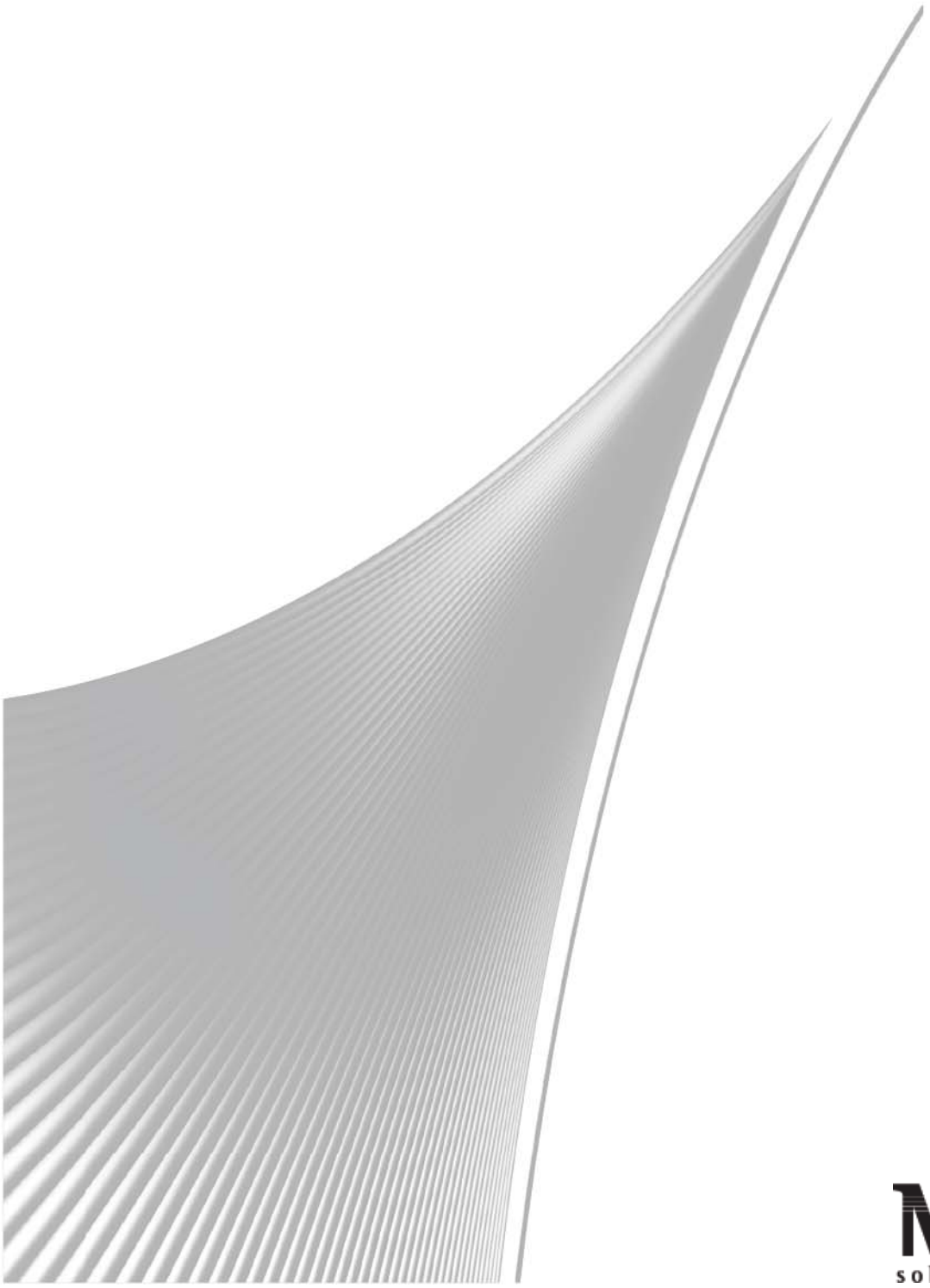
ThruPut Manager

Dataset Contention Services (DCS)

System Programming Guide



MVS solutions inc.

This manual applies to ThruPut Manager® Version 7 Release 1.0 PTF level 7109 and to all further releases until otherwise indicated. This material contains proprietary information on ThruPut Manager. Use of this manual is subject to the terms of the user's software license agreement for ThruPut Manager.

The complete list of manuals relevant to this release is available from the Customer Center on www.mvssol.com, from where any of the manuals may be downloaded.

The Acrobat® PDF Format

This set of manuals has been formatted to make it suitable for soft copy as well as hard copy formats. If reading the manual online, note that:

- Entries in the Table of Contents are hot-linked and can be used for document navigation.
- Other cross-references appear in blue type and are also hot-linked for easy navigation.

For best results, turn on the Bookmarks feature in your Acrobat® PDF reader.

Preface

About This Manual

This manual provides information for systems programmers involved in planning, installing, and customizing the Drive Booking Services (DBS). There are five system programming manuals as listed below. Information described in other manuals is referred to by guide name and chapter as a quick and easy cross reference.

- Base Product System Programming Guide
- Drive Booking Services (DBS) System Programming Guide
- Job Binding Services (JBS) System Programming Guide
- User Control Services (UCS) System Programming Guide
- Exits System Programming Guide

Summary of Changes

V7R1-7109

(April 2017)

- No changes.

V7R1-7108

(February 2017)

- No changes.

V7R1-7107

(May 2016)

- No changes.

V7R1-7106

(November 2015)

- No changes.

V7R1-7104

(July 2015)

- No changes.

V7R1-7101

(July 2014)

- This is a base manual for ThruPut Manager Version 7 Release 1.1.

Table of Contents

Dataset Contention Services

Chapter 1. The TM/DCS Function—An Optional Playground

This chapter describes the functions provided by the Component TM/Dataset Contention Services.

The TM/Dataset Contention Services Component

ThruPut Manager offers a Component that is installed on top of the ThruPut Manager Base Product. The Component is named "TM/Dataset Contention Services", or TM/DCS for short. It provides a complete solution to the automation of the dataset contention resolution process. This chapter introduces the different functions provided by this package.

Functions Provided With TM/DCS

This Component provides the following additional functions:

1. Job Action Language extensions for DCS.
2. DAL—A Detail Action Language particularly suited for DD/dataset processing.
3. Dataset Contention and Repossessing Services.
4. ALERT Services.
5. NAGGING Services.
6. Dataset Disposition ALTER Services.
    a. Dataset Reservation
7. JECL Service Extensions.
8. Dataset Contention Reports.
9. User Exit.
10. Addition of a DCS Display to the User Display Facility.
11. File Definition Services Extensions.

Job Action Language (JAL) Extensions for DCS

Extensions to JAL are provided to support DCS. They allow you to instruct DCS for job related actions. For specific DD/dataset requests, DAL is used.

A detailed description of the JAL extensions is found in the ***DAL/JAL User Guide*** and ***JAL Reference*** publications.

Detail Action Language—DAL

A Detail Action Language, similar to JAL, is provided to define rules and request DCS actions at the DD/dataset level.

A complete set of supporting facilities to load, refresh, and trace DAL is also provided.

A detailed description of DAL is provided in the publications ***DAL/JAL User Guide*** and ***DAL Reference***.

Dataset Contention and Repossessing Services

Complete automation facilities are provided to help you manage jobs that must be delayed because of dataset contention. The DCS application provides:

- Complete automation of the holding and releasing of jobs.
- A separate hold reason for jobs awaiting datasets.
- An implementation of the concept of dataset service level. This allows you to control which jobs are to receive priority when datasets become available.
- Facilities to repossess datasets automatically from TSO users.
- Operator Commands.
- JAL/DAL interfaces.
- JECL interfaces.

With the above facilities, you have both automation and control of your dataset contention resolution process.

☞ ***Note: If DCS is installed and enabled in your system, all jobs are managed for dataset contention, including any jobs submitted in an Exempt class.***

A detailed description of the dataset service level and repossessing facilities is included in "Chapter 3. Requesting Dataset Service Levels."

The Operator commands are described in the ***Operating Guide***.

Nagging Services

DCS provides a mechanism to nag TSO users that are holding datasets needed for a job. This facility is called—what else?—NAG services. The following controls are provided:

- You can modify the second line of the NAG to reflect whatever text you want.
- You can decide which user or group of users are nagged.
- You can also suppress nagging to a user or users.
- You can request the nagging frequency.
- You can indicate how many times users are nagged.

You can combine any of the above options to fine-tune nagging to cover practically any situation. A detailed description of Nagging Services is found in "Chapter 4. Nagging."

ALERT Services

The ALERT Service provides you with a facility to notify a group, such as Production Control, when a job has been held for more than a specified number of minutes because of dataset contention.

DCS does not produce ALERTs unless it is instructed to do so. You can control:

- Whether or not an ALERT is generated for a job.
- Header and Trailer text for ALERTs.
- The interval time before the ALERT is produced.
- The destination of an ALERT.

The mechanism to produce, spool, and print ALERTs is the same mechanism ThruPut Manager uses for Volume Lists.

A detailed description of ALERT Services is found in "Chapter 5. Alerting."

Dataset Disposition Alteration Services

With DCS, you can analyze the usage of the first disposition parameter for each DD statement that is associated with a permanent dataset. Depending on what you find, you can instruct DCS to take one of the following actions:

- Fail/warn the job.
- Alter the disposition for that DD statement.

The disposition alteration is for dataset enqueueing purposes.

Requests to DCS to alter the disposition are done at the DD statement level. All the relevant information about the job and the particular DD statement is available to you. As a result, you can provide very specific rules for the ALTERation of the first parameter of the DISP keyword.

A detailed description of Dataset Disposition Alteration Services is found in "Chapter 6. Altering Dataset Disposition."

Dataset Reservation

DCS can eliminate certain job failures and volume contention problems by ENQing during job initiation on the real dataset name for GDGs and datasets referred to by aliases. At that time, if there is a conflict the job can be requeued. Once Dataset Reservation support is activated, it automatically applies to all jobs. If necessary, you can turn off this service for individual jobs through the JAL statement DCS SET.

For more discussion of Dataset Reservation and how to implement it, see "Chapter 6. Altering Dataset Disposition."

JECL Services Extensions

DCS can also be instructed with JECL statements. JECL services give your installation the opportunity to "distribute" DCS facilities to your users.

It is important to realize that your installation does not lose control by "outboarding" these facilities. JAL and DAL allow you to determine whether or not a job has DCS control statements and the type of statement. As a result, you can allow job submitters to use as much or as little functionality as desired. If the job submitter is not allowed to use a particular capability, you can generate an informative message and fail the job.

The JECL facilities available are documented in the ***JECL Reference Guide***

DCS Reporting

DCS collects all the necessary information to be able to produce, on request, two types of reports:

- One report gives you information on jobs that were delayed. It shows how many times the job was delayed, the number of minutes that the job was delayed waiting for datasets, and the datasets that were involved.
- The other report gives you information on datasets that cause contention problems. The number of minutes that the dataset was in contention are shown, together with the jobs affected and the holders of the dataset.

Both reports together give you a complete picture of the problem: the jobs that were affected, the datasets that were involved, and the holders of the datasets.

The reporting system allows selection criteria. For example, you can request that only jobs that "experience a 20-minute delay or more" be shown.

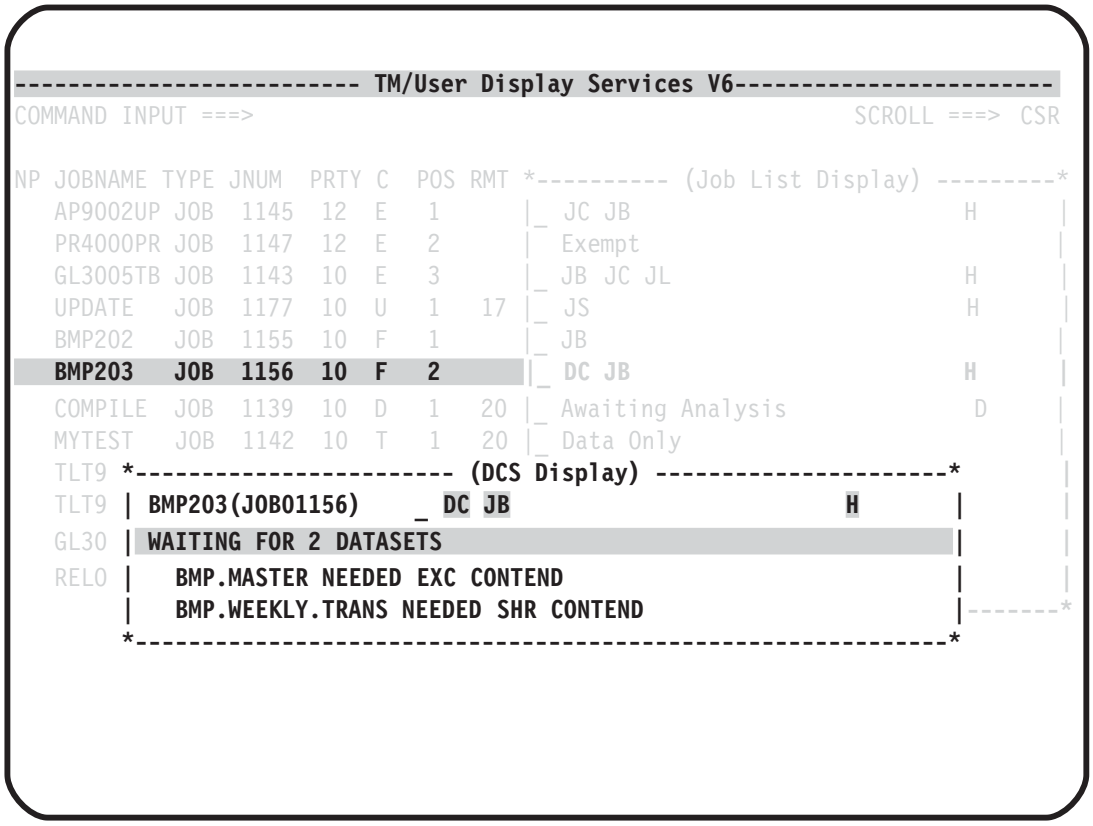
For DCS to collect data in contention situations, you must:

- Request recording for the job in JAL.
- Activate the recording file with a TMSS initialization statement.

The Dataset Contention Reporting Facilities are documented in "Chapter 8. The Dataset Contention Reporting System."

UDF Extensions For DCS

If the User Display Facility shows the DCS acronym for a job, a user can select that acronym and open the DCS Window. A sample DCS Window is shown here.



The DCS Display Window

The first line of the DCS Window is the Information Summary Line for the job.

Subsequent lines describe each dataset that the job requires but cannot yet allocate because of dataset contention. The dataset name is shown, accompanied by the level of service requested.

User Exit

A job analyzer exit is provided to inspect JECL statements. This exit is Exit 19 and is fully documented in the ***Exits: System Programming Guide*** publication.

File Definition Services Extensions

DCS needs one mandatory file, and depending on the options, two other files:

- The Control File. This file is mandatory.
- The SPOOL File. This file is needed if you want to generate ALERTs using Console Printing Services.
- The CMF File. This file is needed if you want DCS to generate management reporting data when dataset contention occurs.

These files are documented in ***Base Product: System Programming Guide*** "Chapter 3. File Definition Services (FDS) Function."

Chapter 2. TM/DCS—The Easy Way

This chapter discusses the initial implementation of TM/DCS and the steps to take after it is running successfully in your installation.

Introduction

The TM/DCS Component provides you with all the facilities necessary to exercise complete control in the automation of the dataset contention resolution process.

DCS capabilities can be categorized into two related but distinct categories:

1. Automation.
2. Control.

To get the benefits of DCS automation you do not have to resort to any customization. We advise you to initially implement the TM/DCS Component the "easy way". Later on, you can start exercising control by fine-tuning the product to reflect your particular installation needs. This chapter describes the behavior of DCS when it is allowed to do "its own thing".

Automating Dataset Contention Resolution

A straight implementation of DCS, without providing any installation "guidance", gives you superior automation.

In fact, if you leave DCS alone to do "its own thing", it eliminates ***all the problems*** described in the ***TM/DCS Concepts and Facilities*** publication under the heading "System Pollution". This is possible because DCS assumes control of the process to determine whether or not all the datasets are available for a job.

A Self-Employed DCS

DCS allows the system to do all the hard work in determining the datasets needed, the last step in the job that references them, and the type of control to request for each dataset.

Once the system has completed that task, DCS takes over. It makes the request to ENQ/DEQ services on behalf of the initiator. This request is made as early as possible in the job initiation process. It is done before IEFUJI is invoked.

If the datasets are available, then DCS gets out of the way so work can proceed in the usual manner, but with the assurance that all datasets have already been secured. By the time IEFUJI is executed, the potential for the job being requeued no longer exists.

If the datasets are not available, DCS produces a terse message indicating that job initiation cannot proceed because of dataset contention. The job is then ***quietly*** requeued and placed in a special TM HOLD for jobs experiencing dataset contention. All the SMF exits, SMF records, requeuing messages, etc., are eliminated.

As you would expect, DCS monitors the availability of datasets to determine when the job should be released. When the dataset, or datasets, become available, DCS proceeds to release the job ***from its DCS HOLD***. This technique totally eliminates the potential conflict with a JES2 HOLD that might have been issued while the job was waiting for datasets.

So a DCS operating under its own rules gives complete automation of the dataset conflict resolution and also eliminates all "system noise":

- It eliminates all superfluous messages. No clutter.
- It prevents IEFUJI from executing. Your job schedulers or in-house software do not have to undo, at job termination, whatever IEFUJI may have done at job initiation.
- Jobs are placed in a special dataset contention HOLD, completely eliminating mix-ups with JES2 HOLD.

DCS also gives notification to TSO users that are holding datasets needed by jobs or started tasks. This is referred to as "Nagging".

DCS Nagging

Without any installation guidance, DCS takes it upon itself to Nag TSO dataset holders. The NAG is a two line message with the following format:

```
DTM7108I  PLEASE FREE DATASET dataset-name time
DTM7109I  THIS DATASET IS REQUIRED BY jobnumber jobname
```

The holder is notified 3 times 5 minutes apart if a batch job needs the dataset.

Installation Summary

For DCS to fully automate the dataset contention process, ***you do not have to do anything*** beyond your product installation.

DCS will operate under the following defaults:

- Automatic requeuing of jobs in DCS HOLD when contention occurs.
- Automatic releasing of jobs when datasets become available.
- STANDBY as the dataset service level.
- 3 Nagging Messages to TSO users, 5 minutes apart when batch jobs need a dataset.
- 99 Nagging messages to TSO users, 1 minute apart when started tasks need a dataset.
- Automatic freeing of datasets when the TSO holder of the dataset is the job submitter. This only takes place if ***the dataset is allocated but not in use***.

What To Do Next

Once you have DCS taking care of your dataset contention management "in its own way," you might want to learn more by making use of its reporting capabilities to see which situations can be improved. The next sections present a brief discussion of the reporting facilities available, followed by a description of how to implement your own DCS rules.

DCS Reporting

From the reports (or simply because you already know) you might find a number of situations that require attention. To DCS, unless otherwise instructed, "all jobs are equal." From your position of knowledge, you know that "some jobs are more equal than others," therefore, they should be getting preferential treatment. You might also find that some datasets are causing contention because they are being requested for exclusive control by some jobs for which you know SHR will do. It is also possible that some TSO users are holding datasets when they should not. You probably can think of many other situations that need remedy.

So how can you remedy these situations? You would like DCS to be a bit less independent and become more accommodating to your specific needs. You want to convert DCS from being self-employed to being your faithful servant. Here is where DCS truly excels.

DCS can be instructed to handle dataset contention situations in a way which precisely reflect your needs, no matter how detailed and specific they are.

DCS collects all the necessary information to be able to produce, upon request, two types of reports:

- One report gives you information on jobs that were delayed. It shows how many times the job was delayed, the number of minutes that the job was delayed waiting for datasets, and the datasets that were involved.
- The other report gives you information on datasets that cause contention problems. The number of minutes that the dataset was in contention are shown, together with the jobs affected and the holders of the dataset.

The two reports together provide you with a complete picture of the problem: the jobs that were affected, the datasets that were involved, and the holders of the datasets.

Activating the reporting system should probably be your next task after you have DCS running. This is covered in "Chapter 8. The Dataset Contention Reporting System."

Implementing Your Own DCS Rules

After you have determined what actions you would like DCS to take when dataset contention occurs, you can code DAL to implement your rules. See the ***DAL/JAL User Guide*** and the ***DAL Reference Guide*** for details of how to code DAL.

DAL source statements must be converted into a format that ThruPut Manager can use. The Language Processor is supplied for this purpose.

Your DAL source statements are the input to the Language Processor, and the output is DAL internal text, suitable for use by ThruPut Manager. For complete details about the JCL and usage of the Language Processor, see the ***DAL/JAL User Guide***.

The sequence required to implement DAL is:

1. Write DAL statements defining your installation's rules.
2. Run the Language Processor to verify your DAL source statements, and convert them into the internal text required by ThruPut Manager.
3. Add the statement DAL LOAD to the TMSS initialization statements. This indicates that there are installation rules and where they can be found (DAL internal text).

   ☞ ***To load a new DAL does not require that you start TMSS again. The DAL REFRESH command is provided to allow you to refresh DAL dynamically.***

If your DAL does not work as intended, you can use the DAL trace facility to help you debug the problem. DAL tracing can be turned on at startup by using the DAL TRACE initialization statement, or after startup by using the DAL TRACE operator command.

You can find more information about loading and tracing DAL in the manual ***DAL/JAL User Guide***.

The chapters that follow describe the facilities that make DCS a "precision tool" to suit your installation requirements.

Facilities Summary

DAL Initialization Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
DAL LOAD	Specifies the dataset name and member name (if applicable) where the DAL internal text is to be loaded from.	2
DAL TRACE	Allows you to request DAL trace at TMSS start up time. You can trace specific jobs or all jobs.	2

Chapter 3. Requesting Dataset Service Levels

This chapter shows you how to use the various facilities of DCS to request dataset service levels to meet your batch service objectives.

Introduction

One of the key facilities provided with DCS is the ability to associate service levels with any or all the datasets in a job. This chapter shows you how to use the various facilities of DCS to request dataset services levels for jobs, so their service objectives can be met.

Before discussing how to request service levels for datasets, we need a detailed discussion on how the different levels work. This is the subject of the next section.

Dataset Service Levels—How They Work

Three different dataset service levels are provided:

1. CLAIM
2. CONTEND
3. STANDBY

They are listed by priority, that is, CLAIM is the highest level and STANDBY is the lowest. They take effect whenever a job runs into dataset contention.

Service Level Rules

This section provides the rules for each level of service. The rules are first introduced for situations where a job runs into contention for only one dataset. Contention situations where several datasets are not available for a job are discussed later.

CLAIM Service

CLAIM is the highest dataset service level. The rules are as follows:

- The job is allowed to wait in the initiator. This ensures that when the dataset becomes available, ***this job*** will get it.
- For a particular dataset, ***only one job for the entire DCS complex*** is allowed to wait in the initiator. This prevents the "clogging" of initiators.
- If another job with CLAIM service for the same dataset is selected for execution, DCS detects the situation and places the job in a high priority wait queue, in DCS HOLD.
- When the dataset becomes available:
  - The job that was waiting in the initiator is allowed to proceed to execution.
  - The next job (by JES2 rules) with CLAIM service for that dataset is then allowed to wait in the initiator, assuming there is an initiator available.
- Dataset repossessing can be requested.

The effect of the above rules is that jobs with CLAIM service for a dataset take precedence over CONTEND and STANDBY. Jobs with CLAIM services for the same dataset follow the normal system rules to resolve which one is next.

The dataset repossessing services available to CLAIM are explained later in this chapter under the heading "Repossessing Datasets."

You should not use this service level as the default level. This is your priority mechanism.

CONTEND Service

CONTEND is the next level of service, and as its name indicates, allows jobs to contend with an equal chance (from a dataset point of view). The rules are as follows:

- The jobs are not allowed to wait in an initiator.
- They are requeued in DCS HOLD status.
- DCS makes a dataset enqueue request on their behalf.
- When the dataset becomes available, DCS knows it immediately. All the jobs that requested CONTEND services, and were placed in DCS HOLD, are released at the same time (equal chance within CONTEND requests). So, which job gets the dataset is determined by the normal system rules for job selection.

- There are some implications for CONTEND requests in situations where:
  - The dataset ***is in use with the attribute SHARED***.
  - Dataset contention occurs for a dataset requesting EXCLUSIVE control.
  - The dataset requests CONTEND service.

  Because DCS places an EXCLUSIVE dataset enqueue request on behalf of the job, no other jobs can get the dataset even though they may need it as SHARED. This is a characteristic of the CONTEND service to give EXCLUSIVE requests ***an equal chance*** to compete with SHARED requests. Note that since all jobs "contend" with an equal chance, there is no guarantee that the job with exclusive requirements will get the dataset.

  There may be some situations where you want to use this level of service as the default value. *In general this is not advisable*, so unless you have very well understood reasons, do not use CONTEND as the default service level.

STANDBY Services

STANDBY is the lowest service level and should be used as the default. The rules are as follows:

- The jobs are not allowed to wait in an initiator.
- They are requeued in DCS HOLD status.
- For SHARED requests, DCS places a dataset enqueue on their behalf. For EXCLUSIVE requests, DCS uses a sampling technique.
- The difference between this service and the CONTEND service is in the following situation:
  - The dataset ***is in use with the attribute SHARED***.
  - Dataset contention occurs for a dataset requesting EXCLUSIVE control.
  - DCS does not make an EXCLUSIVE enqueue request on behalf of the job (it does sampling). As a result, any other job that requests the dataset as SHARED is allowed to proceed.
  - The job requesting the dataset as EXCLUSIVE is allowed to proceed to execution only when there are no SHARED requests (hence its name STANDBY).

This service "favors" SHARED requests.

Dataset Service Level—Job and Dataset Request

Service is not an attribute of a job. ***It is an attribute of a dataset within the context of a job.***

Facilities exist in JAL and JECL to set the dataset service level for a job. This might appear to contradict the above assertion, but in reality it does not. Setting the service level for a job is a short form for ***setting the service level for all the datasets in the job***.

Facilities are also provided, in DAL and JECL, to set the service level for each dataset individually. This means that we can have a situation where more than one dataset is not available for a job, and they have different service levels. What happens then?

CLAIM versus CONTEND/STANDBY

A job is treated as CLAIM if, and only if, all the datasets that are not available have CLAIM service associated with them. If any of the datasets that are not available have CONTEND or STANDBY service, the job is not given CLAIM service.

CONTEND versus STANDBY

In this situation, each dataset is monitored according to its service level. All CONTEND datasets, and STANDBY for SHARED control, result in an enqueue request on behalf of the job. STANDBY for EXCLUSIVE control triggers the sampling technique. In either case the jobs are held by DCS. Once all the datasets that caused the original contention become available (regardless of their service level) the job is released.

Releasing Jobs—Which One First?

DCS releases jobs in groups so that the chances that they are selected by JES2 for execution reflect the dataset service level. The grouping and order of release are as follows:

1. Jobs that were waiting for datasets with CLAIM service only.
2. Jobs that were waiting for datasets with CLAIM service mixed with other services.
3. Jobs that are waiting for datasets with CONTEND service only.
4. Jobs that are waiting for datasets with CONTEND and STANDBY service.
5. Jobs that are waiting for datasets with STANDBY service only.

The groups are released one at a time, in the above order. Within each group, jobs are released FIFO by contention time.

Dataset Service Level Aging

DCS provides you with the ability to "age" the initial service level requested for a dataset. When you make a service level request for a dataset, you can indicate that after a certain interval has passed, the service level is to be modified. You are allowed to upgrade or downgrade the service level.

The way you communicate to DCS service level modifications is with an ***aging service list***. For example:

```
DCS SERVICE (STANDBY(20),CONTEND(30),CLAIM)
```

The above statement says to DCS:

- Assign an ***initial service*** level of STANDBY
- If after 20 minutes waiting for the dataset, you are unable to get it, upgrade the service to CONTEND.
- Again, if another 30 minutes pass, and the dataset is not available yet, upgrade the service to CLAIM.
- CLAIM is to be treated as the final service level, since it is the last element in the list. For the final service level, you do not specify an interval.

When an interval expires, DCS manages all the actions required for the transition from one service level to another.

The service transition actions of DCS are as follows:

- For a STANDBY to CONTEND transition, the monitoring technique is adjusted if necessary (from sampling to enqueue pending).
- For a CONTEND to STANDBY transition, the monitoring technique is adjusted if necessary (from enqueue pending to sampling).
- For a STANDBY to CLAIM transition, if no other job is in CLAIM mode for the dataset, the job is released so it can be selected and wait in an initiator. If another job is already waiting in an initiator, the monitoring technique is altered and the job is placed in the high priority queue.
- For a CONTEND to CLAIM transition, the actions are analogous to the STANDBY to CLAIM transition.
- For a CLAIM to STANDBY transition, if the job was waiting for the dataset in the initiator then the job is requeued and placed on DCS HOLD status. Dataset availability monitoring is initiated. If the job was in DCS HOLD

status, the dataset monitoring technique is adjusted. The job is removed from the high priority queue.

- For a CLAIM to CONTEND transition, the actions are analogous to a CLAIM to STANDBY transition.

To sum up the above rules, ***the actions of DCS for service transitions are what you would expect***. There are no unusual quirks.

Aging Service—When Does the Clock Start?

The Aging timer starts when the first dataset contention situation for the job occurs. It continues "ticking" until the job is either cancelled, manually released from DCS control, or begins execution (all datasets available).

Requesting Dataset Service Level

There are three facilities that you can use to request dataset service levels:

1. JAL
2. DAL
3. JECL

For a particular job you can use any or all of the three facilities. As a result, since the request can come from three different sources, there are rules of precedence. These rules are discussed later in this chapter.

Requesting Service Level in JAL

An action statement is available to request dataset service level in JAL. The statement is fully documented in the ***JAL Reference Guide***. Here we present the statement by way of an example:

```
DCS SERVICE ($DEFAULT(30),CLAIM)
```

The DCS SERVICE statement in JAL ***requests service for all the datasets in the job***. In the example, $DEFAULT means that the initial dataset service level is the one provided by the installation as the default service. If contention occurs and 30 minutes elapse without getting the datasets, the service is to be upgraded to CLAIM.

Requesting Service Level in DAL

DAL has been designed specifically for DCS so service requests can be made at the dataset level. The action statement has exactly the same format as in JAL, so a statement such as:

```
DCS SERVICE ($DEFAULT(30),CLAIM)
```

has the same meaning as in JAL with one fundamental difference: ***it only applies to the dataset being processed in DAL at the time the action statement is executed***.

A DAL service request always overrides a JAL service request.

Since DAL processes one DD statement at a time, it is possible that the same dataset may appear in more than one DD statement for the job. If you chose to make several service requests for the same dataset, the question then is which service request takes effect? Each time DAL is invoked, you can make a DCS SERVICE request.

☞ ***If several requests are made for the same dataset in different invocations of DAL, DCS uses the one requesting the highest initial service level.***

This is not to be confused with making several DCS SERVICE requests during the same invocation of DAL. In this case, the last one takes effect.

Requesting Service Level with JECL

JECL facilities are provided so users can replicate the services available in DAL and JAL. This gives your installation the opportunity to "distribute" to your users the ability to influence the behavior of DCS.

It is important to realize that your installation does not lose control by "outboarding" these facilities. JAL and DAL allow you to determine whether or not a job has DCS control statements and the type of statement. This is done by testing the Descriptor $DCJECL. As a result, you can allow job submitters to use as much or as little as desired. If the job submitter is not allowed to use a particular capability, you can generate an informative message and fail the job.

Replicating JAL Services with JECL

A statement is available that lets you make a service request at the job level. It is fully documented in ***JECL Reference Guide*** This is an example:

```
/*DCS SERVICE JOB (DEFAULT(30),CLAIM)
```

You can see the similarity between this JECL statement and the DCS SERVICE statement for JAL shown earlier.

This JECL request always overrides a JAL request.

Replicating DAL Services with JECL

DAL requests are always associated with a specific DD statement. To provide the equivalent service in JECL, two statements are available:

- /*DCS FORDSN
- /*DCS SERVICE

The /*DCS FORDSN statement identifies the dataset for which a service level is requested. It is fully documented in ***JECL Reference Guide*** Here is an example:

```
/*DCS FORDSN AR900M.COLLECT.* SID=10
```

This statement refers to any dataset in the job that has three levels, the first and second must match 'AR900M.COLLECT'. Because the third level is represented by the wildcard character'*', the only requirement is that a third level be present.

The /*DCS SERVICE statement with the SID keyword associates a request with a /*DCS FORDSN statement. For example:

```
/*DCS SERVICE 10 (CONTEND(20),CLAIM)
```

The value 10, after the SERVICE keyword, associates this statement with the DCS FORDSN statement shown above.

A JECL dataset service level request (/*DCS FORDSN) takes precedence over:

- A JECL job service level request (/*DCS SERVICE JOB).
- DAL requests.
- JAL requests.

☞ ***If more than one /*DCS FORDSN statement applies to a given dataset, DCS will use the one requesting the highest initial service level.***

Precedence Rules

As indicated earlier, requests for dataset services can be made from different sources. When contention occurs DCS looks for instructions in the following order:

1. JECL—Dataset request.
2. JECL—Job request.
3. DAL request.
4. JAL request.
5. DCS SET initialization statement.
6. DCS internal default.

For each dataset causing contention, DCS determines if a request was made from a given source. If the answer is yes, ***no other lower level source is interrogated***. All the information is then gathered from a single source.

If the particular source is either 1 (JECL dataset request) or 3 (DAL request), then there may be more than one request for the same dataset. If that is the case, the DCS chooses the request with the highest initial service level.

Self-Inflicted Contention

A common dataset contention situation is the result of job submitters having one or more of the datasets allocated in their own TSO session.

DCS provides your installation with the ability to detect this situation and automatically free the dataset. For this to take place, the following conditions must be satisfied:

- AUTOFREE is allowed. The default is no AUTOFREE. You can alter this default by coding AUTOFREE(YES) in the DCS SET TMSS statement.
- The TSO user causing the dataset contention is the same as the job submitter.
- The dataset in question ***is allocated but not in use***.

This is not considered to be an “act of dataset repossessing” since the holder and the submitter are one and the same.

Repossessing Datasets

For CLAIM service, it is possible to request dataset repossessing. The rules are as follows:

- To request repossessing, you code CLAIM(R) in your dataset service list. For example:

  ```
  DCS SERVICE (CONTEND(10),CLAIM(R))
  ```

- When CLAIM(R) is specified, ***it must be the last element in the service list***. For example:

  ```
  DCS SERVICE (CLAIM(R),CONTEND)
  ```

  is an invalid statement.

- CLAIM(R) takes effect when the job is awaiting datasets in an initiator. If the job cannot be initiated, no repossessing actions take place.
- Datasets that are ***allocated but not in use*** by TSO sessions are automatically repossessed.
- You can also request repossessing for ***datasets that are opened for input***. To do so, you need to code a DCS REPO statement. This statement is described below.
- If Dataset Reservation is active, a GDG base is not allowed to receive CLAIM service. This ensures that the absolute data set name, associated with a relative generation data set, remains accurate.

The DCS REPO Service

As indicated earlier, one of the attributes of CLAIM service is the ability to repossess datasets from TSO holders. Requesting CLAIM(R) results in the automatic repossessing of a dataset if, and only if, it is allocated but not in use.

If you want repossessing of datasets from TSO sessions that have them opened for input, you must instruct DCS explicitly with a DCS REPO request (in addition to requesting CLAIM(R) service). You can make the request from JAL, DAL, and JECL.

JAL Dataset Repossessing Requests

If you only want to repossess ***allocated but not in use datasets***, you simply need a DCS SERVICE statement with a CLAIM(R) as the last element of the list.

All datasets associated with the job are eligible for repossessing, ***except*** relative GDGs (for example: BASE(-1)) or GDG BASEs.

If you also want to repossess datasets opened for INPUT, you code a DCS REPO statement. Its format is as follows:

```
DCS REPO USERID(pat1,...,patn) {INPUT | EXCLUDE}
```

You can specify the TSO users for which repossessing for input datasets is to take place.

You can also exclude particular users from any dataset repossessing.

This statement is fully documented in ***JECL Reference Guide***.

DAL Dataset Repossessing Requests

This follows the same format as in JAL. The request applies only to the particular dataset being processed. Repossession will not take effect for either relative GDG's (for example: BASE(-1)) or a GDG BASE datasets.

JECL Dataset Repossessing Requests

Job Level Requests

If you only want to repossess ***allocated but not in use datasets*** you simply need a "/*DCS SERVICE JOB" statement with a CLAIM(R) as the last element of the list.

All datasets associated with the job, exept either relative GDGs (for example: BASE(-1)) or a GDG BASE, are eligible for repossessing.

If you also want to repossess datasets opened for INPUT you code a "/*DCS REPO JOB" statement. Its format is as follows:

```
/*DCS REPO JOB USERID=(pat1,...,patn),{INPUT | EXCLUDE}
```

You can specify the TSO users for whom repossessing for input datasets is to take place. You can also exclude particular users from any dataset repossessing.

This statement is fully documented in the ***JECL Reference Guide.***

Dataset Level Requests

In JECL, dataset level requests are associated with the statement:

```
/*DCS FORDSN dataset-name,SID=nn ...
```

The SID=nn provides the link to the DCS SERVICE statement. If you want to repossess ***allocated but not in use datasets*** you simply need the "/*DCS SERVICE nn" statement to have CLAIM(R) as the last element of the list.

If you also want datasets open for INPUT then you do the following:

- Code an RID=nn in the "/*DSN FORDSN" statement.
- Make sure that the service statement pointed to by the SID specifies CLAIM(R).
- Code a "/*DCS REPO"statement.

An example will illustrate what is needed:

```
/*DCS FORDSN Z5900.INPUT,SID=10,RID=20
/*DCS SERVICE 10,(CONTEND(20),CLAIM(R))
/*DCS REPO 20,USERID=(Z5*),INPUT
```

In this example, a service request is made for a dataset named "Z5900.INPUT". For the first 20 minutes, a service level of CONTEND is requested. After 20 minutes a service of CLAIM with the repossess attribute is to be initiated. The dataset will be repossessed from all TSO users who have it allocated but not in use. It will also be repossessed from any TSO user who has a userid starting with 'Z5' and has the dataset opened for input.

Examples

The examples shown here are for the purpose of clarifying the rules for service level assignment. They might not represent a realistic situation.

For our examples, we'll use the job shown below. We'll repeat this example as necessary on the following pages so that you will not have to keeping flipping back to this page.

```
//TEST JOB ....
//STEP1 EXEC ...
//DD1 DD DSN=FILE.A,...
//DD2 DD DSN=FILE.B,...
//STEP2 EXEC ...
//INDD DD DSN=FILE.A
```

Sample Job

This job has three DD statements but only two unique datasets.

Example 1

In our JAL we have the following unconditional statement:

```
...
DCS SERVICE (CONTEND)
...
```

In this case both datasets have a service level of CONTEND.

Example 2

In this example, we have created a JAL where initially we have an unconditional DCS SERVICE statement. Later in the JAL logic we have a conditional DCS SERVICE statement that is executed only for jobs with the name 'TEST':

```
...
DCS SERVICE (STANDBY)      /*UNCONDITIONAL SERVICE ASSIGNMENT*/
...
IF $JOBNAME(TEST)
   DCS SERVICE (CONTEND)   /*EXECUTED FOR JOBS CALLED TEST*/
ENDIF
```

The second DCS SERVICE statement overrides the first one, so the service assigned to all the datasets in the job is CONTEND.

Example 3

Let's now introduce DAL requests in addition to JAL requests:

```
/*   JAL LOGIC   */
...
IF $JOBNAME(TEST)
   DCS SERVICE (CONTEND)   /*EXECUTED FOR JOBS CALLED TEST*/
ENDIF
...
/*   DAL LOGIC   */
...
IF $DDNAME(INDD)
   DCS SERVICE (CLAIM)
ENDIF
...
```

The JAL SERVICE request assigns a service of CONTEND to all the datasets for jobs called 'TEST'.

```
//TEST JOB ....
//STEP1 EXEC ...
//DD1 DD DSN=FILE.A,...
//DD2 DD DSN=FILE.B,...
//STEP2 EXEC ...
//INDD DD DSN=FILE.A
```

Sample Job (Repeated)

The DAL service request assigns a service of CLAIM to any dataset associated with a DD name of INDD. Our job has a DD with that name with a dataset 'FILE.A', so for that dataset the service is CLAIM. This is because DAL requests override JAL requests.

Example 4

For this example, we do not have any requests in JAL. The following DAL has been coded:

```
...
IF ($DSNAME(FILE.A) & $DDNAME(DD1))
   DCS SERVICE (CONTEND)
ENDIF
...
IF ($DSNAME(FILE.A) & $DDNAME(INDD))
   DCS SERVICE (CLAIM)
ENDIF
...
```

The first time FILE.A is processed by DAL is with the DD statement named 'DD1', as a result FILE.A is assigned a service of CONTEND. The second time FILE.A is processed is with the DD statement named INDD, it is assigned a service of CLAIM. Here we have a situation where two different service levels have been requested for the same dataset. DCS chooses the highest service level. In this case it is CLAIM.

FILE.B is given the default service level, since its DD name does not satisfy either conditional statement.

Example 5

This is like example 4 but we want to request dataset repossessing from "allocated only" TSO users for the file named 'FILE.A'.

```
...
IF ($DSNAME(FILE.A) & $DDNAME(DD1))
   DCS SERVICE (CONTEND)
ENDIF
...
IF ($DSNAME(FILE.A) & $DDNAME(INDD))
   DCS SERVICE (CLAIM(R))
ENDIF
...
```

In this case 'FILE.A' will be automatically repossessed from any TSO user who has it "allocated only" because of the CLAIM(R) request.

Example 6

This is like example 5, but we want to request the following:

- Repossessing of FILE.A from all "allocated only" TSO users.
- Repossessing of FILE.A from TSO users with an id starting with "USER", if the file is opened for input.
- No repossessing for TSO user "ADMIN01".

```
...
IF ($DSNAME(FILE.A) & $DDNAME(DD1))
   DCS SERVICE (CONTEND)
ENDIF
...
IF ($DSNAME(FILE.A) & $DDNAME(INDD))
   DCS SERVICE (CLAIM(R))
   DCS REPO USERID(USER*) INPUT
   DCS REPO USERID(ADMIN01) EXCLUDE
ENDIF
...
```

In this case, 'FILE.A' will be automatically repossessed from any TSO user who has it "allocated only" because of the CLAIM(R) request. The other two conditions are satisfied with the DCS REPO statements added.

Facilities Summary

Dataset Contention Services JECL Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
/*DCS FORDSN	Identifies the dataset for which service is being requested.	7
/*DCS SERVICE	Defines the service requested for a dataset identified by a /*DCS FORDSN JECL statement.	7

Chapter 4. Nagging

This chapter shows you how to request and control the nagging services provided by DCS.

Introduction

The term "nagging" is used to describe the process of informing TSO users that they are holding a dataset that is required by a batch job or a started task. ***The nagging process is automatic***. Whenever a dataset contention situation occurs, DCS takes it upon itself to NAG.

The NAG is a two line message with the following format:

```
DTM7108I  PLEASE FREE DATASET dataset-name time
DTM7109I  THIS DATASET IS REQUIRED BY jobnumber jobname
```

Default values are provided to control the number of times a TSO user is nagged and the frequency, in minutes, of nagging. The default values are different for datasets required by a job or a started task.

Implementation Summary

Nagging is an automatic process that does not require any action on your part. Explicit actions are necessary only when you want to adjust the nagging process.

Dependencies

The only dependencies are the system default values for the number of times a user is nagged, and how frequently nagging occurs. You can provide system defaults with the DCS SET initialization statement.

TMSS Initialization Statement

DCS has built-in default values to NAG TSO users. The default values are:

- For datasets required by batch jobs:
  - 3 times 5 minutes apart.

- For datasets required by started tasks:
  - 99 times 1 minute apart.

These global default values can be modified with the DCS SET initialization statement of TMSS.

Adjusting NAGs

You can adjust the nagging process. The following controls are provided:

- You can modify the second line of the NAG to reflect whatever text you want.
- You can decide which user or group of users are NAGGED.
- You can also suppress NAGGING to a user or users.
- You can request the NAGGING frequency.
- You can indicate how many times users are NAGGED.

You can combine any of the above options to fine-tune NAGGING to cover practically any situation. The requests to fine tune nagging can originate in JAL, DAL, and JECL.

Adjusting NAGs in JAL

The JAL NAG adjusting mechanism provides:

- A definition statement, DCS_NAGDEF, where you define the text of the message.
- A communication statement, DCS NAG, to request DCS to NAG a user, or group of users, holding a dataset. You can also request the number of times the user is NAGGED and the time interval between NAGs.

To illustrate the above statements, we can use the following situation. For a particular job, you want to increase the tempo of nagging. You want any holder of datasets needed by the job to be nagged every 2 minutes, up to 99 times, but you do not want to change the text of the default message. Since you do not want to alter the standard text message, you do not need a DCS_NAGDEF statement. You only need to issue the DCS NAG statement:

```
IF ...   /*IDENTIFY THE JOB*/
   DCS NAG $DEFAULT USERID(*) REPEAT(99,2)
   ...
```

The keyword $DEFAULT indicates that the standard message is to be issued. The use of an asterisk as a parameter to the USERID keyword indicates “all us-

ers". So the above statement says: if contention occurs, issue the standard NAG message to all holders of the dataset, every 2 minutes, up to 99 times.

Since the request is made in JAL, it applies to all the datasets associated with the job.

You might want to be more discriminating in your message. Let's say you want to send a special message to a user with an id of APM9001, but you do not want to change the frequency of nagging. Since you are altering the message text, you need a DCS_NAGDEF statement:

```
DCS_NAGDEF PAYNAG ('the-text-you-want....')
...
IF ...   /*IDENTIFY THE JOB*/
   DCS NAG PAYNAG USERID(APM9001)
   ...
```

The name 'PAYNAG' links the definition statement with the DCS NAG statement.

In the next example we have a situation where we do not want to nag a particular user whom we know is authorized to do administrative work on the dataset. For illustrative purposes, we show the same user id:

```
IF ...   /*IDENTIFY THE JOB*/
   DCS NAG $SUPPRESS USERID(APM9001)
   ...
```

The keyword $SUPPRESS indicates that APM9001 is not to be nagged.

More than one DCS NAG statement can be coded with different adjustments for different TSO users. When determining if a DCS NAG for a TSO user holding a dataset exists, DCS gives preference to user IDS that have been coded in full. (For example, 'Z5922JD' is given preference over'Z59*'.)

The DCS_NAGDEF and DCS NAG statements are documented in the manual ***DAL Reference Guide***.

Adjusting NAGs in DAL

The same facilities that are available in JAL are also available in DAL. The essential difference is that a DAL DCS NAG request is assigned to the dataset associated with the DD statement being processed.

NAG requests from DAL take precedence over JAL.

More than one DCS NAG statement can be coded with different adjustments for different TSO users. As with requests from JAL, DCS gives preference to user

IDS that have been coded in full. (For example, 'Z5922JD' is given preference over 'Z59*'.)

NAGGING From JECL

Nagging can be adjusted by users from JECL. To allow installation control of this service, facilities are provided in JAL to detect JECL NAG requests. This is done by testing the Descriptor $DCJECL.

Replicating JAL Services with JECL

The following facilities are available in JECL for ***job level nagging*** adjustments:

- A DCS NAG JOB statement that allows you to indicate the following:
  - A list of userid patterns.
  - The repeat interval and number of times to nag.
  - An optional JECL statement, DCS NAGTEXT, with modified text for the message.
- A DCS NAGTEXT that allows you to specify the text, if you want your own message.

We can illustrate this with an example:

```
/*DCS NAG JOB USERID=(PAY9*,PER*),REPEAT=(10,2)
/*DCS NAGTEXT JOB '**** CRITICAL PAYROLL DATASET ****'
```

In this example, we are requesting that any holder of a dataset needed for the job who has a userid starting with 'PAY9' or 'PER' be nagged 10 times every 2 minutes.

For these users, the second line of the standard Nagging text is to be substituted with the '**** CRITICAL PAYROLL DATASET ****' message.

JECL requests override JAL and DAL requests.

Replicating DAL Services with JECL

This is similar to the previous section, but in the case of DAL the requests are for specific datasets. We need a facility in JECL that allows us to specify the particular dataset that the NAG is to be associated with. The JECL statement DCS FORDSN gives us that capability.

Let us replicate the previous JECL job level nagging, but in this case only for one dataset.

For example, assume the dataset name is PAY800.WEEKLY.DETAIL. The JECL statements needed will look like this:

```
/*DCS FORDSN PAY800.WEEKLY.DETAIL,NID=10
/*DCS NAG 10 USERID=(PAY9*,PER*),REPEAT=(10,2),TID=1
/*DCS NAGTEXT 1 '**** CRITICAL PAYROLL DATASET ****'
```

The first statement identifies the dataset. It is associated with the NAG statement because of the NID=10. The DCS NAG statement is associated with the NAGTEXT because of the TID=1. The identification values are arbitrary. You choose a value from 1 to 99 that allows you to link the appropriate statements.

☞ ***The ID linkages are a bit awkward, but they are needed because JES2 JECL statements cannot be continued.***

Precedence Rules

Since NAG requests can originate from multiple sources, precedence rules are needed. They are as follows:

1. Dataset JECL request with a specific TSO user id match.
2. Dataset JECL request with a generic TSO user id match.
3. Job JECL request with a specific TSO user id match.
4. Job JECL request with a generic TSO user id match.
5. DAL request with a specific TSO user id match.
6. DAL request with a generic TSO user id match.
7. JAL request with a specific TSO user id match.
8. JAL request with a generic TSO user id match.
9. DCS SET initialization parameter
10. DCS built-in default.

As soon as a match occurs, the search is terminated. DCS takes the NAG specifications from the NAG request that matched the TSO user id.

Considerations

- The initial message is sent to a TSO dataset holder when the job requesting the dataset goes into contention. Subsequently, and if needed, further messages are issued at the requested time interval.
- DCS nags any TSO holder who is running in a system that is part of the DCS complex. JES2 MAS Node considerations are not applicable. DCS communicates across multiple JES2 nodes using ThruPut Manager's XCFM support.
- A TSO user holding more than one dataset required by a job can be notified more than once a cycle.

Facilities Summary

NAG Services JECL Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
/*DCS FORDSN	Identifies the dataset for which service is being requested.	7
/*DCS NAG	Defines the frequency of nagging and who will be nagged.	7
/*DCS NAGTEXT	Specifies the text to be used when nagging.	7

Chapter 5. Alerting

This chapter discusses the ALERT facility of DCS. It shows the services available and how to request them.

Description

DCS allows you to "alert" groups, such as production control, about important jobs that are experiencing delays because of dataset contention. ALERTs use the spooling and printing facilities available with Printing Services.

The ALERT facility should not be confused with the NAG facility. An ALERT is addressed to the group (or person) responsible for the job experiencing the delay. NAGs are addressed to TSO users holding the dataset(s) needed for the job to execute.

With ALERTs you can control:

- Whether or not an ALERT is generated for a job.
- Header and Trailer text.
- The interval time before it is produced.
- The destination.

An ALERT Example

An example of an ALERT is shown on the following page. A brief explanation of the information provided with an ALERT is discussed.

In this figure:

Type 1 line

Identifies the ALERT. The date and time represents when the ALERT was spooled.

Type 2 lines

These represent the two headers that you can insert to customize the ALERT.

```
 DATASET CONTENTION SERVICES ALERT FEB 10, 1991 13:35
          *** YOUR OWN HEADER 1 ***
          *** YOUR OWN HEADER 2 ***
 JOB 10127 PAY90001  NODE CENTRAL
 CONTENTION          : ON SYS3  FEB 10, 1991 13:15
STATUS               : WAITING IN INITIATOR
DSN                  : AP900.MASTER.PAY  G/EXC/CLAIM
HELD BY              : USR932  TSU SYS02 SHR
          *** YOUR OWN FOOTER 1 ***
          *** YOUR OWN FOOTER 2 ***
```

An ALERT Example

Type 3 and 4 lines

Identifies the job that triggered the ALERT.

Type 5 line

This shows whether the job is waiting in an initiator (CLAIM service) or HELD.

Type 6 and 7 lines

There can be several lines of this type. For each dataset affecting the job, there is one type 6 line. Then, for each type 6 line, there are as many type 7 lines as there are dataset holders.

For type 6 lines, in addition to the dataset name, the following is shown:

- The scope of the enqueue: L for system and G for systems (local/global).
- The current usage: SHR for shared and EXC for exclusive.
- The service requested: CLAIM, CONTEND, or STANDBY.

Type 8 lines

These represent the two footers that you can insert to customize the ALERT.

Implementation Summary

The steps for implementing ALERT requests are:

- Determine the type of ALERTs needed.
- Decide the number of places you want ALERTs printed. Normally, one per shared spool complex is all that is needed.
- If you are using CPS, ensure that a Writer has been defined that is eligible to print ALERTs. The Writer must service the destination assigned to ALERTs in JAL or JECL. To use SPOOLing, ensure that the SPOOL File is defined and active.
- If you choose to use SPS, ensure that the proper SPS DEFINE has been done, either through a TMSS initialization statement or through an operator command.

Dependencies

Printing Services

ALERTs require that you select one of the available Printing Services.

CPS Writer

If you use CPS to print ALERTs, you must define at least one CPS WRITER that services the ALERT destination.

SPOOL File

If you use CPS, the SPOOL File must be active.

SPS Definition

If you use SPS, you must ensure that an SPS DEFINE has been done to establish the SYSOUT characteristics needed to print ALERTs.

TMSS Initialization Statements

The DCS SET statement is available. It is not mandatory but allows you to set defaults for the ALERT interval and destination.

ALERT Formatting

Formatting for ALERTs varies somewhat depending on the printing facility chosen.

ALERTs Using CPS

The formatting of ALERTs printed with CPS is done with a CPS Writer. A number of keywords are provided to describe the number of lines per page, the number of blank lines on the top margin, and the number of blank lines on the bottom margin. They are described with the TMSS initialization Statement CPS WRITER in ***Base Product: System Programming Guide*** **"**Chapter 2. TMSS Initialization Statements."

ALERTs Using SPS

The formatting of ALERTs printed with SPS is determined by doing an SPS DEFINE. This can be done with an SPS DEFINE operator command, but usually should be done with the SPS DEFINE TMSS initialization statement. Both methods provide keywords to describe the number of lines per page, the number of blank lines for top margin, and the number of blank lines for the bottom margin.

The SPS DEFINE initialization statement is described in ***Base Product: System Programming Guide*** "TMSS Initialization Statements." The SPS DEFINE operator command is documented in the manual ***Command Reference Guide***.

Requesting ALERTs

DCS does not produce ALERTs unless it is instructed to do so.

You can request ALERTs in JAL or in JECL.

JAL Requests

The JAL ALERT facility provides:

- A definition statement, DCS_ALERTDEF, where you define the destination for an ALERT and the headers and footers to "envelope" the ALERT text.
- A communication statement, DCS ALERT, to request ALERTing after a period of time has elapsed and the job is still waiting for a dataset.

To introduce the JAL ALERT facilities we can use the following example: for all payroll jobs, we want an ALERT generated if dataset contention has not been re-

solved in 20 minutes. We want to highlight the ALERT with a heading indicating we are dealing with a payroll job. To do this, we first define the ALERT header with a definition statement:

```
...
DCS_ALERTDEF PAYROLL HEADER('****  NOTIFY PAYROLL BATCH CONTROL  ****')
...
```

Then we request that the ALERT be generated after 20 minutes:

```
IF ....   /*IDENTIFY PAYROLL JOBS*/
   DCS ALERT PAYROLL TIME(20)
   ...
ENDIF
```

The JAL statements are documented in the ***JAL Reference Guide***.

JECL Requests

Users can request ALERTs with JECL statements. To allow installation control of this facility, facilities are provided in JAL to detect JECL ALERT requests. This is done by testing the Descriptor $DCJECL.

☞ ***In JAL, you can either allow or disallow jobs with JECL ALERT requests. You cannot override their JECL requests with JAL requests.***

Three JECL statements are provided to request ALERTs:

1. The /*DCS ALERT statement. It is used to request an ALERT.
2. The /*DCS ALERTH statement. It is used to define the text for the two optional headers.
3. The /*DCS ALERTF statement. It is used to define the text for the two optional footers.

These statements are documented in ***JECL Reference Guide***

Facilities Summary

ALERT Services Initialization Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
CPS WRITER	Defines a CPS Writer to be used for ALERTs.	2
DCS SET	Defines default time interval and destination for ALERTs.	2
FILE SPOOL	Defines a SPOOL File for use with a CPS Writer.	2
SPS DEFINE	Defines an SPS Writer to be used for ALERTs.	2

ALERT Services JECL Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
/*DCS ALERT	Requests an ALERT.	7
/*DCS ALERTF	Defines footer text for an ALERT.	7
/*DCS ALERTH	Defines header text for an ALERT.	7

Chapter 6. Altering Dataset Disposition

This chapter introduces you to the ALTER facility of DCS, which allows you to alter the first parameter of the DISP keyword, and DCS Dataset Reservation.

Introduction

With DCS, you can analyze the usage of the disposition parameter for each DD statement that is associated with a permanent dataset. Depending on what you find, you can instruct DCS to take one of the following actions:

- Fail or warn the job.
- Alter the disposition for that DD statement.

Requests for DCS to alter the disposition are done at the DD statement level. All the relevant information about the job and the particular DD statement is available to you. As a result, you can provide specific rules for the ALTERation of the first parameter of the DISP keyword.

DCS also supports Dataset Reservation, which alleviates problems with ENQs for GDGs and dataset aliases by changing the timing of the actual ENQ.

Implementation Summary

The ALTER facility is only available in DAL. To make use of this facility:

- Code the appropriate DAL statements.
- Create DAL internal text with the Language Processor.
- Implement the DAL.

The Dataset Reservation facility is activated with the DR keyword of the DCS SET statement as described in "TMSS Initialization Statements" in the ***Base Product: System Programming Guide***. To use this facility:

- Code DR(YES) on the DCS SET initialization statement.
- If there are jobs for which you do not want Dataset Reservation, code JAL to detect these jobs and turn off Dataset Reservation support.

Dependencies

There are no dependencies to request ALTER services, other than activating the DAL you have coded with the ALTER rules.

Dataset Reservation support does not have any dependencies.

TMSS Initialization Statements

To activate the DAL for disposition alteration, a DAL LOAD statement is required.

To activate Dataset Reservation support, DR(YES) must be included on the DCS SET statement.

These statements are documented in ***Base Product: System Programming Guide*** "TMSS Initialization Statements."

How to Request Disposition ALTERing

Requesting alteration of disposition is done with a DCS action statement in DAL. Normally you set up the rules for alteration using the logic capabilities available in DAL. For example, you might want to enforce that any reference to SYS1.MACLIB as SYSLIB have a disposition of SHR. The DAL statements would look like this:

```
...
IF ($DSNAME(SYS1.MACLIB) & $DDNAME(SYSLIB)) /*DETECT USAGE OF MACLIB */
   DCS ALTER DISP1(SHR)
...
```

The DCS ALTER request operates only on DD statements with a DD name of SYSLIB and a DSN of 'SYS1.MACLIB'.

A more reasonable approach might be the following: in addition to the above statements, include logic to test for the disposition. If it is already SHR, do not do anything. If it is OLD, issue a warning message and ALTER the disposition:

```
MSGDEF MSG1 ('IMPROPER USE OF DISPOSITION FOR ',$DDNAME, 'DD STATEMENT')
MSGDEF MSG2 ('REQUEST ALTERED TO SHR')
...
...
IF ($DSNAME(SYS1.MACLIB) & $DDNAME(SYSLIB)) /*DETECT USAGE OF MACLIB*/
   IF ($DISP1(SHR))
      RETURN
   ELSE
```

```
      WTU MSG1
      WTU MSG2
      DCS ALTER DISP1(SHR)
      RETURN
   ENDIF
ENDIF
```

Considerations for Disposition Altering

- The ALTER is only applicable to the first parameter for the DISP keyword. This parameter is known in the JCL manual as the dataset's status.
- The alteration is for the purpose of dataset enqueueing.
- A DCS ALTER is associated with the DD statement being processed by DAL. It does not alter other DD statements referring to the same dataset.
- If you want to change the disposition for all the occurrences of a dataset in a job, you must do it for each DD statement.
- If the original specification of the DD statement is NEW, or MOD for a non-existing dataset, the DCS ALTER statement does not take effect.
- The intended use is to ALTER OLD to SHARED so unnecessary serialization can be prevented. It can also be used to ALTER SHARED to OLD, if a particular situation calls for this change. For example, you might not want to allow tape datasets to be referenced as SHARED since it causes problems with volume contention.
- If you are "looking" for a JOBLIB DD statement, remember that it occurs before the first EXEC statement. Any descriptor associated with a step, such as program name or step name, will contain blanks.
- Only DD statements for permanent datasets cause DAL to be invoked.

Using Dataset Reservation

Datasets referred to by aliases or relative GDG numbers can result in dataset reservation failures at step allocation time, when the real dataset name is resolved. This is because without Dataset Reservation, ENQs for the absolute dataset name are not obtained until that point.

DCS can eliminate these failures by ENQing on the real dataset name during job initiation. At that time, if there is a conflict the job can be requeued. For tape

datasets, the Dataset Reservation ENQ is upgraded to "exclusive" if necessary, eliminating problems with volume contention.

All that is required to take advantage of DCS Dataset Reservation is to activate it as described above. To make a job exempt from Dataset Reservation, add JAL statements to detect the job and switch off the support:

```
DCS SET DR(NO)
```

You can detect whether a DCS SET statement has been executed for a job with:

```
IF ($JXDCS_DR(YES))
 ...
```

Note that when Dataset Reservation is active, a GDG base is not allowed to receive DCS SERVICE CLAIM. This ensures that the absolute data set name, associated with a relative generation data set, remains accurate.

Facilities Summary

Dataset Alteration Initialization Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
DAL LOAD	Specifies the dataset name and member name (if applicable) where the DAL internal text is to be loaded from.	2
DCS SET	Sets global options for DCS, including whether Dataset Reservation support is activated.	2

Dataset Alteration JAL Statements Refer to *JAL Reference Guide*	
Statement	**Description**
DCS SET	Controls whether the job uses Dataset Reservation.
$JXDCS_DR	Allows you to determine whether a DCS SET statement has been executed.

Dataset Alteration DAL Statements Refer to *DAL Reference Guide*	
Statement	**Description**
DCS ALTER	Allows you to alter the SHR/OLD attributes of the dataset currently being processed.

Chapter 7. Collecting SMF Data for TM/DCS

This chapter describes the facilities provided to collect SMF data about the TM/DCS application.

SMF Monitoring of TM/DCS Activity

A ThruPut Manager facility is provided for an installation to gather statistical information about the activity of TM/DCS. An SMF record containing this information is generated for each occurrence of dataset contention that TM/DCS manages.

The data collected includes:

- Date and time of the contention event.
- The system on which the event occurred.
- The type of job: STC, TSU, or JOB.
- The job name, one job per record. If a job encounters multiple contention events, each one is described in a separate record.
- The starting and ending date and time of the contention event.
- For each dataset unavailable to the job:
  - The dataset name.
  - The type of usage: exclusive or shared.
  - The service requested: STANDBY, CONTEND, CLAIM, or CLAIM(R).
  - Information about each holder of the dataset:
    - The JES2 node.
    - The system ID.
    - The type of job: STC, TSU, or JOB.
    - The jobname and JES2 job number.
    - The type of usage: exclusive or shared.
    - The OPEN status: INPUT, OUTPUT, or not OPEN.

Activating Data Collection for TM/DCS

To collect data from the DCS application, you must indicate this on the TM SMF initialization statement:

```
TM SMF TYPE(255) ANALYZER(9,SMFDATA) DCS
```

Identifying TM/DCS Records

TM/DCS records are identified by their subtype of 6. The subtype is found in the field named SMFTMSTP in the DTMSMFPR mapping macro.

Summary of TM/DCS SMF Data

The mapping macro DTMSJCR is provided to assist you in analyzing the collected data.

☞ ***The following table is intended for use as a guide only. It is not a complete representation of the collected data.***

Refer to the mapping macro for a complete description. If there is a discrepancy between this summary and the mapping macro DTMSJCR, the macro is correct.

SUMMARY OF THE THRUPUT MANAGER SMF TM/DCS RECORD For complete mapping, refer to macro DTMSJCR			
Name	**Type**	**Length**	**Description**
SMF Header Segment			
SMFSJTME	binary	4	Contention time (see note 1)
SMFSJDTE	packed	4	Contention date (see note 2)
SMFSJSID	char	4	JES2 System on which contention occurred
SMFSJWID	char	4	JES2 Subsystem identification
DCS Header Segment			
CMFHDTE	packed	4	Contention date (see notes 1, 3)
CMFHTME	packed	4	Contention time (see note 4)

SUMMARY OF THE THRUPUT MANAGER SMF TM/DCS RECORD For complete mapping, refer to macro DTMSJCR			
Job Segment			
SJCSJNOD	char	8	JES2 node
SJCSJSID	char	4	JES2 system ID
SJCSJBTP	char	4	Job type: STC, TSU, or JOB
SJCSJBTP	binary	8	JES2 job number
SJCSJBNM	char	8	JES2 job name
SJCSRST	binary	4	Reader start time (see note 1)
SJCSRSD	packed	4	Reader start date (see note 2)
SJCSCID	binary	4	Unique identifier for this contention event
SJCSCST	binary	4	Start time of contention event (see note 1)
SJCSCSD	packed	4	Start date of contention event (see note 2)
SJCSCET	binary	4	End time of contention event (see note 1)
SJCSCED	packed	4	End date of contention event (see note 2)
Dataset Segment—One per dataset unavailable due to contention			
SDCSDSNM	char	44	Dataset name
SDCSUSG	char	1	Usage: EXCLUSIVE or SHARED
SDCSSERV	char	1	DCS Service requested: STANDBY, CONTEND, CLAIM, or CLAIM(R)
Holder Segment—One for each holder of the dataset			
SDHSJNOD	char	8	JES2 node of holder
SDHSJSID	char	4	JES2 system ID of holder
SDHSJBTP	char	4	Job type: STC, TSU, or JOB
SDHSJBNOF	binary	8	JES2 job number of holder

SUMMARY OF THE THRUPUT MANAGER SMF TM/DCS RECORD For complete mapping, refer to macro DTMSJCR			
SDHSJBNM	char	8	JES2 job name of holder
SDHSUSG	char	1	Usage: EXCLUSIVE or SHARED
SDHSOPN	char	1	OPEN status: INPUT, OUTPUT, or not OPEN

Notes

1. Time in hundredths of seconds since midnight.
2. Julian date in the form ***cyyddd*** as returned by the TIME macro.

Chapter 8. The Dataset Contention Reporting System

This chapter introduces you to the reporting system available with DCS. A sample of the two reports available is also included.

Introduction

DCS reporting facilities include:

- Facilities in JAL to request contention data recording.
- Generation of data and recording in the CMF file.
- A snap-shot facility to capture the data recorded in the CMF file. The output file created by the snap-shot utility is used as input to the reporting program.
- Selective reporting by type of job, duration of contention, dataset name, and other criteria.
- A reporting program that generates two reports that allow you to quickly correlate "causes and effects". The reports are:
  - The Job Analysis report.
  - The Dataset Analysis report.

  An example of each one and an explanation of their contents is shown below.

The Job Analysis Report

Its purpose is to show:

1. The jobs that were affected.
2. The total duration of the contention situation.
3. The datasets that were involved.
4. How much time each dataset contributed to the problem.

A sample is shown on the next page. Let's take a look at job PAY01300:

- Since the report is organized alphabetically by job name, we can locate the entry quickly by looking at the left side of the report.

94FEB21 12:30:10 DATASET CONTENTION REPORT - JOB ANALYSIS PAGE: 1

IMPACTED JOB					OCCURRENCE				DATASETS			
NAME	NO.	DATE	TIME	RECORD ID	FROM	FOR	JESID	F	NAME	S	FOR	C
PAY00301	1235	FEB20	10:30:05	PAYROLL	10:42:03	11:09	SYS2	1	PAY900.DAILY.SLIPS	S	10:09	P
PAY00322	1338	FEB20	10:35:02	PAYROLL	10:47:01	6:11	SYS2	1	PAY900.DAILY.SLIPS	S	6:11	P
					11:02:06	18:06	SYS2	1	PAY900.DAILY.SLIPS	S	18:06	P
PAY01300	3221	FEB20	12:42:01	PAYROLL	12:55:42	26:22	SYS2	1	PAY900.DETAIL.BRANCH	C	6:13	P
									PAY900.INPUT.EDIT	S	26:22	P
					13:48:17	08:22	SYS2	1	PAY900.INPUT.EDIT	S	8:22	J
PAY04908	4697	FEB20	17:03:43	PAYROLL	17:35:42	18:10	SYS2	1	PAY923.EMPLOYEE.HOURS	S	7:02	P
									PAY983.FACTORS	C	3:18	P
									PAY985.TAX.WITHHOLD	S	18:09	P

Job Analysis Report

- The second column shows the JES2 job number, in this case 3221.
- The third column shows the submission date.
- The next column shows the submission time.
- The RECORD ID column shows the optional identification field. The value in this field is assigned at JAL time and can be used for reporting selection purposes. In this case the job is identified as 'PAYROLL'.

The 'OCCURRENCE FROM' shows you the time that the job ran into dataset contention. For this job there are two entries, indicating that contention occurred twice before the job executed

  - The first delay occurred at 12:55:42. The FOR column tells us that it lasted 26 minutes 22 seconds.
  - The second delay occurred at 13:48:17. The FOR column tells us that it lasted 8 minutes 22 seconds.

- The JESID column shows the 4-byte JES2 system identification.
- The F column is a flag that indicates how the job contention terminated. The possible values are:

  0 Another contention cycle for the job has been found.

  1 All datasets available.

  2 Operator command.

  3 ThruPut Manager was restarted—Reconciliation Process.

  4 Job purged.

  5 DCS internal error.

  6 ThruPut Manager shut-down.

  7 Service Modification CLAIM mode—Job was allowed to go to the initiator.

  8 Internal error.

  9 MAXCLAIM value for node is no longer exceeded.

- The dataset column shows one or more datasets per contention occurrence. In this example, two datasets were involved in the first occurrence:
  - PAY900.DETAIL.BRANCH
  - PAY900.INPUT.EDIT

- The dataset PAY900.DETAIL.BRANCH became available after 6 minutes and 13 seconds.
- The dataset PAY900.INPUT.EDIT took 26 minutes and 20 seconds to become available, so it was this dataset that caused the long delay.

- In the second delay, only one dataset was not available:
  - PAY900.INPUT EDIT
  - It became available after 8 minutes 22 seconds.
- The S column shows the dataset service level requested:
  - S for STANDBY.
  - C for CLAIM.
  - Q for CONTEND.
  - R for REPOSSESS.
- The C column indicates how the reporting program detected that a dataset became available:
  - A P indicates that a dataset "popped-up" record was encountered, this represents the normal case.
  - A J indicates that the dataset "popped-up" record is missing. In that situation the reporting program considers the dataset to be available when the job contention situation ended. Occasionally, you might see a J in this column. If J becomes prevalent, it is an indication that you might not be recording from all systems in your DCS complex.
- **Note:** You might see an asterisk next to a field under the following conditions:
  - The contention occurrence time is earlier than the job's reader-on time.
  - The contention occurrence time is earlier than the end of the previous contention occurrence for the job.
  - The interval time associated with one of the datasets is longer than the contention interval for the job.

  This situations can occur in multisystem environments as a result of the differences in clock values.

```
FEB21/94 12:30:10                    DATASET CONTENTION REPORT - DATASET ANALYSIS                              PAGE:   1

DATASET NAME: PAY900.DAILY.SLIPS
--NODE--  ------INTERVAL-------  --------------------JOB OCCURRENCES----------------------------  -----------HOLDERS----------
  NAME     DATE   FROM     FOR     NAME    NO.   DATE    TIME   JESID      FROM     FOR   F S C    NAME   T JESID    H    AT
--------  ----- -------- ------  -------- ----- ------ -------- -------- --------- ------ - - -   -------- - -------- - --------
TAMPA     FEB20 10:42:03  10:09  PAY00301  1235 FEB20  10:30:05 SYS02    10:42:03   11:09 1 S J   PAY00292 J SYS02    E 10:42:03
TAMPA     FEB20 10:47:01  10:09  PAY00322  1338 FEB20  10:35:02 SYS02    10:47:01    6:11 1 S J   PAY00292 J SYS02    E 10:47:01
                                 PAY00322  1338  FEB20 10:35:02 SYS02    11:02:06   18:06 1 S J   PAY00301 J SYS03    E 10:47:01

DATASET NAME: PAY900.DETAIL.BRANCH
--NODE--  ------INTERVAL-------  --------------------JOB OCCURRENCES----------------------------  -----------HOLDERS----------
  NAME     DATE   FROM     FOR     NAME    NO.   DATE    TIME     JESID     FROM     FOR   F S C    NAME   T JESID    H    AT
--------  ----- -------- ------  -------- ----- ------ -------- -------- --------- ------ - - -   -------- - -------- - --------
TAMPA     FEB20 12:55:42   6:13  PAY01300  3221 FEB20  12:42:01 SYS02    12:55:42    6:13 1 C J   UAP90MJ  T SYS01    E 12:55:42

DATASET NAME: PAY900.INPUT.EDIT
--NODE--  ------INTERVAL-------  --------------------JOB OCCURRENCES----------------------------  -----------HOLDERS----------
  NAME     DATE   FROM     FOR     NAME    NO.   DATE    TIME     JESID     FROM     FOR   F S C    NAME   T JESID    H    AT
--------  ----- -------- ------  -------- ----- ------ -------- -------- --------- ------ - - -   -------- - -------- - --------
TAMPA     FEB20 12:55:42  20:09  PAY01300  3221 FEB20  12:42:01 SYS02    12:55:42   20:09 1 S J   PAY01209 J SYS03    E 12:55:42
                                 PAY01300  3221 FEB20  12:42:01 SYS02    13:48:17    8:22 1 S J   PAY01220 J SYS03    E 13:48:17

DATASET NAME: PAY923.EMPLOYEE.HOURS
--NODE--  ------INTERVAL-------  --------------------JOB OCCURRENCES----------------------------  -----------HOLDERS----------
  NAME     DATE   FROM     FOR     NAME    NO.   DATE    TIME     JESID     FROM     FOR   F S C     NAME  T JESID    H    AT
--------  ----- -------- ------  -------- ----- ------ -------- -------- --------- ------ - - -   -------- - -------- - --------
TAMPA     FEB20 17:35:42   7:02  PAY04908  4697 FEB20  17:03:43 SYS02    17:35:42    7:02 1 S J   PAY00450 J SYS03    S 17:35:42
```

Dataset Analysis Report

TM7R1-7109

The Dataset Analysis Report

The purpose of this report is to show:

1. The datasets that were involved in contention situations.
2. The jobs that were delayed.
3. The "holders" of the dataset.

This report complements the Job Analysis report. We can show this by continuing with the example used to describe the Job Analysis report. From the previous report, we might be interested in determining the holders of the following two datasets:

- PAY900.DETAIL.BRANCH
- PAY900.INPUT.EDIT

Since the report is organized as "mini-reports" per dataset, in alphabetic order, we can quickly locate the information:

- The first column under the DATASET NAME represents the JES2 node name.
- The INTERVAL DATE/TIME shows the initial date and time that DCS detected this dataset causing contention.
- The INTERVAL FOR shows the total duration of the contention. Several jobs could have been affected during this period.
- The next subsection of the report, JOB OCCURRENCES, shows the job or jobs that were affected for the interval.
- The job name, JES2 number, submission date and time, the JES2 identification, and when the contention occurred for this job are all shown. This allows you to relate the entry to the Job Analysis report.
- Three columns containing flags are included here.
  - The F column is the same as the F column in the Job Analysis report. It indicates how the job contention terminated. The possible values are documented with the Job Analysis report.
  - The S column is the same as the S column in the Job Analysis report. It shows the dataset service level requested by this job:
    - S for STANDBY.
    - C for CLAIM.
    - Q for CONTEND.

        - R for REPOSSESS.
    - The C column is the same as in the Job Analysis report. It indicates how the reporting program detected that a dataset became available. Normally a dataset "popped-up" record is encountered, this is indicated with a P. In cases where the dataset "popped-up" record is missing, the dataset is considered to be available when the job contention situation ended. This is indicated with a J. Occasionally, you might see a J in this column. If J becomes prevalent, it is an indication that you might not be recording from all systems in your DCS complex.
- The next subsection of the report, HOLDERS, shows "who" was holding the dataset. The name of the holder is shown. It could be a batch job, a started task, or a TSO user. This is indicated in the T column:
    - J for batch job.
    - T for TSO user.
    - S for started task.
- The next column, JESID, shows the id of the system where the holder is executing.
- The H column shows the dataset status:
    - S for shared.
    - E for exclusive.
- The AT column shows the first time that the holder was detected.
- **Note:** You might see an asterisk next to a field under the following conditions:
    - The contention occurrence time is earlier than the job's reader-on time.
    - Where the HOLDER time reported under the heading "AT" is not within the boundaries of the dataset contention.

  These situations can occur in multisystem environments as a result of the differences in clock values.

Implementation Summary

The steps to implement the data collection and reporting system are as follows:

- Activate the CMF file. DCS uses this file for the data collection.
- Update the JAL to request recording for the particular job, or jobs, for which you want contention data to be collected. This is done with the DCS RECORD

statement. If you want data collection for all the jobs processed by ThruPut Manager, code an unconditional DCS RECORD statement.

- Decide what kind of reporting cycle you want.
- Put in place the procedures to run the snap-shot utility at the appropriate intervals to reflect the cycle you have chosen. For information, refer to the heading "The Snap-Shot Utility" later in this chapter.
- Decide the selection criteria you want for the reporting system. For information, refer to the heading "The Report Selection Options" later in this chapter.
- If the reports are to generated at regular intervals, put in place the procedures to ensure that the reporting program is run.

Dependencies

The CMF file

This file must be active in all the systems participating in the DCS complex.

TMSS Initialization Statements

A FILE CMF initialization statement is required.

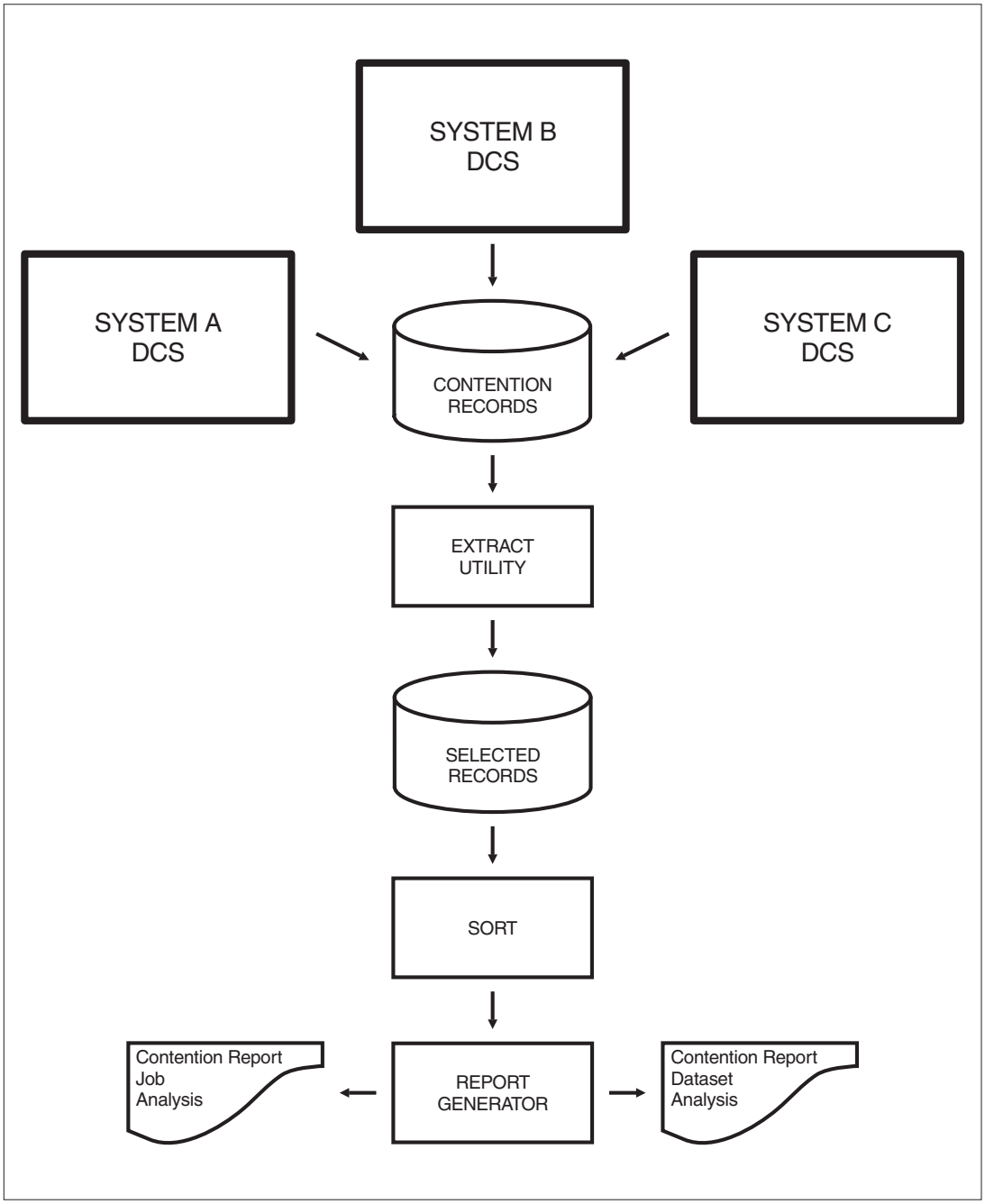
JAL

When contention occurs, DCS determines if recording was requested in JAL with the DCS RECORD action statement. If no request was made contention records are not generated.

The CMF File Considerations

The mechanism to record contention data is similar to SMF recording. The records are written to a file called CMF. ***A wrap-around technique is used to ensure that the traditional problems resulting from out-of-space conditions are eliminated.*** The CMF file is shared by all the systems participating in the DCS complex.

A diagram showing the relationships among the systems running DCS, the CMF file, and the reporting system is shown below:



The CMF File and the Reporting System

The space needed for this file depends:

- On your reporting cycle (daily, weekly, on demand, or any other interval you may choose).
- The frequency of dataset conflicts.

One cylinder of a 3380/3390 device accommodates the records generated by 700 contention situations.

Recording

Recording is done by the Contention Management Facility (CMF). This facility records contention-related data designed to be input to the contention management reporting program.

CMF has a number of similarities with SMF. Variable length records are generated to collect all the necessary information. It is unlikely that you will have to be concerned with these records, but for your information here is a list of the different record types:

Type 1—Job Contention Record

This is generated whenever a job encounters dataset contention during initiation.

Type 2—Job Release Record

This is generated whenever a job is released from DCS HOLD.

Type 3—Dataset Popup Record

This is generated whenever a dataset becomes available.

Type 4—Dataset Holder Record

This is generated whenever a holder of a contention dataset is observed for the first time.

Type 5—Job Status Change Record

This is generated whenever a dataset service is modified.

Unlike SMF, the CMF facility records all its data in a single shared file, so merging records from different systems is not required. DCS ensures that the integrity of the CMF file is maintained.

The Snap-Shot Utility

The DCS Snap-Shot utility, DTMDCSU**n** (where **n** represents the current ThruPut Manager release number) creates a copy of selected data in your CMF

```
//jobname  JOB  ...
//stepname EXEC PGM=DTMDCSUn,PARM='start-date start-time end-date end-time'
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=the-CMF-file,DISP=SHR
//SYSUT2 DD DSN=the-output-file-name,DISP=(,CATLG),
//            UNIT=unitname,SPACE=(CYL,(n)),
//            DCB=(RECFM=VB,BLKSIZE=4096)
```

JCL to run the CMF Snap-Shot Utility

file to use as input to the reporting program. Sample JCL is shown here:

start-date

is the Julian date at which to begin copying the file, in the form *yyyyddd* or *-nnn* indicating the relative number of days. See the discussion below.

start-time

is the time at which to begin copying the file, in the form *hh:* or *hh:mm* or *hh:mm:ss*.

end-date

is the Julian date at which to stop copying the file, in the form *yyyyddd* or *-nnn* indicating the relative number of days. See the discussion below.

end-time

is the time at which to stop copying the file, in the form *hh:* or *hh:mm* or *hh:mm:ss*.

the-CMF-file

is the dataset name for the CMF file.

the-output-file-name

is the name of the file to be created for input to the reporting program.

n

Is the size, in cylinders, of the output file that will contain the selected data.

All parameters for specifying start and stop dates and times are optional:

- If no parameters are specified, the entire CMF file is copied.
- If the end date and time are omitted, the current date and time are assumed.
- If either start or end time is omitted, the entire day's data is copied.

Formats for Start and End Dates

The start and end date parameters can be specified in two ways:

- As a Julian date, expressed as *yyyyddd*.
- As a count of days relative to the current date, indicated by specifying a 1-3 digit number prefixed with a minus sign (-). Using this method, the numeral 0 (zero) is a special case specifying the current date. It does not require the leading minus sign.

The specification for both start and end dates must use the same format.

Format for Start and End Times

Both start and end times are optional. If present, you can specify times as one of:

- ***hh:***, where *hh* is a 1-2 digit number from the range 0-23 representing the hour using a 24-hour clock. The trailing colon (:) is required to distinguish a time from the special relative date 0, as described above.
- ***hh:mm***, where *hh* is as described above, and *mm* is a two digit number from the range 00 to 59, indicating the number of minutes after the hour.
- ***hh:mm:ss,*** where *hh:mm* is as described above, and *ss* is a two digit number from the range 00-59, indicating the number of seconds after the minute.

Examples

Consider these examples:

- PARM="0" selects today.
- PARM="-1 -1" selects yesterday.
- PARM="-1" selects yesterday and today.
- PARM="96250" selects from September 6, 1996, up to today.
- PARM="96250 15: 96257 17:30" selects from 3 PM September 6, 1996, up to 5:30 PM September 13, 1996.

Notes

- DTMDCSUn can be run as a separate job prior to running the reporting program, or as the first step of the reporting job.
- Please note that this utility does not remove data from the CMF file. Because of the wrap-around technique used to do the recording, this is not necessary. The data is simply copied to the output file.
- At the end of the run, the utility prints a summary of the records copied to the output file.

The Reporting Program

The DCS Reporting Program is DTMDCSR**n** (where **n** represents the current ThruPut Manager release number). It produces two reports:

1. The Job Analysis Report.
2. The Dataset Analysis Report.

This section of the chapter explains the input data requirements and considerations. It also documents the selection capabilities available with this program.

The Input File

The input data file for the reporting program is created with the Snap-Shot utility that was discussed in the previous section. The file must be sorted before it can be processed by the reporting program. Before a sample of the job stream is shown, some considerations are discussed.

Depending on how you collect the data, it is possible to end up with duplicate records in your input. The reporting program recognizes the situation and discards any record for which an identical copy has already been read. ***That is, duplicate input records do not cause problems for the reporting program***.

Duplication of records can occur when you use the following techniques:

1. You choose to do multiple runs of the Snap-shot utility, each run creating an independent output file. ***You then concatenate all these files as input to the sort***.
2. You choose to do multiple runs of the Snap-shot utility to the same output file with a disposition of MOD. ***This file is then used as the input to the sort program***.

Both techniques are valid, and the reporting program takes care of any duplicate records.

The Reporting Program JCL

An example of the JCL needed to run the reporting program is as follows:

```
//jobname  JOB ...
//*  COPY RECORDS FROM CMF FILE
//EXTRACT  EXEC PGM=DTMDCSUn
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=the-CMF-file,DISP=SHR
//SYSUT2   DD DSN=the-output-file-name,DISP=(,CATLG),
//            UNIT=unitname,SPACE=(CYL,(n)),
//            DCB=(RECFM=VB,BLKSIZE=4096)
//* SORT CMF RECORDS
//SORT    EXEC PGM=SORT,PARM='SIZE(MAX)'
//SYSINDD *
    SORT FIELDS=(27,1,CH,A,29,1,CH,A,33,10,CH,A,28,1,CH,A)
/*
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=sysut2-name,DISP=OLD
//SORTOUT  DD DSN=&&SORTOUT,disp=(,PASS),UNIT=name,
//            SPACE=(CYL,(n)),
//            DCB=(RECFM=VB,BLKSIZE=4096)
//SORTWK01 DD UNIT=name,SPACE=(CYL,(n))
//SORTWK02 DD UNIT=name,SPACE=(CYL,(n))
//SORTWK03 DD UNIT=name,SPACE=(CYL,(n))
//* REPORT GENERATION
//REPORT   EXEC PGM=DTMDCSRn
//SYSINDD *
      your control statements for report selection
/*
//SYSPRINT DD SYSOUT=*
//DCSRIN   DD DSN=&&SORTOUT,DISP=(OLD,DELETE)
//DCSRJRPT DD SYSOUT=*
//DCSRDRPT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
```

Pattern JCL to run the Reporting Program

The Report Selection Options

Report Selection

The first level of selection is the Report Selection. You can select which one of the two reports you want the reporting program to print by including or excluding its corresponding DD statement in the JCL:

- If a //DCSRJRPT DD is included, the Job Analysis Report is produced.
- If a //DCSRDRPT DD is included, the Dataset Analysis Report is produced.

You can include both in a single run.

Job Analysis Report Selection Options

The following selection options are available with this report:

- The number of lines per page.
- Minimum contention duration. For example, you might not want to clutter your report with any contention occurrence that did not last at least 10 minutes.
- The beginning date for data to be considered.
- The beginning time for data to be considered.
- The end date for data to be considered.
- The end time for data to be considered.
- The job name.
- The job identification pattern. An identification character string can be assigned to each job in JAL to be used here for selection. This is discussed later.

These selection criteria are specified as SYSIN input records, one selection parameter per record. Their syntax and effects are now discussed in detail.

`JOB_LINES=n`

Specifies the number of lines per page.

n must be greater than 10. The default value is 60.

If you do not want page breaks specify 0.

`JOB_NAME_PATTERN=pattern`

Specifies a job name pattern to be used in the selection of jobs to be included in the report.

JOB_RECORD_PATTERN=pattern

The key to the reporting selection capabilities is provided by the job record id facility. At JAL time, you can assign a record id to each individual job when you request DCS RECORDING. The record id is an arbitrary character string containing up to 16 characters of your choosing. The character string substitution capabilities of JAL are available to you. This allows you to insert Job Descriptors, such as account, or any hard coded string.

For example, you might want to insert string identifying categories such as 'PRODUCTION' or 'PAYROLL' and/or the type of service requested. At report selection time you could code:

```
JOB_RECORD_PATTERN=PAYROLL*
  or
JOB_RECORD_PATTERN=*CLAIM*
```

Only the jobs that contain a matching string are included in the report. This represents the highest selection level. That is, before any other criteria are applied, the job must satisfy the record id pattern specified with this parameter.

JOB_MIN_TIME=hh:mm:ss

Specifies the minimum contention duration for a job to be included in the report. This represents the sum of the values in the OCCURRENCE/FOR column in your report.

hh:mm:ss represents the total time interval below which you do not want the job included in your report. The default value is 1 minute.

When specifying the time, you can omit lower-order components that are not required.

JOB_DATE_LO=yymmmdd

This represents the earliest date for data to qualify. Any data that was generated earlier than this date is not reported.

You can omit the year, month, or day component. This results in a wildcard effect. For example, if you code 94FEB, any record created in 1994 during the month of February is included in the report. If only two digits are included, they are assumed to be the day.

JOB_DATE_LO=-nnn

This is an alternate form for specifying the earliest date for data to qualify. You can specify up to three digits indicating a *relative* date. For example, -3 indicates three days prior to the current date. To specify the current day's data, you can use -0.

JOB_DATE_HI=yymmmdd

This represents the latest date for data to qualify. Any data that was generated later than this date is not reported.

You can omit the year, month, or day component. This results in a wildcard effect. If only two digits are included, they are assumed to be the day.

JOB_DATE_HI=-nnn

This is an alternate form for specifying the latest date for data to qualify. You can specify up to three digits indicating a *relative* date. For example, -3 indicates three days prior to the current date. To specify the current day's data, you can use -0.

JOB_TIME_LO=hh:mm:ss

The time specification operates within the date boundaries specified with the JOB_DATE parameters. It represents the earliest time for data to qualify. It is expressed as a 24-hour time specification. You can omit lower-order components that are not required.

Any data that was generated at an earlier time is not reported.

JOB_TIME_HI=hh:mm:ss

The time specification operates within the date boundaries specified with the JOB_DATE parameters. It represents the latest time for data to qualify. It is expressed as a 24-hour time specification. You can omit lower-order components that are not required.

Any data that was generated at a later time is not reported.

Dataset Analysis Report Selection Options

The following selection options are available with this report:

- The number of lines per page.
- Minimum contention duration. For example, you might not want to clutter your report with any contention occurrence that did not last at least 10 minutes.

- The beginning date for data to be considered.
- The beginning time for data to be considered.
- The end date for data to be considered.
- The end time for data to be considered.
- Dataset name matching.

These selection criteria are specified as SYSIN input records, one selection parameter per record. Their syntax and effects are now discussed in detail.

DATASET_LINES=n

Specifies the number of lines per page.

n must be greater than 10. The default value is 10.

If you do not want page breaks specify 0.

DATASET_NAME_PATTERN=pattern

Allows you to select the datasets you want the report to include. The standard dataset matching capabilities of ThruPut Manager are allowed here.

DATASET_MIN_TIME=hh:mm:ss

Specifies the minimum contention duration for a dataset to be included in the report. This represents the INTERVAL FOR column in your report.

hh:mm:ss represents the time interval below which you do not want the job included in your report. The default value is 1 minute. You can omit lower-order components that are not required.

DATASET_DATE_LO=yymmmdd

It represents the earliest date for data to qualify. Any data that was generated earlier than this date is not reported.

You can omit the year, month, or day component. This results in a wildcard effect. For example, if you code 94FEB, any record created in 1994 during the month of February is included in the report. If only two digits are included, they are assumed to be the day.

DATASET_DATE_LO=-nnn

This is an alternate form for specifying the earliest date for data to qualify. You can specify up to three digits indicating a *relative* date. For example, -3 indicates three days prior to the current date. To specify the current day's data, you can use -0.

DATASET_DATE_HI=yymmmdd

It represents the latest date for data to qualify. Any data that was generated later than this date is not reported.

You can omit the year, month, or day component. This results in a wildcard effect. If only two digits are included, they are assumed to be the day.

DATASET_DATE_HI=-nnn

This is an alternate form for specifying the latest date for data to qualify. You can specify up to three digits indicating a *relative* date. For example, -3 indicates three days prior to the current date. To specify the current day's data, you can use -0.

DATASET_TIME_LO=hh:mm:ss

The time specification operates within the date boundaries specified with the DATASET_DATE parameters. It represents the earliest time for data to qualify. It is expressed as a 24-hour time specification. You can omit lower-order components that are not required.

Any data that was generated at an earlier time is not reported.

DATASET_TIME_HI=hh:mm:ss

The time specification operates within the date boundaries specified with the DATASET_DATE parameters. It represents the latest time for data to qualify. It is expressed as a 24-hour time specification. You can omit lower-order components that are not required.

Any data that was generated at a later time is not reported.

MAX_DATASETS=nnnnn

Specifies the maximum number of datasets for this report. This can be a number in the range 1-99999. The default value is 4096.

MAX_HOLDERS=nnnnn

Specifies the maximum number of dataset holders for this report. This can be a number in the range 1-99999. The default value is 4096.

MAX_JOBS=nnnnn

Specifies the maximum number of jobs for this report. This can be a number in the range 1-99999. The default value is 4096.

Facilities Summary

DCS Initialization Statements Refer to *Base Product: System Programming Guide*		
Statement	**Description**	**Chapter**
FILE CMF	Specifies the dataset name of the file used to hold Dataset Recording information.	2

Chapter 9. MIM and GRS Considerations

This section describes the DCS implementation considerations in environments running GRS or the Multi-image Integrity component of MIM.

Introduction

There are a number of considerations when you implement DCS in a GRS or MIM Multi-image Integrity environment. This chapter describes the implications and tasks required to implement DCS.

GRS (ONLY) Environments

As you would expect, DCS treats a dataset enqueued with scope SYSTEM as "local" and a dataset with the scope SYSTEMS as "global". As a result, nagging for SYSTEM datasets is restricted to the local system. Only SYSTEMS datasets trigger cross-system nagging. Consideration should be given to the following areas:

- The Control File.
- DCS Private QNAME Propagation.

The required actions are described below.

The Control File

The ThruPut Manager Control File should not be shared across multiple JES2 nodes. Instead, DCS uses XCFM support to communicate across nodes. A discussion of the Control File implementation considerations is included in ***Installation Guide***

DCS Private QNAME Propagation

DCS uses private enqueues to do its job. DCS depends on the propagation of these enqueues for certain functions. You should make sure that your installation's GRS exclusion lists ***do not change the scope to SYSTEM***:

```
QNAME    DTMDCSQX
Scope    SYSTEMS
```

The DTMDCSQX enqueues are used for the following purposes:

- To preserve the integrity of the CMF file.
- Cross-system functions such as nagging.
- Controlling the number of CLAIM jobs waiting in initiators.

Multi-Image Integrity (MIM) Environments

Consideration should be given to the following areas:

- The Control File.
- The DCS SET statement.
- Running DCS with MIM ECMF.
- DCS Private QNAME Propagation.

The required actions are described below.

The Control File

The ThruPut Manager Control File should not be shared across multiple JES2 nodes. Instead, DCS uses XCFM support to communicate across nodes. A discussion of the Control File implementation considerations is included in ***Installation Guide***

The DCS SET Statement

The Multi-image Integrity component of MIM processes all your enqueue requests. During contention situations, it shows as a "resource holder". In those situations DCS needs to know the name of MIM, so it does not misinterpret the situation. A keyword in the DCS SET statement is provided, so you can give the name (or name pattern) of MIM. The format is as follows:

```
DCS SET MIMNAME(name)
```

Here "name" represents a 1 to 8 character job name. This is the name of the started task, or job, under which MIM runs. For a full description of the DCS SET statement, please refer to ***Base Product: System Programming Guide*** "Chapter 2. TMSS Initialization Statements."

Running DCS with MIM ECMF

The ECMF facility of MIM is controlled at the QNAME level. If you are currently running MIM with ECMF=NO for QNAME SYSDSN then you do not have to be concerned with this section.

If you are running MIM with ECMF=YES for the SYSDSN QNAME, then you can get the most out of DCS and MIM working together by doing the following:

- Let DCS have full control of dataset enqueue contention management ***for batch jobs and started tasks*** (static allocation).
- Let ECMF handle ***dynamic allocation conflicts*** that are common with TSO users (and also possible with batch and started tasks).

To get both facilities to work together you must do the following:

1. Install the MIM exit ECMCONXT. The exit is supplied with the DCS distribution tape. A further explanation of this exit is included in "The ECMF Exit Routine" section in this chapter. ***Please note, if you are running CA MIM r11.6, the installation of the ECMCONXT routine is no longer available. The exit function is automatically performed by the dynamic exit routine.***
2. The QNAME statement for SYSDSN should have REPORTAFTER=0, or it should be omitted since the default value is 0. If you specify a non-zero value, there will be an unnecessary wait (equal to the value specified) before DCS receives control.

DCS Private QNAME Propagation

DCS uses private enqueues to do its job. DCS depends on the propagation of these enqueues for certain functions. ***Please ensure that enqueues for the following QNAME are propagated***:

```
QNAME    DTMDCSQX
Scope    SYSTEMS
```

If you use ALLSYSTEMS as the PROCESS mode in your GDINIT statement, the enqueues will be propagated by default. If you use the SELECT mode, you must code the following QNAME statement:

```
QNAME=DTMDCSQX SCOPE=ALL
```

The DTMDCSQX enqueues are used for the following purposes:

- To preserve the integrity of the CMF file.
- Cross-system functions such as nagging.

- Controlling the number of CLAIM jobs waiting in initiators.

The ECMF Exit Routine

MIM provides you with an exit point so the installation can alter the way ECMF handles conflicts. DCS makes use of this facility so it can extend the dataset contention management capabilities of ECMF. The name of the exit is ECMCONXT. Its exact purpose and how to implement it from a MIM perspective is documented in the ***Multi-image Integrity System Guide.*** The installation details can be found in the ***Installation Guide***.

To activate this routine, you should code a SETOPTION command with the following values:

```
SETOPT MIM ...,EXIT=(ECMCONXT,STATUS=ACTIVE,LOAD=ECMCONXT)...
```

A ready-to-implement exit routine is provided with the DCS distribution tape. This routine has been implemented in a way that you can easily incorporate your existing routine, if you are already using this exit. The name of the routine is DTMCONXT.

☞ ***Note: Please note, if you are running CA MIM r11.6, the installation of the ECMCONXT routine is no longer available. The exit function is automatically performed by the dynamic exit routine.***

Chapter 10. Dataset Name Pattern Matching Rules

This chapter describes the rules for specifying dataset name patterns.

String Pattern Matching for Dataset Names

The dataset name pattern format for JECL is identical to the format that is used in DAL/JAL. The rules are repeated here, so you do not have to refer to another manual.

Rules

1. A question mark indicates that its position in the pattern is to be ignored. Any character satisfies the match, including a blank. *There must be a character* in that position. It works at the *qualifier level*.
2. An asterisk indicates that any number of characters can be substituted for it in the dataset name, including blanks or the absence of any character. It works at the *qualifier level*. Note that the asterisk can either be at the beginning of a level, at the end of a level, or in both places. *An asterisk cannot appear in the middle of a dataset level*.
3. An asterisk matches any character or number of characters, but only affects the qualifier in which it is used. The question mark requires the presence of a character but only within the qualifier where it is coded.
4. Coding an asterisk implies that there is a qualifier at that position.
5. The special pattern '**' (two asterisks coded in succession) matches any number of qualifiers, including the absence of a qualifier. This is used in place of *complete qualifiers*.

Examples

Use of Asterisk

An asterisk matches any number of characters, including the absence of a character. For example:

```
AP*.LOAD
```

matches these dataset names:

```
AP123.LOAD      AP.LOAD      APOLD.LOAD
```

but not these dataset names:

```
MY.AP123.LOAD      AP.OLD.LOAD      AP.MYLOAD
```

Note that coding an asterisk implies that there is a qualifier at that position.

For example, the pattern:

```
A.*.B
```

will match these dataset names:

```
A.X.B      A.TEST.B      A.LOADLIB.B
```

but not these dataset names:

```
A.B      A.B.C
```

Use of Double Asterisk

The case where several qualifiers are to be ignored is handled by extending the pattern matching function to include the special pattern '**' (two asterisks coded in succession). Where this sequence is used, it matches any number of qualifiers, including the absence of a qualifier. For example:

```
**.LOAD
```

matches these dataset names:

```
AP123.LOAD      AP.OLD.LOAD      LOAD
```

but not these dataset names:

```
AP123.LOAD.OLD      AP123.MYLOAD
```

Use of Single Asterisk and Double Asterisk

To check for dataset names having AP as the first two letters of the first qualifier, the pattern:

```
AP*.**
```

will match these dataset names:

```
AP900.LOAD      AP.MONTHLY.QUOTA      AP500
```

Index

Index

Special Characters

A

C

D

E

K

L

M

N

O

P

Q

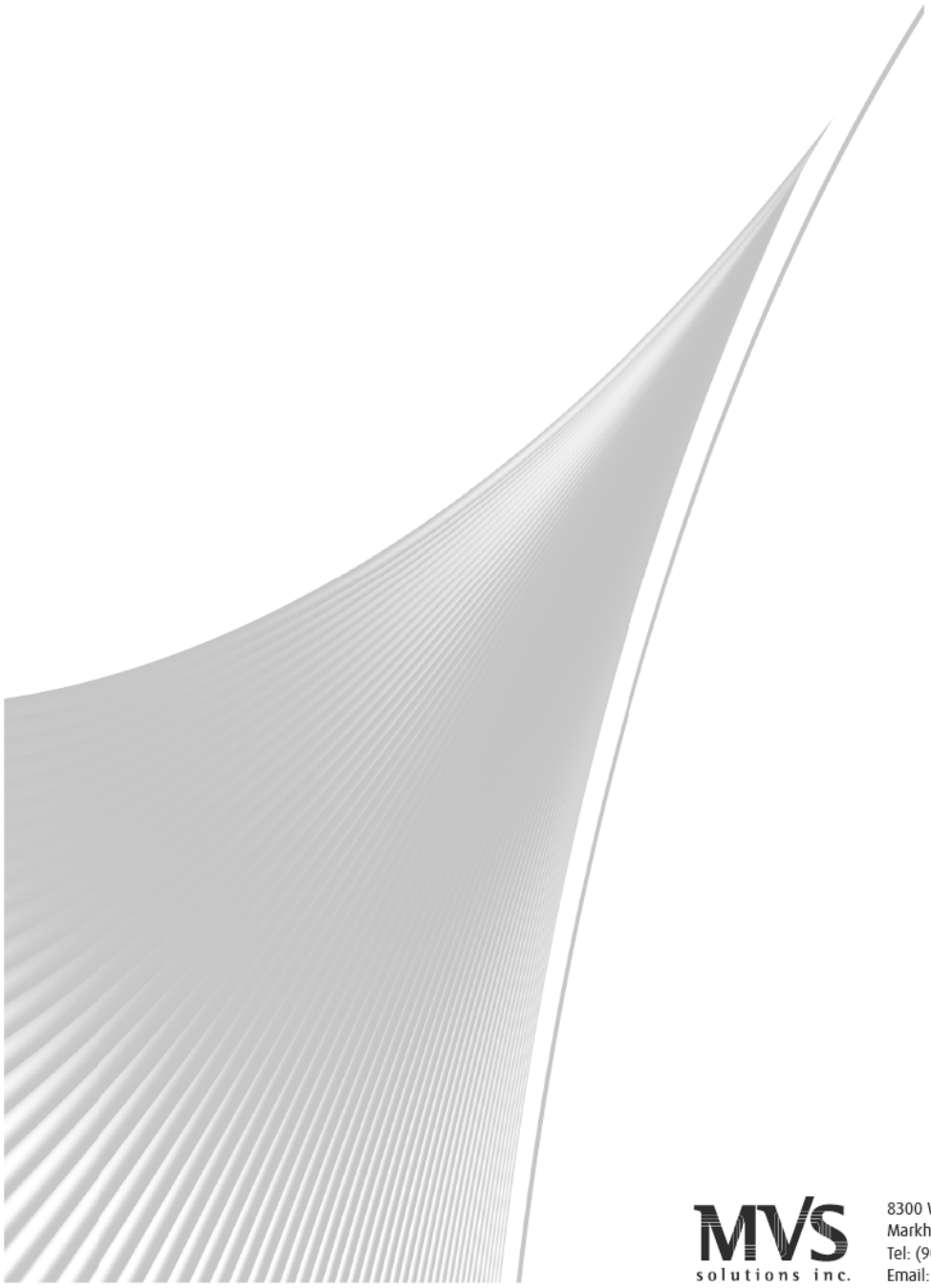
R

S

T

U

W



MVS solutions inc.

8300 Woodbine Avenue, 4th Floor
Markham ON Canada L3R 9Y7
Tel: (905) 940-9404 Fax: (905) 940-5308
Email: marketing@mvssol.com www.mvssol.com