

Application Recovery and *JobQGenie*

Author: Chris Hird

Shield Advanced Solutions (Canada) Ltd

December 2006

Overview

This white paper looks at how to use the data and interfaces provided by JobQGenie to recover your applications after a system failure. The techniques used are general in nature due to the wide variety of scenarios which occur when a system fails. However it should give the reader a better understanding of the role of JobQGenie in this type recovery situation.

There are a number of options for the providing High Availability on the System i5. These range from IBM provided technology to ISV generated products. The technology you choose to implement for HA is made for a variety of reasons. The main purpose however, of any solution is to provide the users with access to the application for as much time as possible with little or no downtime. As with any system the i5 still requires a level of downtime to allow certain activities to take place such as backups and PTF installation etc. These are known as 'planned outages' because you know they have to be carried out and have time to schedule the related activity with minimal downtime for the users. The biggest concern for customers however, is the 'unplanned events' where they have no control over when it will occur. The System i5 reliability has improved over the years which has reduced the possibility of these events, but they still occur and often create major problems, even when a customer has an action plan in place for such eventualities.

One of the solutions which many of the customers choose to protect their systems is the use of a HA product such as Lakeview, Data Mirror, iTera, Traders or Vision Solutions High Availability Suites.

In a planned event the data and object replication switch-over requires little, if any, administration and management due to automated switch processes. However when an unplanned event occurs the time it takes to recover the application and its data can be significant.

A major client implementing a HA solution saw a problem with the products lack of ability to capture the job data. They understood that the HA product did a great job of keeping the two systems in almost perfect synchronization, but it had no concept of the data structure as it applies to the application. One of the problems they faced after an unplanned outage was understanding the data state. They had literally thousands of jobs a day being submitted, this generated most of the data activity in the database, they had no way of knowing what state the jobs were in at any one point in time. *JobQGenie* was developed to help solve that problem for the customer.

The OS provides a notification technology which allows a program to watch the progress of a job through the stages of being loaded to the Job Queue, becoming active and then ending. Initially the information was very sparse and the effort required to collect all of the relevant information was significant. Now the data being provided makes it a much easier to collect and store for replication to the target system.

Using this information and either program based data repair or the RMVJRNCHG commands you can bring the data back to a state where the jobs which were active can now be reloaded back onto the job queues and run again. Any jobs which were sitting on a job queue can also be reloaded and run in the same manner as they were on the source system. The following is a guide to some of the functionality

which JobQGenie provides to help you determine the position the data should be rolled back to. It does not provide the remove capability, but does show the user where they need to get back to in order to start the processes again. It is flexible process which can be adapted for each application. Here we show how the product can provide the information required for an example of application.

Finding a Start Point

The first thing we have to do is identify which jobs were running at the time of the system failure. These are important because they should be resubmitted to allow them to complete successfully before jobs which were on the job queue can be released. The other jobs to review are the jobs which ended in error, this is because these jobs could have ended in error due to the system starting to go down. It could be that they need resubmitting so they can complete successfully.

Apply all Data on the Target.

Before we look into the data which has been collected we need to ensure that the replication process and JobQGenie (if active), have been ended. It is also important that the replication processes end after all of the application and job data have been applied. This will ensure the JobQGenie data and the application data are in sync with each other.

To ensure JobQGenie is down take option 4 from the operations menu, this will show a list of active jobs in the JobQGenie subsystem. The Jobs will only be active if you have been collecting the data on the 'target system' which was being replicated to the 'source system'. The Replication software will have its own interface to allow you to determine if the data has been applied and the jobs ended correctly.

Check for Jobs in Active Status

Next identify the jobs which are showing as *ACTIVE in JobQGenie. They show as *ACTIVE because the files only get updated once the job has ended and the replication software has carried the new state over to the target system. The job data should be in sync with the application data. The only time this would not be true is if JobQGenie was down or the replication process for the JobQGenie data was down.

Take note of jobs which ended recently and in error. This can be done by sorting the data to show recent jobs with abnormal end states. These jobs may not require resubmission; they may have been resubmitted already and completed normally. If you allow users to load jobs themselves, using the data collected by JobQGenie will allow you to contact those users and confirm if they need to be resubmitted.

From the Recovery Menu take option 1 which is the 'Work with Batch Jobs' option. This will prompt you for the sequencing information, the *STATUS option will show the Active jobs at the top of the list. Then you will be prompted for the environment to use. If you have set JobQGenie up with a single environment for each system just select the environment which relates to the source system. This will give you a screen as shown below.

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Work with Batch Jobs
System: SHIELD1
Environment . . . : JG_BASE
Currently viewing : Active / Failed jobs
Ordered by . . . : Job Status

Type options, press Enter.
  3=Journal Info  5=Display  6=Resubmit  7=Submitted jobs
  8=Upstream jobs

  Opt  Job  Job  User  Load  Load  Job  Sub  Status
     ID  Name  Name  Date  Time  Type  Type
  = 068010 NORMAL CHRISH 12/18/06 10:04:49 B Active
  - 068011 SYSSTSJOB CHRISH 12/18/06 10:04:49 B Active
  - 068012 BADCMDJOB CHRISH 12/18/06 10:04:49 B Active
  - 067558 BCHFLOOD1 CHRISH 12/18/06 09:34:25 B Ended
  - 067559 BCHFLOOD2 CHRISH 12/18/06 09:34:26 B Ended
                                     More...

Parameters for options or command
===>
F3=Exit      F4=Prompt  F5=Refresh  F9=Retrieve  F11=Job Start Info
F12=Cancel   F14=Show jobs in Jobq  F16=Resequence  F24=More keys
```

Figure 1 List of active jobs

You will notice that the above screen shows 3 jobs as being active, these are the jobs which we are interested in recovering.

Identify the Data which Requires Clean Up

Once we have the jobs listed, identify the data which has been applied for each of those jobs. Our test environment has a very complex job load structure which provides a complex recovery scenario. This is not normal and is probably not comparative to your environment, however the process is the same, regardless of complexity. Identify the first job to review, which is still active. You can do this either by 'Job Start Date and Time' or the Job ID associated with the job (i5OS assigns the job ID as the job is loaded onto the job Queue, lower numbers generally mean the job was loaded first). Take option 3 against the first job you have identified and this will produce a list of the journals which have entries related to that job. This may take a little time depending on your system configuration.

Take note of the Start and End sequence number (F11 will cycle through the screens which display the start and end information. As this is the first job we are interested in (lowest JOBID) we know that the

first sequence number is most important, the last sequence number could be replaced in a subsequent job.

Note:-

A job may not have data associated with it which is being replicated to the target system. This will output a screen as below. This is not an error and should not cause concern, we only interrogate Remote journals on the target system, this may have run updating data which was not replicated to the target system. It is therefore not necessary to remove its data or resubmit the job again.

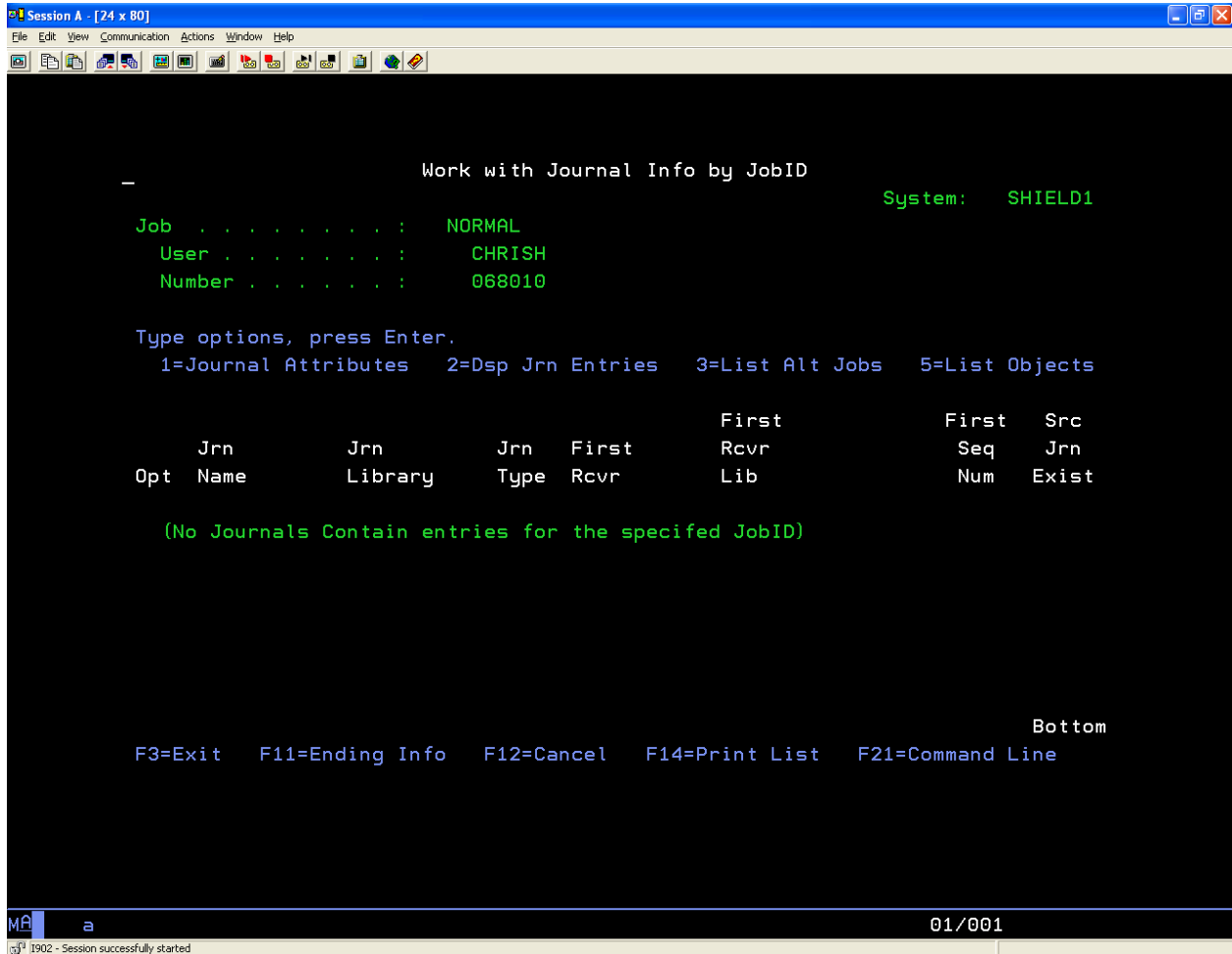


Figure 2 Empty Journal info

The following screen shows a job which has data updates in the remote journal object. This is the data which has to be removed to allow the job to run again and before the system can accessed by the users.

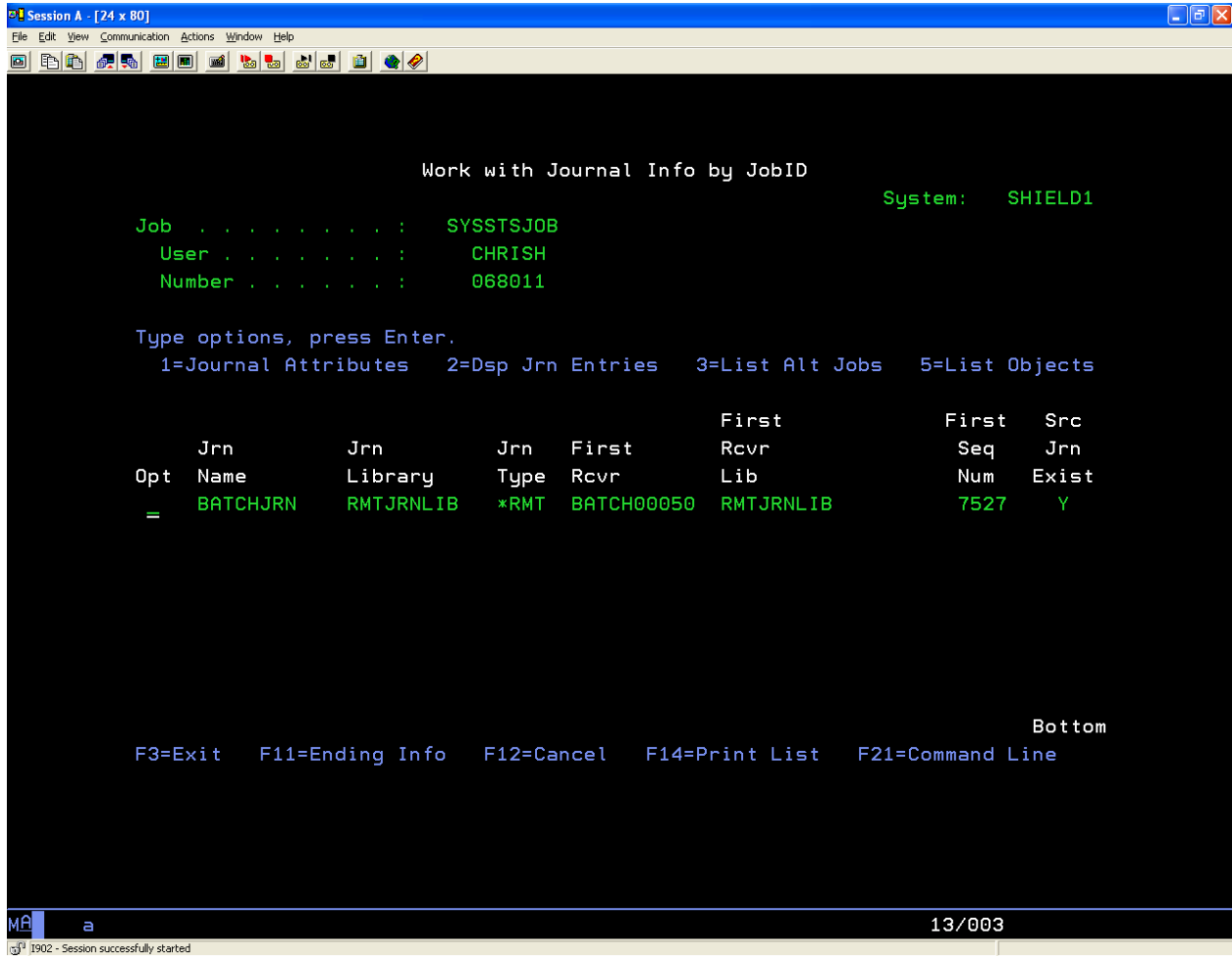


Figure 3 Journal info by JobID

We can see from the above screen that this job has created data updates starting at sequence 7527 in the journal, but it may not be the true start, as we have to make sure that any jobs which may have ended did not have data applied after the start point of this job. To identify the jobs which have entries that occur between our start and end point for this job, we have to take option 3 (List Alt Jobs). This provides us with a list of all of the jobs which were running at the same time as our initial job.

Note:-

The RMVJRNCHG command takes a start entry and an end entry to remove the changes between, it will not take account of any job stream responsible for those updates. So when we run the command we have to make sure that no partial job data updates get removed.

Below is the output from our test system. You will see we have an extensive list to review. If your job queues are single streams they will not have this kind of complexity, each job will end before the next one starts. Another configuration which could create this level of complexity is where you have multiple job streams all updating the same database. Again this is very uncommon but could occur.

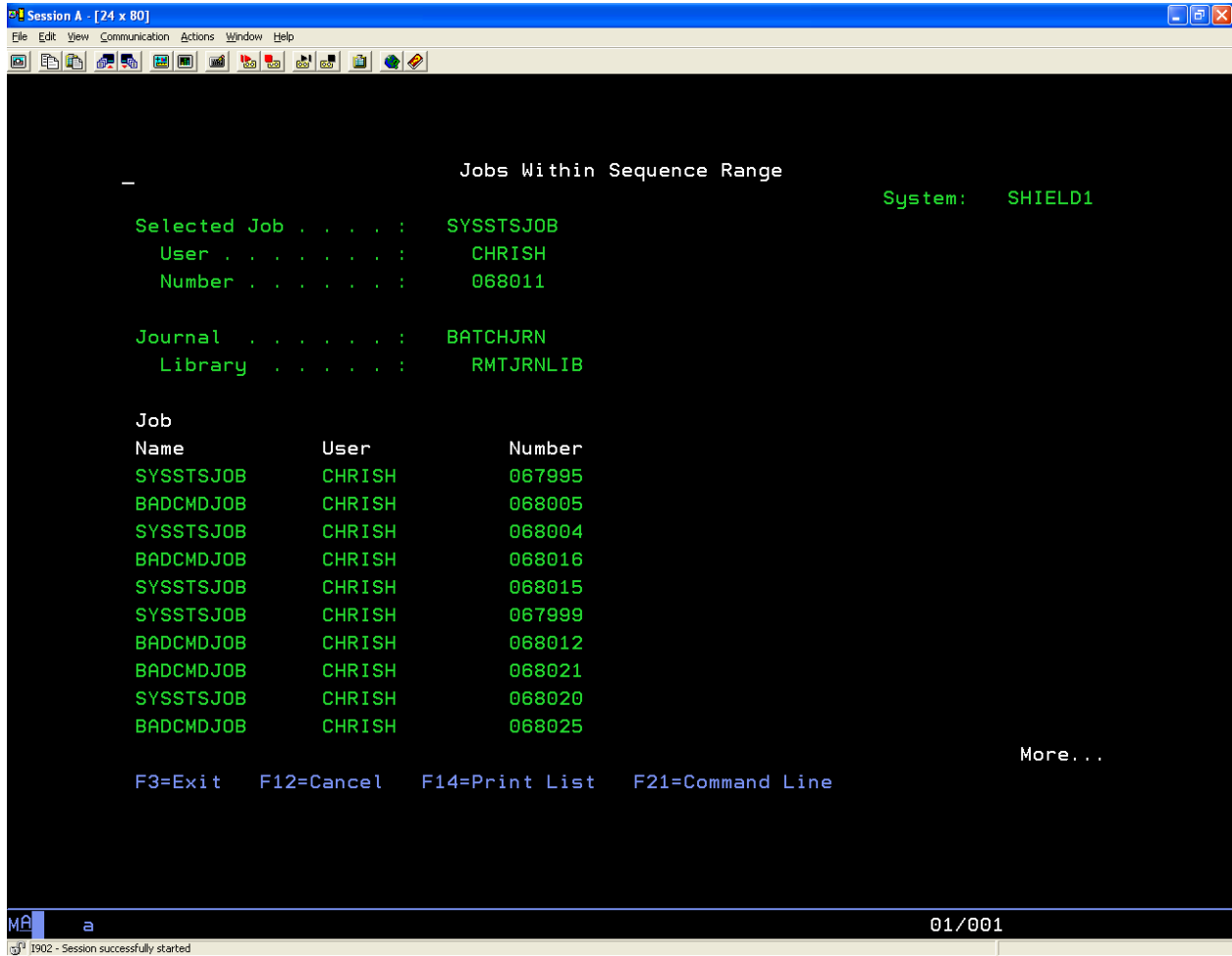


Figure 4 Additional jobs within sequence range

Because we know that each JOBID is generally reflective of its start time we can assume that the earliest job we have (067995 in this case) is where we have to start from. We need to review the job information as we have for this one; this should lead us back to the first entry we have to remove. We also know that the last entry (ours is beyond 068025) should be the last job to create data updates. So again, we would investigate where the last entry would exist in the data stream. Eventually you should end up with a list of all of the jobs which have to be resubmitted plus the Sequence numbers to use for the RMVJRNCHG command.

For the sake of expediency we will assume that the last job which needs to be rolled back is 068012. We therefore need the last sequence number that job applied 7606.

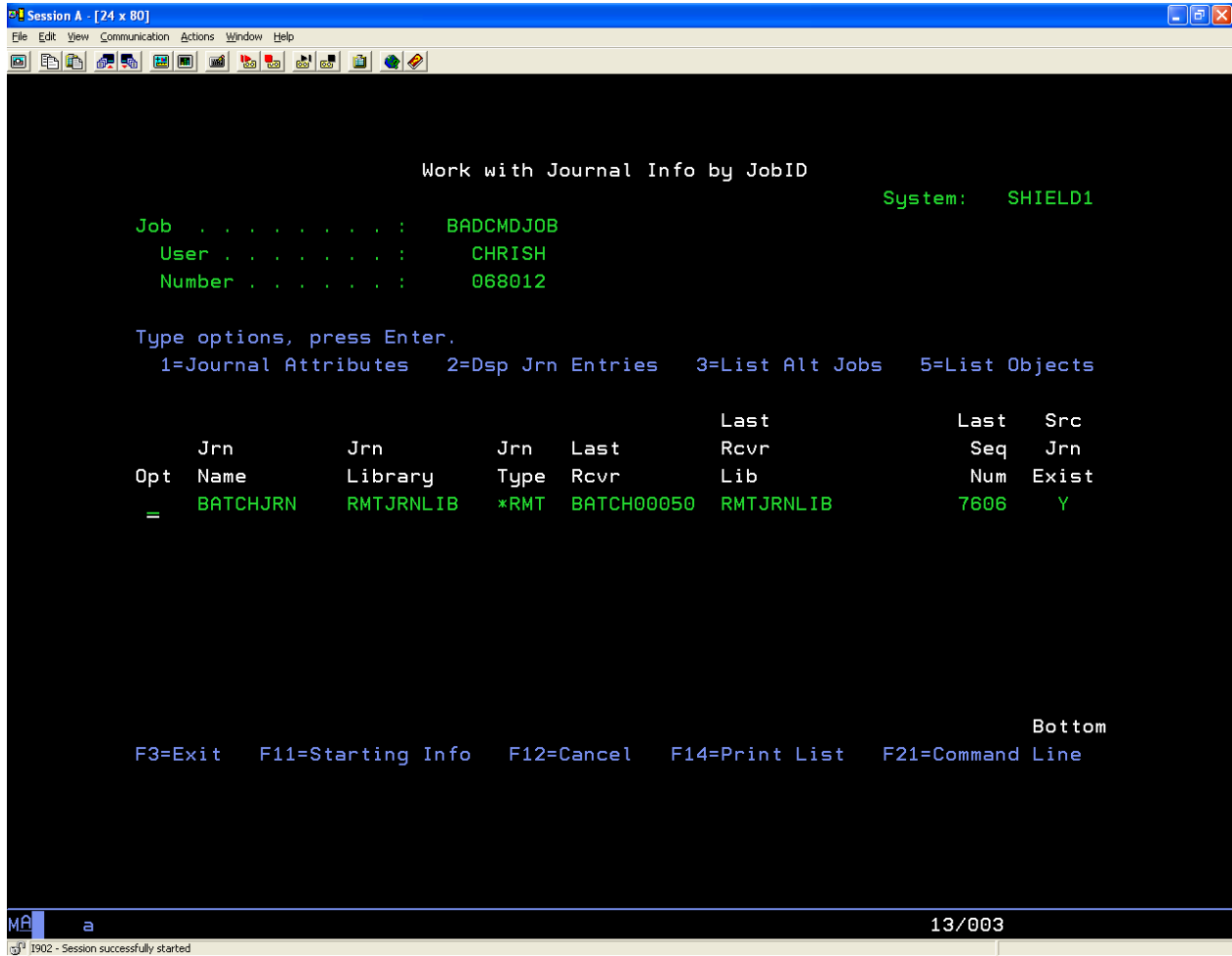


Figure 5 Last sequence number

Check for Additional Submitted Jobs

The job we have identified could have submitted other jobs that affect the data stream. They could have also submitted jobs which affect the data stream. This is important because when we submit our job, it will re-submit the jobs it originally submitted. This could cause the data to be updated or added twice. To avoid this we should now review each job we have identified and check the jobs it submitted.

Note:-

Taking the option to view the up-stream jobs is another alternative but there is a chance that you could miss jobs that have not started yet but have been submitted.

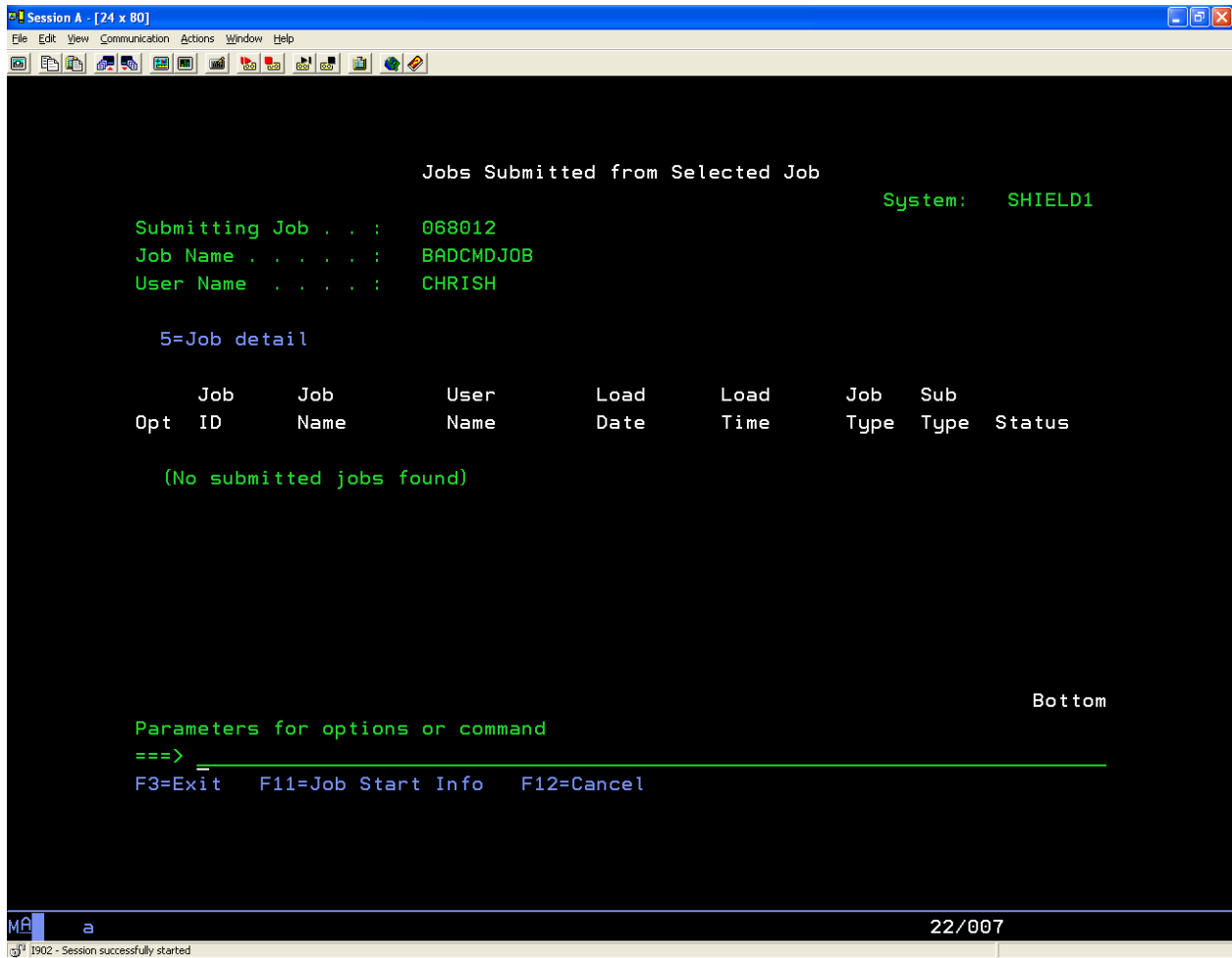


Figure 6 Submitted jobs detail

Jobs on the Job Queue

Before we start to resubmit the jobs, we should look at which jobs were still sitting on the job queue. Pressing F14 from the list of Active/Submitted jobs will display a list of the jobs which were still waiting on the job queue when the system ended. Again an important check here would be the submitting job, we don't want to submit the job twice. We know this job won't have submitted any other jobs and its data is not in the journal yet, so we don't have to detail anything about that.

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Work with Batch Jobs
System: SHIELD1
Environment . . . : JG_BASE
Currently viewing : Jobs in job queues
Ordered by . . . : Job ID
Type options, press Enter.
  3=Journal Info  5=Display  6=Resubmit  8=Upstream jobs

  Job      Job      User      Load      Load      Job      Sub
Opt  ID      Name      Name      Date      Time     Type     Type     Status
  _  068360  QDFTJOB  CHRISH    12/18/06  10:53:57  B        B        JobQ

Parameters for options or command
===>
F3=Exit      F4=Prompt  F5=Refresh  F9=Retrieve  F11=Job Start Info
F12=Cancel   F14=Show Active/Failed jobs  F16=Resequence  F24=More keys

Bottom
MA a 21/007
i902 - Session successfully started

```

Figure 7 Jobs on Job queue list

Resubmit the Jobs

Now we have all of the information we need to start recovery. We know the first job and last job that we need to resubmit. But the sequence we load the jobs is also important. In addition, as previously mentioned, we only want to re-submit jobs which have updated data that has been replicated.

Before you start to reload the jobs onto the job queues you have to use the information you have collected to remove the data from the database. To do this use the RMVJRNCHG command or if you have one, your application tools for such events and remove the data for the jobs being re-submitted .

Now we need to select the first job with data updates and resubmit it to the job queue. Take option 1 from the Recovery Menu again and select a sequence of *JOBID, this will give a list of jobs which reflect the time they were submitted. If you have doubts about the submit time you can always review the 'Load Date and Time' or the 'Start Date and Time' and resubmit jobs based on those. We don't however want to submit every job between the first one we identified and the last one, as you'll recall, some jobs will have been tracked, but they did not affect any data on the target system; so as we look at each job, we need to review the Journal information (option3) before selecting option 6 (resubmit).

Taking Option 6 against the first job and pressing enter will present you with a confirmation screen.

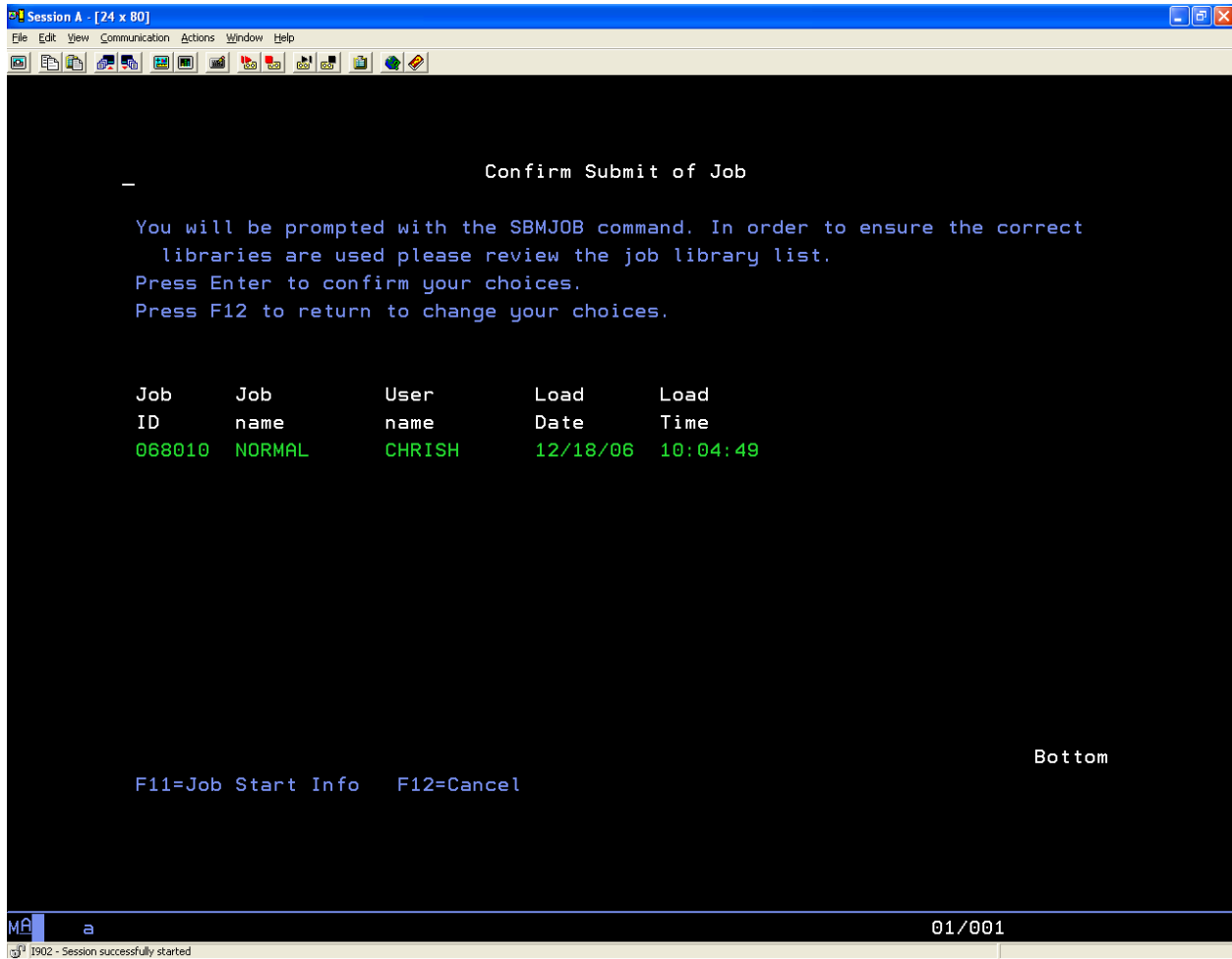


Figure 8 Confirm Submit Job

Pressing F11 will cycle you through some of the job data which has been collected; it does not show you all of the information but does contain the main parts. Pressing Enter will prompt the SBMJOB command with all of the fields filled in with the collected data. One of the things we do check as part of the submission is the library list which was used on the source system and the library list which is in effect on this system. You will notice that the initial library list is already filled in for you.

If the resubmission was carried out successfully a message is sent out confirming this and also informs the user that the resubmission flag will not be set in the files if this is identified as a *REMOTE environment.

System is Now Ready for User Access

Once you have cleaned up all of the data and resubmitted the necessary jobs, you can start the application services again and allow the users back onto the system. The data should be in the same state as it was when the users last accessed the source system unless they have interactive jobs which allow data updates to be placed before the job completes.