ECS Configuration Change Request

1. Originator	2. Log Date:	3. CCR #:	4. Rev:	5. Tel:	6. Rm #:	7. Dept.
Gary Lake	4/13/00	00-0411	-	(301) 614-5751	N126-6	Sys Eng

8. CCR Title: Promote DPR Metrics Table and stored procedure to OPS.

9. Originator Signature/Date	10. Class	11. Type:	12. Need Date: April 21, 2000
	II	CCR	

13. Office Manager Signature/Date	14. Category of Change:	15. Priority: (If "Emergency" fill in Block 28).
	Initial ECS Baseline Doc.	Routine

16. Documentation/Drawings Impacted:	17. Schedule Impact:	18. CI(s) Affected:
Planning and Data Processing Database Design 311-CD-503-001	None	pdps

19. Release Affected by this Change:	20. Date due to Customer:	21. Estimated Cost:
5A		None - Under 100K

22. Source Reference: ☐NCR (attach) ☐Action Item ☐Tech Ref. ☒GSFC ☐Other:
GSFdd00111

23. Problem: (use additional Sheets if necessary)
Promote the table, "PlDprMetrics" in the pdps database to OPS mode with the stored procedure, GetLatestDPRStatistics and executable file GetLatestDPRStatistics to OPS mode.

24. Proposed Solution: (use additional sheets if necessary)
One new table will be added to the pdps database to store data on the total number of DPRs completed during one hour time periods, and the totals for the corresponding CPU usage parameters. The new table is expected to receive about 3MB of data per year (if the number of PGE's doubles, then the storage rate will double to 6 MB per year). A stored procedure will be written in Sybase's Transact SQL to select sums of values from the PlDprCompletion table in the pdps database for intervals starting one hour prior to the present and ending at the present. The results of this query will be stored in a new pdps database table with

25. Alternate Solution: (use additional sheets if necessary)
A similar solution can be implemented to that which is proposed, however, instead of storing the data in a table in the pdps database, the data may be stored in a newly created database. The advantage to this approach is in the complete separation of the new code from existing code. The disadvantage is that the newly created database must reside on g0pls02_srvr, otherwise, g0pls02_srvr and

26. Consequences if Change(s) are not approved: (use additional sheets if necessary)
It will not be possible to create a user interface that displays DPR data to the production planners who would use this information in scheduling.

27. Justification for Emergency (If Block 15 is "Emergency"):
Intended as DUE info only. No perceived ECS impact.

28. Site(s) Affected: ☐EDF ☐PVC ☐VATC ☐EDC ☒GSFC ☐LaRC ☐NSIDC ☐SMC ☐AK ☐JPL ☐EOC ☐IDG Test Cell ☐Other

29. Board Comments:	30. Work Assigned To:	31. CCR Closed Date:

32. EDF/SCDV CCB Chair (Sign/Date):	Disposition: Approved App/Com. Disapproved Withdraw Fwd/ESDIS ERB Fwd/ECS
33. M&O CCB Chair (Sign/Date): 25 Apr 00	Disposition: Approved App/Com. Disapproved Withdraw Fwd/ESDIS ERB Fwd/ECS
34. ECS CCB Chair (Sign/Date):	Disposition: Approved App/Com. Disapproved Withdraw Fwd/ESDIS ERB Fwd/ESDIS

ORIGINAL

00-0411
2/3

ADDITIONAL SHEET

CCR #: 00-0411 **Rev:** **Originator:** Gary Lake

Telephone: (301) 614-5751 **Office:** N126-6

Title of Change: Promote DPR Metrics Table and stored procedure to OPS.

The following Transact SQL code creates the required database table, stored procedure and function to be executed by cron on an hourly basis.

```
create table PIDprMetrics (pgeName varchar(17) not null,
 HourlySumPgeCpuTime float null,
  HourlySumSystemCpuTime float null,
   HourlySumPgeElapsedTime float null,
   dprIdCount int null,
    SampleStartTime  datetime not null,
     SampleEndTime datetime not null,
 primary key (SampleStartTime, pgeName))


Create Proc GetLatestDPRStatistics
as
insert PIDprMetrics
 select "DPREP",sum(pgeCpuTime),sum(systemCpuTime),sum(pgeElapsedTime),count(distinct dprId), dateadd(hh, -1,getdate()),
getdate()
  from PIDprCompletion
   where dprId like '%Att%' or dprId like '%Eph%'
    and completionDate > dateadd(hh, -1,getdate())
     and completionDate < getdate()

insert PIDprMetrics
 select "PGE01",sum(pgeCpuTime),sum(systemCpuTime),sum(pgeElapsedTime),count(distinct dprId), dateadd(dd, -1,getdate()),
getdate()
  from PIDprCompletion
   where dprId like '%PGE01%'
    and completionDate > dateadd(hh, -1,getdate())
     and completionDate < getdate()

insert PIDprMetrics
 select "PGE02",sum(pgeCpuTime),sum(systemCpuTime),sum(pgeElapsedTime),count(distinct dprId), dateadd(hh, -1,getdate()),
getdate()
  from PIDprCompletion
   where dprId like '%PGE02#%'
    and completionDate > dateadd(hh, -1,getdate())
     and completionDate < getdate()

insert PIDprMetrics
 select "PGE02A",sum(pgeCpuTime),sum(systemCpuTime),sum(pgeElapsedTime),count(distinct dprId), dateadd(hh, -
1,getdate()), getdate()
  from PIDprCompletion
   where dprId like '%PGE02A%'
    and completionDate > dateadd(hh, -1,getdate())
     and completionDate < getdate()

insert PIDprMetrics
 select "PGE03",sum(pgeCpuTime),sum(systemCpuTime),sum(pgeElapsedTime),count(distinct dprId), dateadd(hh, -1, getdate()),
getdate()
  from PIDprCompletion
```

ORIGINAL

```
  where dprId like '%PGE03%'
   and completionDate > dateadd(hh, -1,getdate())
    and completionDate < getdate()

insert PIDprMetrics
 select "PGE Hourly Totals",sum(pgeCpuTime),sum(systemCpuTime),sum(pgeElapsedTime),count(distinct dprId), dateadd(hh, -
1, getdate()), getdate()
  from PIDprCompletion
   where completionDate > dateadd(hh, -1,getdate())
    and completionDate < getdate()
return
```

When the following function template is completed, the resulting function, perhaps called "GetDPRStatistics", is created. The function name is entered into a record in the appropriate crontab along with the specification for hourly execution by cron.

```
# BE SURE THAT EXEC PRIVILEGES HAVE BEEN PROPERLY GRANTED TO # $DBUSERNAME FOR PROCEDURE
GetLatestDPRStatistics
DBUSERNAME=<user name>
DBPASSWD= <password>

SYBASE= <file path>/sybase
DSQUERY=pdps
$SYBASE/bin/isql -U$DBUSERNAME -P$DBPASSWD
go
exec GetLatestDPRStatistics
go
EOF
```

The following code needs to be inserted into the crontab:

```
0 * * * * <filename>
```

Subsecond response times have been observed for running this procedure.

The following code selects hourly DPR counts from the DprMetric table with the sample start times, end times, and midpoints for the time interval starting seven hours prior to the present to the present time:

```
select "Hourly DPR Count" = dprId, SampleStartTime, "Sample MidPoint" = dateadd(mm, 30, SampleStartTime),
SampleEndTime from DprMetric where SampleStartTime > dateadd(hh, -7, getdate())
```

Each record will require a maximum of about 60 bytes of storage space. Each query should return 6 records. With hourly sampling, 356 days per year, we expect to consume about 3,153,600 bytes (~ 3MB) per year of storage space.

CM01AJA00

ORIGINAL