Memo

To:	CCB
From:	Lance Al-Rawi
Date:	July 14, 1999
Subject:	PIDC 6.2 -CSCI 3: Distributed processing (DACS)
Sponsor:	Greg Beall
CC:	

Abstract

This memo recommends installation of the PIDC 6.2 version of software for CSCI 3, Distributed Processing (DACS), in PIDC Operations. PIDC 6.2 has been assessed and remediated for Y2K compliance following the method described by Beall et al. (1999) and Skov and Jenkins (1999). PIDC 6.2 will form the basis for IDC Release 2.1, which will be delivered, installed and validated at the International Data Centre this autumn.

PIDC 6.2 DACS includes a few minor changes beyond the principle Y2K changes. The *TupleProc* application was ported to *libgdi* to eliminate dependence on the obsolete *libdb30* and related libraries. The Pipeline[1] applications were enhanced to remove an obsolete parameter and enforce mutual exclusivity among two other parameters to solve a problem documented during PIDC 6.1.0 integration testing and operations.

Unit testing uncovered several minor software problems that were introduced during the Y2K remediation. These problems were resolved in time for the PIDC 6.2 release. Integration testing was a complete success in that all integration test procedures, Y2K focused and otherwise, revealed no new problems assignable to the PIDC 6.2 software changes. Several pre-existing problems established and documented during PIDC 6.1 integration testing and operations were observed during the integration test. All observed problems are contained within the current set of outstanding SMRs, some of which are scheduled for resolution during IDC R3 development. There are no outstanding SMRs that must be resolved as a condition of promotion of PIDC 6.2 DACS to operations.

1. *tis_server, tiseg_server, ticron_server*

Statement of Objective

The objective of this proposal is to install a Y2K-compliant version of CSCI 3 software into PIDC operations.

Summary of Proposed Change

The principal change between PIDC 6.1 and PIDC 6.2 is that all time functions in PIDC 6.2 are handled by a standard, Y2K-compliant, time library, *libstdtime*. Additional changes to CSCI 3 software are:

- *TupleProc*[1] was ported to *libgdi*, the standard database access API for all IDC software. As a result of the port, this application is no longer dependent on the obsolete database access libraries: *libd30* and *libdbims*.
- SMR CMR-460, a problem that was identified during PIDC 6.1 integration testing, was resolved. The Pipeline applications were changed to eliminate the obsolete and confusing *logical-dest* parameter. In a related change, the *physical-dest* and *tpcall-dest* parameters were made to be mutually exclusive, thus avoiding the potential for replicated interval enqueues via the two destination routes or channels.
- The CSCI 3 man page set was updated to be consistent with the PIDC 6.2 DACS implementation.

Expected Benefits

This release of software has the benefit of demonstrated effectiveness at operating in the year 2000 as well as operating across the millennium and the leap year boundaries.

The non-Y2K changes will improve the usability and maintainability of the DACS, although they are quite limited in scope with the exception of the significant upgrade in the man page set. The improved man pages will increase user accessibility to the information needed to configure and run DACS.

Possible Risks and Dependencies

Changes between PIDC 6.1 and 6.2 have been limited mostly to changing all time and date calls to use *libstdtime*. Other changes in this release introduce the following risks:

- *TupleProc* has not been well tested given the fairly significant code change resulting from the *libgdi* port. *TupleProc* was not subjected to an integration test since it was not configured to run as part of the PIDC 6.2 system configuration on the PIDC 6.2 testbed. However, since

1. *TupleProc* was not used in the PIDC 6.2 testbed system configuration and as such it has only been unit tested in San Diego.

TupleProc is not used in operations, this risk is quite limited. If *TupleProc* is to be used in PIDC or IDC operations, it should be tested on the PIDC 6.2 testbed in a scheduled manner that would allow one week for a possible San Diego code fix.

PIDC 6.2 and PIDC 6.2 DACS has been extensively tested, as described in the following section. Outstanding problems identified during testing, along with their severity, are shown in Table 2 in Appendix A. None of these problems are considered severe enough to warrant delay of installation of the PIDC 6.2 release into operations.

Summary of Testing

All applications were unit tested before release. This testing included specific Y2K tests. The general integration test plan for PIDC 6.2 is described by Skov (1999). Procedures in this plan that are relevant to this CSCI are the normal live data test, and the Y2K live data test. This testing is summarized for all CSC's in this CSCI in Table 1 in Appendix A.

PIDC 6.2 DACS unit testing was dramatically improved for this release following the recent creation of the San Diego testbed. This evolving testbed system provides a reasonable facsimile of the operational system including simulated data acquisition (e.g., live wfdisc creation). This testbed environment facilitated the relatively easy discovery of several minor software problems that were introduced during the Y2K remediation. These problems were resolved in time for the PIDC 6.2 release and paved the way for a successful integration test period.

Integration testing was a complete success in that all integration test procedures, Y2K focused and otherwise, revealed no new problems encountered at the Y2K transition or due to the PIDC 6.2 software changes. Several pre-existing problems established and documented during PIDC 6.1 integration testing and operations were observed during the integration test. All observed problems are contained within the current set of outstanding SMRs, some of which are scheduled for resolution during IDC R3 development. None of the SMRs are High priority and there are no outstanding problems that must be resolved as a condition of promotion of PIDC 6.2 DACS to operations.

CSCI 3 software operated as expected for the unit testing although several minor problems related to the Y2K code fix were uncovered and fixed in time for the PIDC 6.2 software release. Detailed results are given in the Unit Test Report on file with the MSO test group.

CSCI 3 software operated as expected for the normal live data test. Summarized test procedures and results of the integration test are included in Table 3 in Appendix A and detailed results are given in the Integration Test Report on file with the MSO test group.

CSCI 3 software also operated as expected for the Y2K live data test. High-level Y2K test results are given by Skov and Jenkins (1999). No anomalies assignable to PIDC 6.2 code changes or Y2K processing were encountered in CSCI 3 which accounts for the empty list of outstanding DACS problems in Table 2 in Appendix A. Several pre-existing anomalies established during PIDC 6.1 integration testing were observed during integration testing. Resolution of these pre-existing SMRs is not required for installation of PIDC 6.2 DACS into operations.

Schedule and Plan for Implementation

Installation of PIDC 6.2 will follow the Plan for Implementing PIDC 6.2 into PIDC Operations (Skov, Mangino, Bahavar and Prasad, 1999). This plan allows for a 4-day installation period followed by a one-week break in period prior to stabilizing operations.

Costs and Resources Required for Implementation

It is estimated that implementation of PIDC 6.2 as a whole will require four or five integration and Operations staff members 2-4 days. No special resource requirements are expected for this CSCI.

References

Beall, G., J.R. Bowman, W. Farrell, B. Hedline, R. Jenkins, B. MacRitchie, N. Prasad, D. Salzberg and M. Skov, Systems Design Review: International Data Centre Release 2.1, Technical Report SAIC-99/3001, 1999.

Skov, M., PIDC 6.2 Integration Test Plan, Technical Report CMR-99/10, 1999.

Skov, M. and R. Jenkins, PIDC 6.2: Assessment, remediation and testing for Y2K compliance, Technical Report CMR-99/20, 1999.

Skov, M., S. Mangino, M. Bahavar and N. Prasad, Plan for Implementing PIDC 6.2 into PIDC Operations, Technical Report CMR-99/21, 1999.

Appendix A: CSCI 3 Testing and Release Details

Table 1: Summary of Y2K Assessment and Remediation, and Testing for CSCI 3[1]

CSCI/CSC			Y2K Assessed and Remediated	Unit Tested[2]	Integration Tested on PIDC Testbed	Y2K Tested on PIDC Testbed[3]
3 Distributed Processing						
3.1	Application Services					
	3.1.1	dbserver	yes	yes	yes	yes
	3.1.2	interval_router	yes	yes	yes	yes
	3.1.3	recycler_server	yes	yes	yes	yes
	3.1.4	scheduler	yes	yes	yes	yes
	3.1.5	schedclient	yes	yes	yes	yes
	3.1.6	tis_server	yes	yes	yes	yes
	3.1.7	tiseg_server	yes	yes	yes	yes
	3.1.8	ticron_server	yes	yes	yes	yes
	3.1.9	tuxshell	yes	yes	yes	yes
	3.1.10	TupleProc (currently not used)	yes	yes	no	no
3.2	Process Monitoring and Control					
	3.2.1	Launch[4] (currently not used)	yes	yes	no	no
	3.2.2	Launchd[5] (currently not used)	yes	yes	no	no
	3.2.3	WorkFlow	yes	yes	yes	yes
	3.2.7	WF:ProcessInterval (csh)	yes	no	yes	yes
	3.2.9	WF:SendMessage	yes	no	yes	yes
	3.2.11	dman	yes	yes	yes	yes
	3.2.12	tuxpad (Perl/Tk)	yes	yes	yes	yes
	3.2.13	qinfo (Perl/Tk)	yes	yes	yes	yes
	3.2.14	schedule_it (Perl/Tk)	yes	yes	yes	yes
	3.2.15	operate_admin (Perl/Tk)	yes	yes	yes	yes
	3.2.16	msg_window (Perl/Tk)	yes	yes	yes	yes

Table 1: Summary of Y2K Assessment and Remediation, and Testing for CSCI 3[1]

CSCI/CSC		Y2K Assessed and Remediated	Unit Tested[2]	Integration Tested on PIDC Testbed	Y2K Tested on PIDC Testbed[3]
3.3	Libraries				
3.3.1	libipc	yes	included	included	included
3.3.2	libipc:birdie, xbirdie	yes	yes	yes	yes
3.3.3	libAdsn (64% lex & Yacc)	yes	included	included	included
3.3.4	libXadsn	yes	included	included	included

1. The Fixed Data Set test does not use the DACS system and so CSCI 3 is not part of this test.
2. Applications developed in San Diego were unit tested before release. Test forms are on file with the MSO Test Manager.
3. Client (i.e. non-server) applications such as tuxpad (and its related scripts), WorkFlow, Send-Message, schedclient, dman, and birdie may have not been explicitly tested across the Y2K boundary. However, all such applications were tested with successful results for dates before and after Y2K, and there is a high degree of confidence that all applications will work for any date, before, at, and after Y2K.
4. Launch was unit tested on the USNDC testbed in San Diego where it is configured to control the USNDC continuous data system. This application is not used by the PIDC/IDC.
5. Launchd was unit tested on the USNDC testbed in San Diego where it is configured to control the USNDC continuous data system. This application is not used by the PIDC/IDC.

Outstanding Problems for CSCI 3

Table 2 lists problems identified during testing which have not yet been addressed along with an indication of their severity:

- *High* this problem should be addressed before promotion to operations
- *Medium* this problem should be addressed before R2.1 is delivered to Vienna
- *Low* this problem can be addressed after R2.1 is delivered to Vienna.

Table 2: Outstanding Problems in CSCI 3[1]

CSCI/CSC	PROBLEM	SEVERITY
3 Distributed Processing		
3.1 Application Service		

1. No new problems were identified during integration testing for DACS.

Summary of Unit and Integration Testing

The primary testing methodology for DACS is examination of log files to assess proper DACS operation, known error conditions, and new error conditions. Cooperating database queries and Tuxedo queue examination are included with the log file examination when appropriate. Both unit and integration tests are organized via CSC test procedures. However, DACS is a highly integrated system and so the validation of each component is based upon the proper operation of at least several other DACS components (e.g., *tuxshell* will fail even the most basic test procedure if *dbserver* and/or the Tuxedo queuing system is not available and running correctly).

The PIDC 6.2 DACS integration test consisted of three tests or test periods including the live data test, Y2K millennium boundary test, and Y2K leap boundary test. The same test procedures were carried out for each of the three test periods. Basically, the integration test procedures validate operation of DACS and so the Y2K tests assessed whether or not PIDC 6.2 DACS will successfully run before, at, and after Y2K. The test procedures were upgraded with a focus on date and timestamp diagnostics in the log files. These changes matched and were motivated by the specific Y2K remediations included in PIDC 6.2 DACS.

Table 3 summarizes the unit and integration test procedures and results. In the table, the Y2K test type is denoted by **M** for tests that analyzed system function during the millennium boundary test period (i.e. before, at, and after Y2K), **Y** for tests that were run during the Y2K test period, but after the millennium boundary, and **L** for tests that analyzed system function across the Y2K leap test period.

Table 3: Summary of Unit and Integration Testing

Test Procedure	In Plan (Y/N)	Brief Description	Result[1]	Y2K Test Type[2]
UNIT TEST				
Data Monitor Interval creation	Y	Verified that *tis*, *tiseg*, and *ticron* do basic interval creation by examining and confirming evidence of expected intervals in log files, tuxedo queues, and the database.	Passed	**Y**[3]
scheduler works and does not crash	Y	Verified that *scheduler* schedules data monitor services and does not spontaneously crash by examining log files and running scheduler for 10 continuous crash free days.	Passed	**Y**
tuxshell error handling for compound processing	Y	Verified proper error handling/reprocessing of compound (i.e. GA) pipelines that fail at any pipeline step by examining log files and Tuxedo queuespace.	Passed	**Y**
Priority based intervals.	N	Verified that priority based interval creation worked by examining tuxedo queue elements using *qmadmin* and *birdie*	Passed	**Y**

Table 3: Summary of Unit and Integration Testing

Test Procedure	In Plan (Y/N)	Brief Description	Result[1]	Y2K Test Type[2]
SendMessage	N	Verified valid enqueues of interval messages to the Tuxedo queues using *qmadmin* and *birdie*.	Passed	**Y**
DACS	Y	Verified DACS interval updates, interval routing, and interval recycling by examining the *dbserver*, *interval_router*, and *recycler_server* log files and database.	Passed	**Y**
WorkFlow / ProcessInterval / SendMessage	Y	Verified that *WorkFlow* GUI accurately presents the state of the interval table using database queries. *SendMessage* based Interval reprocessing was verified via database queries and *WorkFlow* GUI updates.	Passed	**Y**
tuxpad / tuxpad scripts	N	Verified proper function of *tuxpad* by successfully using the GUI tool to control DACS during unit testing.	Passed	**Y**
INTEGRATION TEST				
dbserver test	Y	Verified multiple *dbservers* correctly update intervals in the database by examining the testbed log files and database.	Passed	**M**,**Y**,**L**
interval_router	Y	Verified multiple *interval_routers* route interval messages to the intended Tuxedo queues by examining testbed log files.	Passed	**M**,**Y**,**L**
recycler_server	Y	Verified that multiple *recycler_servers* will recycle interval messages by examining testbed log files. Noted misleading log messages. Noted that controlling *TMQFORWARD* will drop messages if the Tuxedo queue space fills up.	CMR-452	**M**,**Y**,**L**
scheduler / *schedclient*	Y	Verified that the primary *scheduler* server consistently schedules all configured data monitors by examining testbed log files. Never observed *scheduler* crash upon issue of the 'INIT' command from *schedclient*. A workaround exists for this problem.	CMR-461	**M**,**Y**,**L**
ticron_server	Y	Verified that *ticron* creates intervals at the configured frequency by examining testbed log files. Verified that *ticron* correctly rolls back interval creation during a Tuxedo queueing system failure by examining the testbed log files and database.	Passed	**M**,**Y**,**L**

Table 3: Summary of Unit and Integration Testing

Test Procedure	In Plan (Y/N)	Brief Description	Result[1]	Y2K Test Type[2]
tiseg_server	Y	Verified that *tiseg* creates FULL and PARTIAL intervals, updates PARTIALS to FULL, and suppresses creation of duplicate intervals by examining the testbed log files and database. Noted that *tiseg* wfdisc query can be refined to improve duplicate interval filtering. Noted that additional testing is required to verify valid DB rollback and recovery following queueing failures.	CMR-457 CMR-458	**Y**,**L**[4]
tis_server	Y	Verified via log file examination and database queries that *tis* correctly performs new interval creation and skipped interval creation including subsequent promotion of skipped intervals to current intervals. Verified proper enqueuing operation for each interval formed.	CMR-362 CMR-376 CMR-447 CMR-448	**M**,**Y**,**L**
WaveGet_server	N	*WaveGet* has proven to work during the course of the integration test period. There are no known problems.	Passed	**M**,**Y**,**L**
tuxshell	Y	Verified that *tuxshell* correctly manages child execution, child time-out, child success, and child failure for single and compound pipelines by examining testbed log files and database. Noted case where tuxedo queue and interval state desynchronize.	CMR-449	**M**,**Y**,**L**
WorkFlow / ProcessInterval / SendMessage	Y	Verified that *WorkFlow* GUI accurately presents the state of the interval table using database queries. *SendMessage* based Interval reprocessing was verified via database queries and *WorkFlow* GUI updates.	Passed	**Y**,**L**
tuxpad / tuxpad scripts	Y	Verified proper function of *tuxpad* by successfully using the GUI tool to control DACS during integration testing.	Passed	**Y**,**L**

1. Result is **passed** or a reference to a pre-existing SMR. The resolution of the SMRs is not a condition for promotion of PIDC 6.2 DACS in operations.
2. **M**=Millennium boundary test, **Y**=Y2K test, **L**=Y2K leap boundary test.
3. Unit testing included millennium boundary testing that was successful. However, the test results were not included in the unit test forms.
4. Auxiliary station interval creation was disabled during the millennium boundary test period due to operational limitation outside the scope of DACS.