LEVEL II

12

RADC-TR-81-124
Final Technical Report
June 1981

NSW EXECUTIVE ENHANCEMENTS

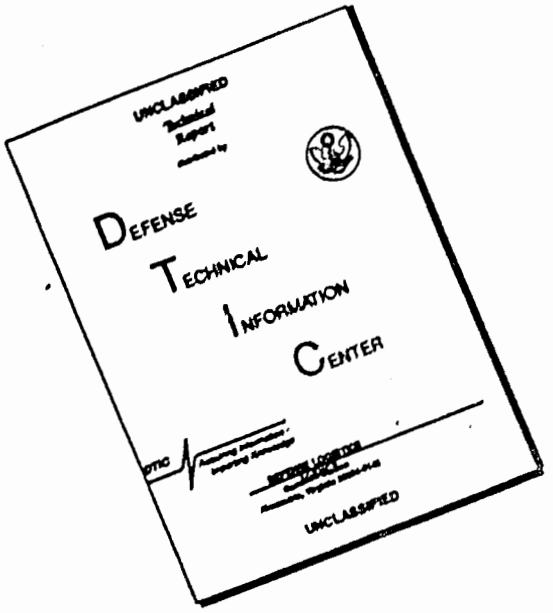
Massachusetts Computer Associates, Inc.

Sponsored by
Defense Advanced Research Projects Agency (DoD)
ARPA Order No. 3686

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U. S. Government.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-124 has been reviewed and is approved for publication.

APPROVED: Patricia J. Baskinger

PATRICIAL J. BASKINGER
Project Engineer

APPROVED: Alan R. Barnum

ALAN R. BARNUM
Acting Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCP) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

NSW EXECUTIVE ENHANCEMENTS

Charles A. Muntz

Contractor: Massachusetts Computer Associates
Contract Number: F30602-79-C-0100
Effective Date of Contract: 12 December 1978
Contract Expiration Date: 12 December 1980
Short Title of Work: NSW Executive Enhancements
Program Code Number: 8P10
Period of Work Covered: Dec 78 - Dec 80

Principle Investigator: Charles A. Muntz
Phone: 617 245-9540

Project Engineer: Patricia J. Baskinger
Phone: 315 330-7007

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Patricia J. Baskinger (ISCP), Griffiss AFB NY 13441 under Contract F30602-79-C-0100.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: RADC-TR-81-124	2. GOVT ACCESSION NO.: AD-A102805
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): NSW EXECUTIVE ENHANCEMENTS	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report, Dec 78 - Dec 80
6. PERFORMING ORG. REPORT NUMBER: CADD-8101-1201	
7. AUTHOR(s): Charles A. Muntz	8. CONTRACT OR GRANT NUMBER(s): F30602-79-C-0100
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Massachusetts Computer Associates, Inc. 26 Princess Street Wakefield MA 01880	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63728F 25310101
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington VA 22209	12. REPORT DATE: June 1981
	13. NUMBER OF PAGES: 128
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Rome Air Development Center (ISCP) Griffiss AFB NY 13441	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Patricia Baskinger (ISCP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed Operating Systems
Computer Networks
Software Systems
Network Operating Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The National Software Works (NSW) is a distributed software system resident on the ARPANET. It is intended to support the development, use, maintenance, and storage of programs and data on which the programs operate. It is principally aimed at the development of software systems and at providing software tools which can be used to support the software development activity throughout its life cycle.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

013745

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The National Software Works (NSW) is a significant development in the fields of distributed processing and network operating systems. Its initial ambitious goal was to link the resources of a set of geographically distributed and heterogeneous hosts with an operating system which would appear as a single entity to a user.

The National Software Works has been developed in response to a growing concern over the high cost of software. The Air Force has estimated, for example, that by 1985 software expenditures will be over 90% of total computer system costs. In the attack on the cost and complexity of developing and maintaining software, both industry and government have made enormous investments in software tools -- automated aids for the implementors of software and for the managers of software projects. These tools include compilers, editors, debuggers, design systems, test management tools, etc.

As a result of this investment, a large inventory of excellent tools has come into being, so that the difficulty confronting the management of a programming project lies not in the existence of suitable tools, but in their availability. If some essential tool does not happen to have a version which runs on a computer to which the project has access, the manager is forced to choose among the expensive alternatives of (1) foregoing use of the tool, (2) undertaking to acquire or produce a surrogate tool on his hardware, or (3) purchasing (access to) a computing system on which the tool does run.

Alternative (1) -- not using the tool -- has become prohibitively expensive as the size and complexity of programming projects has grown: most projects can hardly choose not to use a compiler, and this, for instance, may lead to choice of a sub-optimal programming language merely because a compiler for it is available.

Alternative (2) amounts to the software portability problem. The computing field has been chipping away at this intractable problem for years, by encouraging the use of standard -- or at least popular -- programming languages, having compilers on many different computers. Lacking this facility, the project must invest effort at the beginning to re-program -- and debug -- the tools it needs before starting on its actual task.

Alternative (3) has been attacked most effectively with networking. Indeed, it was the marked success of the Arpanet in providing programmers economical access to diverse computers which

provided the foundation on which the concept of a "National Software Works" was built. Instead of moving the software from host to host, let the programmer (and manager) use each software tool on whatever host it already occupies.

We can say that NSW tries, in effect, to make Alternative (3) more attractive, by providing the Arpanet without some of its drawbacks. Using the Arpanet in straightforward fashion, by using Telnet and FTP to access hosts other than one's "home" system, does indeed give you a much wider domain of action, but nonetheless:

- You need an account on each host. This involves the allocation of funding, drawing up contracts, etc.
- The operating system on each host is different, so you must learn different login procedures, command languages, interrupt characters, file naming conventions, etc. Further, you must not confuse each system's conventions as you move from tool to tool.
- Files output from one tool (say QEDX on MULTICS) are to be input to another tool (say CMS2M on IBM 360). This involves at least network transmission and usually file reformatting. To appreciate the magnitude of this problem one should try to use FTP (Arpanet File Transfer Protocol) to move a QEDX output file -- a sequential file of 9-bit ASCII characters in 36-bit words -- to an IBM 360 to be a CMS2M input file -- a blocked file of 80 EBCDIC character records in 32-bit words.

These and similar problems will be familiar to anyone who has used several different systems.

The purpose of NSW is to make this solution (of providing programmers access to tools on different hosts) a practical reality. The NSW user should not have to know about OS/360, TENEX, and MULTICS with their differing file systems, login procedures, system commands, etc.; knowledge of how to use the individual tools which are needed for the job should suffice. He should not have to worry about reformatting and moving files from a 360 to a TENEX; file transmission should be completely transparent. The user should not have to worry about obtaining accounts on many different machines, but instead should have a single NSW account.

Thus, the overall design goals of the National Software Works were to provide programmers with a

- Unified tool kit -- distributed over many hosts -- and a
- Single monitor with
  - uniform command language,
  - global file system,
  - single access control, accounting, and auditing mechanism.

CHAPTER 2: HISTORY OF NSW PROJECT

2.1 NSW Goals

2.1.1 Original Directions

As originally conceived, NSW was to provide a unified cross-net operating system of the kind described in the Introduction, and with the following specific external goals in mind:

(1) The first such goal was large scale.

Contemporary operating systems support tens of concurrent users; NSW was to support many more users, possibly as many as one thousand. The catalogue alone of the file system for that many users could easily fill a large disk pack. And the table space required for keeping track of a thousand users and the software tools they are using could easily exceed the virtual memory of TENEX.

(2) The second goal was high reliability.

If there are one thousand online users, then a two-hour system failure costs one man-year of work. The National Software Works -- particularly its monitor and file system -- must degrade gracefully. Failure of a single component -- e.g., a TENEX system on which tools are running -- must only reduce system capacity, not destroy it. Further, only those users actually using a failed component should be affected by its failure.

(3) The third goal was support of project management.

NSW was to provide to managers of software projects a collection of programs, called management tools, which they could use to monitor and control project activities. The underlying assumption here is that a manager's ability to insure that each programmer's efforts contribute most effectively to overall project goals can be greatly enhanced by automating routine management tasks. Furthermore, it is assumed that a good environment for this automation is the system which supports the project programming activities, because it represents an effective point for monitoring and controlling those activities.

(4) The fourth goal of NSW was practicality.

NSW was not to be a "blue sky" system, whose implementation required unrealistic assumptions about its environment. In particular, "practicality" meant:

- Minimum modifications to existing operating systems on Arpanet hosts. "Minimum" was, in fact, to be construed as "none". It was permissible, if necessary, to add privileged (i.e., non-user) code to existing systems, but the solution to the problem should not depend on rewriting the kernel of any existing operating system.

- Minimum modifications to existing tools. Here, "minimum" no longer meant "none". It was permissible to require some change to a tool as part of the process of installing it in NSW, but such changes should be small-scale and straightforward.

- Maximum generality. Any solution which permits the easy installation of existing tools must also allow the easy construction and installation of new tools.

- No experimental hardware. This requirement meant that new hardware-oriented approaches to reliability -- e.g., PLURIBUS -- could not be used. The NSW monitor and file system are to run on already available Arpanet hosts.

Although it was never stated as explicitly as the goals enumerated above, it seemed to follow from the whole concept of a National Software Works that it would be the sole on-line working environment for most of its users. As we went about the initial design, our mental picture of the user's activity was that he would log into NSW and stay in that environment for all of his machine activity -- cheerfully using any tool he had access to, without even knowing, or caring, on what host in the network it was running, or where his files were stored.

2.1.2 Changed Directions

As the initial design was completed and the prototype NSW was implemented and used, the original goals had to be somewhat modified.

The original goal of supporting as many as one thousand users with a single NSW proved not to be economically feasible in the present

state of the art: it would have required excessive hardware resources, and might still not have shown acceptable responsiveness. Hence, the current plan is to produce a self-contained, moderate-sized system -- supporting up to a hundred users -- which can be distributed in toto; that is, a number of such systems could be in operation independently on the same network.

Given this reduction in scale, the projected implementation of a widely distributed, fully replicated, synchronized data base -- the design of which was well along -- was also seen as an inappppropriately large investment. If each NSW system then will have a single monitor host, the principal reliability issue becomes one of preventing a monitor-host crash from aborting ongoing tool operations. Thus, rather than casting the monitor as a set of cooperating distributed processes, as was designed in the large-scale reliability plan, we propose to allow tool execution to continue despite the absence of the monitor, given that the user's rights have initially been verified by the monitor, and by allowing the results of tool executions to be held indefinitely until the monitor is again available to accept delivery.

Our mental picture of NSW utilization has also changed. At current hardware and network speeds, there is a human-time overhead to be paid for using NSW, and anyone with experience on a present-day timesharing system will be conscious of it. Use of a single tool is not too seriously degraded, compared to normal TIP-to-host operation, but inter-tool communication by passing files through the central system seems to take a long time when compared with such operations within a single-host operating system. If the user does indeed have to use multiple tools at separate hosts, then using NSW is a pleasure in comparison with going through the scenarios necessary to perform the same operations using Telnet and FTP; but there doesn't appear to be a large market, as yet, for this ability.

We believe, nonetheless, that there are at present valid uses for such a "Network Operating System" facility, and that there will be an even bigger place for it in the future, considering the forecasts of enormous interconnected networks of personal computers, mass stores, high-performance mainframes, and special-purpose devices; our future plans for NSW therefore include continued improvements in the facility for this mode of operation.

However, our current picture of the "heavy" user of NSW is that he will work, as he does now, primarily within his "home" host system (though he might establish contact with it via NSW mechanisms), and he will use NSW facilities, when he needs to, on an escape basis -- he

will signal NSW that he has produced files to be placed in the NSW File System for others to access, or that he needs to execute some tool on a "foreign" host. Later sections on the work of the Analysis Group will discuss the embodiment of this changed perspective.

The design goals as stated in the Introduction, then, might be updated to say that the intent is to provide programmers -- and project managers -- with a wider and more automated environment, including:

- Easy access to services not normally available on their home systems;
- Controlled access to a cross-net file system of text files, programs, and services;
- A structure and some tools for project coordination and configuration management.

2.2 NSW Architecture

In this section, we summarize the overall structure of the NSW, as it evolved to meet the original design goals of a dispersed toolkit accessed through a central monitor.

2.2.1 NSW Components

The requirement is that many users on many different hosts be able to access many tools on many different hosts, under control of a central monitor, and with reference to a global file system. Analyzing this system according to the functions that must be available on each host, we were led to specify a number of functional processes, whose embodiments are called "NSW Components".

By its very definition, NSW is a distributed system. Tool processes run on different Arpanet hosts, and the monitor process must run on at least one Arpanet host; hence, there must be some form of inter-host process-to-process communication. There are low level Arpanet protocols for moving bits from host to host, and there are also several higher level protocols for moving files and for terminal communication. None of these protocols, however, is oriented toward the kind of inter-process communication which NSW requires. Moreover, even though NSW is being implemented on the Arpanet, we want to keep it as independent as possible of the underlying milieu. Network

technology is evolving, and we wish to be able to realize the NSW architecture on tomorrow's networks as well. Hence, the first technical problem to be solved is the definition and implementation of an appropriate inter-host inter-process communication protocol. The protocol developed for NSW is called MSG, and the component on each host which implements the protocol is also called MSG.

When the NSW user runs an interactive tool within NSW, either NSW must arrange that his terminal appear to the tool as if it were a simple user terminal on the native host -- this is most appropriate when adapting existing tool programs to run within NSW -- or the tool must be able to perform its input/output transactions via the MSG protocol. As well as talking to tools, the user will be giving commands directly to the NSW monitor, and it is not feasible for all NSW users to have direct terminal access to the monitor host. Therefore, the user will be represented within the NSW system by a process (on any host, in principle) which will communicate for him with the monitor via MSG, and which will handle his connections to tools. This component is called the Front End.

A tool running on some machine makes system calls requesting resources -- primarily file access. Since access to NSW system resources is to be controlled, accounted for, and audited by the NSW monitor, such requests must be diverted from the local system and referred instead to the NSW monitor. In addition, if the tool is interactive, it expects to have a terminal for communication with the user, and this in NSW is via the Front End. So, without modifying the operating system, we must divert the tool's communications with the user and the tool's requests for NSW resources, while still allowing the tool to obtain local-service access of other kinds. The NSW component which solves this problem is called the Foreman.

Batch tools are best described as those whose input and output must be completely specified before tool execution begins. Such tools should not (and often cannot) be supervised from a terminal. Instead the central monitor works together with a component called the Batch Job Package, running on the same host as the batch tool, to supervise execution of such "absentee" computations.

It was understood from the beginning that there would be no physically separate NSW file-storage media -- NSW files physically reside within the filing systems of the participating hosts, ideally on (or "near") the hosts on which they are most likely to be needed. Hence, there is need for a process on each type of host which understands that host's file system, and can manage the portion of the

host system which is available for NSW file storage. Also, we expect that the output of one tool will be used as input to another tool. Unfortunately, if the first tool is a MULTICS editor and the second an IBM 360 compiler, this operation involves character translation (ASCII to EBCDIC), file reformatting (sequential file to blocked record file), and file movement (across the Arpanet). To handle these functions of file storage, transformation, and movement, there is an NSW component called the File Package.

It is worth noting at this point that all of the above components are distributed. Every host in NSW has an MSG server process. Every site to which a user is connected has a Front End. Every tool-bearing host has a Foreman. Every host on which NSW files are stored has a File Package. It is also worth noting that implementation details of these components vary from host to host. A MULTICS Foreman will be vastly different from an IBM 360 Foreman. Functional specifications for these components are fixed throughout NSW, but implementation and optimization decisions are left free.

And finally, there must be a component for the NSW monitor, or at least for those normal monitor functions which are not already performed by the Foreman (service calls for file access) or the Front End (terminal handling, command interpretation). This component is named the "Works Manager"; we shall discuss it in more detail in the next subsection.

Let us then summarize the major functional areas of the NSW design and the components which embody those functions:

o Inter-host inter-process communication	MSG
o User interface	Front End
o Diversion of communication with local operating system	Foreman
o Supervision of batch jobs	Batch Job Package
o File storage, transformation, and movement	File Package
o Monitor functions	Works Manager

2.2.2 NSW Monitor and File System

The design of the NSW Monitor -- called the Works Manager -- was probably more affected than any other component by the goals of NSW. Functionally it is not different from any other conventional access-checking, resource-granting monitor (though the mechanisms for defining access domains are unusual). Structurally, however, it is significantly different.

The goals of providing both large scale and reliability on conventional hardware led to a design for distributing the Works Manager and file system. If there are many instances of the NSW monitor on many different hosts, then failure of a host is not catastrophic. Unfortunately, distribution runs counter to the problem-required logical unity of the monitor and file system. If a user inserts a file into the file system using one tool and one instance of the file system, and then requests the same file using a different tool and a different instance of the file system, the two instances of the file system must share a common file catalogue for the system to behave properly. Similarly, all instances of the monitor must share an access-rights data base for proper validation of user requests to run tools.

As mentioned in section 2.1.2, this ambitious design has been shelved. A "large" NSW system -- with multiple Works Managers -- is still feasible, but with a partitioned (rather than a replicated) data base: each Works Manager will control the resources in that piece of the partitioned data base which it "owns", but will have to negotiate with another Works Manager that "owns" other resources. This strategy requires minimum synchronization while providing advantages in reliability and robustness.

2.3 Phases of NSW Development

The design and implementation of the National Software Works has proceeded in five slightly overlapping phases:

- Structural design and feasibility demonstration
- Detailed component design
- Prototype implementation
- Reliability and performance improvement

o Management Plan and Production System

In the following subsections we describe these phases in more detail.

2.3.1 Structural Design and Feasibility Demonstration

The first phase of NSW development began in July 1974 and concluded in November 1975. During this period, the basic architecture of NSW (described in Section 2.2) was established. Further, relatively ad hoc implementations of major components were made. These components were integrated into a system which was demonstrated to ARPA and Air Force personnel at Gunter AFB in November 1975. This demonstration exhibited various system functions, the use of batch tools on the IBM 360 and Burroughs B4700, the use of interactive tools on TENEX, transparent file motion and translation, and a primitive set of project management functions.

This demonstration confirmed that the expected NSW facilities could be implemented and that transparent use of a distributed tool kit was feasible. The NSW System, however, was inefficient and fragile. Further, many of the ad hoc implementations had design weaknesses which limited their general application to a sufficiently broad range of hosts and capabilities. For these reasons, an effort was begun to produce adequate component designs.

2.3.2 Detailed Component Design

This second phase of NSW development was begun in June 1975 with the initial MSG design document. Specifications were developed for Tool-Bearing Host components -- MSG, Foreman, and File Package. All of these specification documents were completed by March 1976. (They have all been revised since then, but the original specifications are still substantially correct.)

During the same period, the external specification of the Works Manager was also produced. Again, although this specification has subsequently been revised, it is still substantially correct. The remaining portions of the core of NSW -- i.e., the batch tool facility, consisting of the Works Manager Operator, Interactive Batch Specifier, and Interface Protocol -- were designed during phase one, and those designs were retained until phase four (see below).

The remaining major NSW component, the Front End, was the subject of several design efforts. Three incomplete specification documents were produced but none of these was wholly satisfactory. Nevertheless, sufficient design to allow implementation of a functionally correct Front End was accomplished.

2.3.3 Prototype Implementation

As specification documents were completed, various contractors began implementation of the NSW components on the initial set of hosts -- TENEX, MULTICS, and IBM 360; these efforts commenced in January 1976. Implementation on TENEX proceeded more quickly than the efforts on the other hosts -- primarily because the MSG system designers were also TENEX implementors. By October 1976 prototype implementations which conformed to the published specifications had been made for all TENEX TBH components. In addition, all components of the core system were available on TENEX.

Implementation of TBH components on MULTICS and IBM 360 proceeded more slowly; however, initial implementations of MSG components on both of these hosts were completed by the end of 1976. By November 1976 sufficient progress had been made on implementation of a File Package and Foreman on MULTICS that it was possible to demonstrate an interactive tool running on MULTICS. Progress on implementation of 360 (interactive) TBH components reached a similar position in September 1977.

Also during this phase, a TENEX Front End which functionally supported the Works Manager and Foreman according to the appropriate specifications was implemented.

An NSW system containing prototype implementations according to the specifications of the core system, TENEX TBH components, TENEX Front End, batch IBM 360 tools, as well as a rudimentary MULTICS interactive tool was demonstrated to Air Force and ARPA personnel in November 1976. At the same time, a demonstration of MSG components on all three hosts was also given.

2.3.4 Reliability and Performance Improvement

Even though implementation of components on MULTICS and IBM 360 was lagging, implementation of the core system, TENEX TBH components, and TENEX Front End had proceeded to the point that the issues of

reliability and performance assumed major importance. The system exhibited sufficient functional capability that it could clearly support use by programmers if it were sufficiently robust and responsive.

The first task attacked was to provide robustness. Work had begun in 1975 on a full-scale NSW Reliability Plan -- this was the design for multiple Works Managers with duplicate data bases. This detailed Plan was released in January 1977. Since it was clear that implementation of the full Plan was a major undertaking, a less ambitious Interim Reliability Plan which ensured against loss of a user's files was begun in mid-1976. This Plan was also released in January 1977. By June 1977 the core system, TENEX Foreman, and TENEX Front End had been modified to incorporate the features of that Interim Plan. In addition, both the MULTICS and IBM 360 Foremen (only partially implemented) were altered to conform externally to the scenarios specified by the Interim Reliability Plan. A system exhibiting the new scenarios was released for use in June 1977.

Performance of NSW had been slow from the initial implementation. The reasons for slow response were many:

- Interaction between components was by a thin wire (MSG and the Arpanet).
- NSW components (which constitute an operating system) nevertheless were executed as user processes under the local host operating system.
- Component implementation had been oriented towards ease of debugging and other concerns of prototype systems rather than towards the performance expected of a production system.

In 1977, efforts to improve NSW performance were begun.

The first effort was the development of a performance measuring package for TENEX MSG. Results of the first set of measurements were reported in April 1977; and a number of more sophisticated measuring packages were complete by February 1978. By May 1978, all TENEX components had been instrumented and measurements of page use, CPU time, elapsed time, use of JSYSes (TENEX monitor calls), etc., had been taken under a variety of system load conditions and on several different TENEX hosts. Efforts were then undertaken to make the performance improvements suggested by these measurements.

Performance improvement is an ongoing concern. Efforts to improve the program efficiency of TENEX components are substantially complete, but there are now plans for re-allocating some of the system functions, with the aim of making whole protocols more efficient.

2.3.5 Production System

Concurrent with the effort to improve NSW reliability and performance, an effort to make NSW a more packaged product were begun. Regression tests for the externally available NSW user system were developed and applied to each system release. A user's manual for the system was published. Documentation of the core system was produced. Finally, a draft configuration management plan was developed.

Work was begun in late 1978 to establish NSW as a software product. The NSW Management Plan was generated. This document identified a number of roles associated with development, operation, and support of NSW as a product. Briefly these are as follows:

Role	Responsibilities	Organization
(PG) Policy Group	Requirements and Policy	RADC/ARPA
(PDC) Product Development	Product Definition	GSG
(OPS) NSW Operations	Operations; User Support	GSG
(ACC) Architecture Control	Product Integration	COMPASS
(DMC) Development and Maintenance	Component Development and Maintenance	BBN, COMPASS, HIS, UCLA
(TM) Tool Manager	Tool Management	IITRI

A product baseline is being established to bring NSW under Configuration Management. A reasonably complete set of requirements/specification documents has been installed as a set of files in the NSW User System. NSW's Information Retrieval System allows queries on sets of documents; the naming scheme for the baseline demonstrates some of the power of NSW's file naming capabilities:

Document Type	Name Syntax

Requirements	NSW.REQUIREMENTS...
A-level Specifications	NSW.A-SPEC...
B-level Specifications for <component>	NSW.B-SPEC.<component>...
C-level Specifications for <component> on <operating-system>	NSW.C-SPEC.<operating-system>.<component>...

where the variables take on values as follows:

<component>	<operating-system>
BJP	TENEX
FE	OS360
FL	MULTICS
FM	UNIX
FP	...
...	

The revision level of all of these documents is also noted, as part of the file name.

A computerized tool for coordinating the reporting, checking, fixing, and testing of software problems has been developed and put into use: it is called MONSTR -- a MONitor for Software Trouble Reporting. It is driven by tabled protocols defining the desired interactions between all the organizations listed above, and handles the passage of messages through the appropriate channels between them. MONSTR's current status is described in Chapter 10.

CHAPTER 3: OVERVIEW OF CURRENT STATUS, RECENT WORK

3.1 The Present System

The NSW system currently available to users is NSW 5.0, released in August 1980. It offers the following general features:

- Twenty interactive TOPS20 tools, running under a new Foreman with extended workspace-handling features.
  (The last TENEX system used by NSW was converted to a TOPS20 during the present contract period.)

- Ten interactive MULTICS tools, running with extensively rewritten and much improved Tool-Bearing Host components.

- Two interactive IBM 360 tools, and eight IBM 360 batch tools. (During the contract period, the IBM 360/91 was removed from the system, for replacement by an IBM 3033; see section 6.6 for current status.)

- Core components (Works Manager and Works Manager Operator) and a TOPS20 Front End, interpreting and implementing a basic set of system commands.

- Better-organized user documentation, and better-coordinated user support, as a result of using the MONSTR coordination tool (see Chapter 10).

- Rudimentary management (node manipulation) tools, the rights to use which can now be assigned and removed with the standard tool-rights mechanism.

- The normal configuration of NSW 5.0 includes the following hosts:

ISIE	(TOPS20 -- Works Manager, Tools)
ISIC	(TOPS20 -- Tool-Bearing Host)
RADC-TOPS20	(Tool-Bearing Host)
CCN-360/91	(Batch and Interactive tools)
RADC-MULTICS	(Tool-Bearing Host)

- During the period of the present contract, significant operational improvements have been made, particularly:

- The release procedure has been significantly formalized, partially automated, and brought more under control:
  - . A source code repository is maintained for the life of a release.
  - . A semi-automatic configuration control facility has been implemented, to handle site-specific parameters.
  - . A release-specific document is produced for each new release, detailing the configuration and installation procedures, operational procedures, and changes from the previous release.
- The MONSTR tool for cataloguing and tracking Software Trouble reports, and their associated fixes, has been put into use.
- A new central component, the Fault Logger, has been added, and the Operator Utility component has been modified to provide access to the Fault Logs.
- A UNIX Front End has been implemented and used successfully with the current system, though it is not officially a part of Release 5.0.

Functionally, the current NSW system is minimally adequate. It has a reasonable collection of tools, but many of these tools have not been adequately tested. The minimal set of user commands is available and tested, but many needed user features are lacking -- e.g. command macros, '*' in file commands, I/O devices, Arpanet mail, etc. Performance has been improved significantly. The documentation of system components has been improved, but much needs to be done.

TOPS20 is available as Works Manager or Tool-Bearing Host according to specification, and TOPS20 tool encapsulation has been extended to include the ability to REBEGIN a suspended tool, and to keep the user better informed about files in the tool's workspace. Additional encapsulated tools can be installed at will to increase NSW capacity.

Eight batch tools were available on the CCN IBM 360/91, and will again be available on the 3033 when the NSW components have been adapted to run with the new operating system; more can be installed as needed. A major overhaul of the entire batch system has made it more

consistent with the rest of NSW, more flexible, powerful, operable and resilient. The IBM 360 interactive Foreman handled only two interactive tools at the time of the switchover, and its development will continue when the new machine is operable within NSW.

The MULTICS implementation of all NSW Tool-Bearing Host components has been greatly improved during this contract period, and further progress is expected.

The current status of the individual component implementations is presented in the Chapters 4 through 8, by host system for the Tool-Bearing Host components, and by functional class for the Core system components and the Front Ends.

3.2 The Redesign Effort

During the present contract period, a group of senior NSW personnel, under the chairmanship of Dr. Robert Thomas of Bolt Beranek and Newman, was formed to reconsider the entire NSW design, and to draft a re-design of the entire system. This group was named the "NSW Analysis Group", and its charter was to rethink the organization and functional distribution of the NSW -- not to throw away all the previous design and plan an entirely new Network Operating System, but to reconsider all the design decisions which had been made since 1974, in the light of experience and the "changed directions" concept of the potential uses of NSW.

The group met frequently during 1980, and exchanged working papers. The final report of this effort is entitled "NSW Functional Specification", Revised Draft September 1980. The conclusions in this document have largely determined the direction of planned changes in NSW.

Following the lead of this effort, a task-list was formulated for the changes and improvements to be made to NSW for the next two releases, choosing from the features of the "Functional Specification" those which could be applied to the present Release 5.0, and those which would be of greatest value for the ongoing Air Force Technology Demonstration. Estimates of the cost and difficulty of the items on this task-list have been made, and some of the new features are already past the preliminary design stage. This task-list, with amplifications, constituted the statement of work for Mass. Computer Associates proposal for continuing effort on NSW. The task-list appears as an Appendix to this report.

CHAPTER 4: CORE SYSTEM COMPONENTS

Tasks relating to the Core System during this contract period naturally divide into two separate activities:

- o baselining existing software and placing it under configuration management
- o analyzing the original concepts and producing a design for a radically improved system.

In this section, we discuss the first of these; the second activity comes under the Redesign Activity mentioned in the previous chapter. The subsections of this chapter give additional details on the present status of individual core components and list the planned extensions and improvements.

The basis for configuration management of the Core System components is a set of TOPS-20 command procedure files which are used to produce executable files (*.EXE) representing a numbered generation of code. In addition, these procedure files automatically generate *.SET files which serve as a configuration index for the *.EXE file. The Works Manager, for example, is composed of approximately twenty BCPL source files. An example is shown below.

```
   PS:<WMSRC>
 WM.EXE.295 120320(36)  11-Mar-80 05:50:16

     includes rel files derived from

   PS:<WMSRC>
 WMDRC.BCP.6     14904(7)    7-Jan-80 10:33:47
 WMFCAT.BCP.36   31664(7)    7-Jan-80 10:38:12
 WMIBS.BCP.122   15298(7)    7-Jan-80 10:46:42
 WMIETT.BCP.6    14891(7)    7-Jan-80 10:51:26
 WMINTJ.BCP.3    3814(7)     7-Jan-80 10:55:30
 WMISEM.BCP.14   20095(7)    7-Jan-80 10:59:03
 WMJSHO.BCP.41   2007(7)     7-Jan-80 11:07:24
 WMLNDS.BCP.25   15679(7)    7-Mar-80 11:18:54
 WMLOGS.BCP.155  24973(7)    7-Jan-80 10:10:28
 WMMAIN.BCP.108  14091(7)   14-Jan-80 11:42:33
 WMNODE.BCP.243  31813(7)    7-Jan-80 11:15:57
 WMOD.BCP.8      15614(7)    7-Jan-80 11:31:28
 WMRGHT.BCP.220  34772(7)    7-Jan-80 11:24:25
```

```
 WMRUEN.BCP.322 22418(7)   25-Feb-80 15:20:32
 WMSCOP.BCP.77  19857(7)    7-Jan-80 11:39:53
 WMSEMU.BCP.85  11009(7)    7-Jan-80 11:44:20
 WMSKUT.BCP.18  12750(7)   11-Oct-79 11:34:30
 WMUTIL.BCP.211 16643(7)    7-Jan-80 11:47:49
 WMWMO.BCP.138  9551(7)     7-Jan-80 11:51:37
 WMXUTS.BCP.68  10228(7)   11-Apr-79 11:25:31

 Total of 20 files
    Associated header/call files are

   PS:<WMSRC>
 WMRHDR.BCP.23  1257(7)    13-Mar-79 10:55:17
 WMRHDR.HDR.211 24576(36)  14-Jan-80 11:31:23
 WMRCALL.BCP.94 9080(7)     7-Jan-80 10:17:09

 Total of 3 files
-----------------------------------------------
```

This file is named WM.SET.295 and was produced by the TOPS-20 Directory command. The lines of text in the file identify the specific BCPL files including revision number and date modified.

A typical update scenario for the Works Manager might be the following:

1. ACC notifies DMC-COMPASS that Software Trouble Report (STR) 497 should be investigated and repaired as appropriate.

2. DMC-COMPASS reads the description of the problem and determines that module WMRUEN should be corrected.

3. WMRUEN.BCP.322 is edited to produce WMRUEN.BCP.323 and re-compiled.

4. The command procedure file MAKE-WM.DO is executed to produce WM.EXE.296 and WM.SET.296 which shows that WM.EXE.296 included the compilation of WMRUEN.BCP.323.

5. Upon successful testing, ACC is informed that WM.EXE.297 fixes STR-497.

The optimization and coordination of this class of activity is a major result of this contract period.

Operational support tools are enhanced, as well. A significant feature is the introduction of a system-wide parameter file named CONFIG.BAS. The contents of this file are lines of text; identical copies are placed in all component-execution directories in an NSW configuration. NSW components read operational parameter values from this file on initialization. These include directory names, time-outs, event logging specifiers, etc. This file contains all operational data which operations personnel may wish to control. Currently, all TOPS-20 sites consult this file; work is underway at other sites to support this feature.

Another support tool which is now improved is SIMWMT, a utility for manipulating sets of items in the Works Manager data base. A comprehensive manual defines and illustrates each of the 15 commands.

Some functionality improvements are included, but of modest scope. Tools can now be rerun in a saved workspace; previously, the only option available after an aborted tool session was selective saving and deleting of workspace files.

The NSW file system design includes tool specification of file attributes when getting and delivering NSW files. This processing has been added to WM-GET and WM-DELIVER. Further work is required in this area to remove an allowed default of TXT. The final design mandates that each NSW file have an operating-system dependent attribute of the form <O/S>-<type>; e.g., 360-PLI.

A significant optimization has been incorporated into the Works Manager's database management system. The keyword components of an NSW file name (such as COMPASS.SATTLEY.FOXIN.MAC) are now searched in an intelligent order (FOXIN/SATTLEY/MAC/COMPASS rather than COMPASS/SATTLEY/FOXIN/MAC). Clearly, lefthand keywords will appear often because of the hierarchical nature of the name space, and existing operating systems encourage or demand attributive (and therefore widespread) names as trailing keywords. To speed up the disambiguation process, then, keywords in the middle of file names are consulted first. Performance testing of this strategy shows that only 1/3 as many disk accesses are required to lookup a file name.

Finally, quite a number of minor problems have been identified during the baselining activity and fixed in the process of releasing system versions 4.1 and 5.0.

4.1 Works Manager

At present, the Works Manager consists of a number of identical concurrent instances of the same program, each one working on a single request at a time. All such processes share two common data bases, the Works Manager Table data base and the NSW File Catalogue. In addition to these processes there is a separate process, the Checkpointer, which makes periodic backup copies of the data bases.

The Works Manager supports 31 different Works Manager procedure calls, which are available to other NSW processes. These procedures are described in the Works Manager System/Subsystem Specification and the Works Manager Program Maintenance Manual.

The management/node manipulation tools are implemented entirely within the Works Manager. These are now invoked under the tool rights mechanism using the same interactive/HELP technique as batch tools. This has allowed removal of the specialized Front End/Works Manager interface formerly used, eliminating five special Works Manager procedure calls, and eliminating all knowledge of these tools from the Front End. Thus the UNIX Front End development is also freed of this knowledge.

The file attribute mechanism was also implemented, allowing Foreman calls to the Works Manager for getting and delivering user files to specify the file type. This feature was required to support future 360/91 interactive tool installation.

The Works Manager also uses the Global Configuration File (see 4.6), which has given it more operational flexibility. In particular, its timeouts on calls to remote procedures can be tuned without affecting other components. Also, event logging can be more flexibly specified, and better accommodates the divergent needs of the developers and operators.

The Works Manager, which consists of approximately 25.7K lines of BCPL code, is structured into a number of layers. At the top level, WMMAIN waits for a procedure call message from another NSW process, does initial decoding and validity checking of any such message, then dispatches the message to the proper routine. The Works Manager Routines, WMRTNS, implement the 31 Works Manager Procedures. At their disposal are a number of lower-level utility packages and subsystems. The Works Manager Table Package, WMTPKG, handles all interactions with Works Manager tables. It serves as an interface to the Information Retrieval System, INFRTV, which manages the NSW File Catalogue and the

Works Manager Tables. All NSW processes written in BCPL have available NSUPKG and BCPPKG. NSUPKG contains a number of facilities to handle MSG messages, create and record NSW fault descriptions, etc. BCPPKG provides basic utilities to handle character strings, do searching and sorting, and so forth.

4.1.1 Works Manager -- Future Directions

The following list of features covers most of the major changes anticipated for the next two releases of NSW.

- Modification of the LOGIN scenario to allow the user to terminate a previously crashed tool-session, if he finds his node "busy" when he attempts to log in.

- Modifications of the scoping and access-checking rules, according to the conclusions of the Redesign Effort (see Chapter 3, and Appendix).

- Direct file access - Use access and read access: Add two new kinds of NSW file access. Use access means that a user has undisputed rights to an NSW file. When he references the file he is given the NSW file copy - not a private copy. Any alterations he makes are immediately reflected in the file. Read access allows a user to read the actual NSW file copy - not a private copy. Thus it is suitable for data base files.

- Tool kits - When a user runs a kit of several tools on one host, the workspace should be left unchanged between tools. Thus, intermediate files can be passed from tool to tool without delivery to NSW file space. Both of these features would greatly enhance and optimize the use of local tools.

- Revision numbers - Design and implement a file revision numbering facility. This facility must be rich enough to support configuration management within NSW.

- History file - Implement the Works Manager routines to record information on the History File. Design and implement at least some interesting management/accounting routines which access this file.

- File groups - Extend the Works Manager command language so that whole groups of files can be copied, renamed, deleted, etc.

- Full file attributes - At present only the filename portion of the complete NSW filename can be used for retrieval. Also, the use of file attributes by tools is only permitted for the Global File Descriptor. The implementation of file attributes should be completed.

- Tool name extensions - The original concept of complete tool host transparency has proven unworkable. Thus, the notion of tool name should be extended to allow (explicit or implicit) host selection. By using the same mechanism as is used for files, the entire file lock system can also be used for tools.

- System status commands - The NSW user needs commands to interrogate system status and configuration: What tools are available? Which resources are up? What is the system load?

- File space maintenance and management - Operations aids to file space maintenance (e.g., reconciliation of the Works Manager's File Catalog with directories distributed at TBHs) and management (e.g., relocation of seldom used physical copies from hosts which are short of file space to those with surplus room) need to be implemented.

4.2 Checkpointer

The Checkpointer status mimics that of the Works Manager, since it consists largely of the entire Works Manager utility package, with a relatively small upper layer of code to implement the specific Checkpointer procedures. The performance improvements realized by the Works Manager Table Facility also apply to some Checkpointer procedures.

The Checkpointer has the following characteristics:

- Implements the FM-GUARANTEE call on the Foreman required by the Interim Reliability Scenarios.

- Manages NSW file deletion. Files deleted by the user are actually deleted by the Checkpointer after a time interval, as required by the Interim Reliability Plan.

- Makes Checkpoint files of all Works Manager database files at configuration controlled intervals.

- Is robust and flexible to about the same level as the Works Manager itself.

The Checkpointer received a major re-write for NSW 4.0, and except for one small bug-fix has remained unchanged since then. A completely new asynchronous remote procedure call handler was written. This allows the Checkpointer to make multiple simultaneous remote procedure calls, usually to delete files without interfering with the timing of database checkpoints.

The Checkpointer also uses the Global Configuration Database file, and has gained significant flexibility as a result. The external procedure call timeout, checkpoint interval, and waiting period before file deletion occurs are all under operator control.

The Checkpointer is halted by an interrupt from the operator utility OPRUTL (4.4).

4.3 Works Manager Operator

The Works Manager Operator has been extensively used with the 360/91 Batch Job Package since November 1978. Detail improvements have been made to improve reliability and job status reporting.

WMO shares a data base (the Job Queue File) with the Interacive Batch Specifier (IBS) module in the Works Manager. We intend to remove this shared access by making all access to this data base be via procedure calls on WMO, which will have sole access. To this end, direct access to the data base by the WM to get a batch job status (NSW: JOB) has been replaced by a call on WMO by WM on the (new) WMO-SHOWJOB procedure. Direct access to the data base by IBS will be replaced by use of a WMO procedure, WMO-ENTERJOB, to be specified and implemented in the future.

WMO also uses the configuration database file.

Some notable characteristics of the current WMO are as follows:

- WMO is responsible for both processing the Job Queue File and handling WMO procedure calls. These two tasks are handled by distinct instances of WMO in any given NSW system.

  (1) There is exactly one instance of WMO processing the job queues. A standard locking discipline guarantees that

precisely one such instance exists. This instance executes the job steps necessary to process a batch job, and initiates all procedure calls to external processes (WM, BJP, FP). It never receives generically addressed MSG messages.

(2) There are zero or more instances of WMO which receive generically addressed MSG messages, and handle all currently defined WMO procedures. These instances never execute job steps or initiate external procedure calls. Thus, these instance(s) provide external access to the data base.

- A primitive retry mechanism exists. WMO will retry an external procedure call indefinitely when it fails due to network or remote host crash. It will retry a failed external procedure call a maximum of three times if the failure is due to resource problems, e.g. no disk space.

- Status reports generated by WMO for display by WM (NSW: JOB) have been made more informative; all information supplied by BJP is reported.

- The maximum number of jobs in the job queue file is currently 64. This may be increased when needed, but requires re-compilation and reloading of WMO.

- The WMO cycle number may be set manually by the WMO utility (WMOUTL), but does not automatically increment with each cold start. "Cold Start" in this version occurs only when a new job queue file is created.

4.3.2 Works Manager Operator -- Future Directions

The Works Manager Operator program has proven to be quite reliable in the face of Works Manager, network, and batch-host failure; no significant changes have had to be made for some time. The following items are features which would enhance batch operation within NSW in general, and they will be considered for inclusion as more, and different, batch hosts are added to the NSW system.

- Background file motion - The delays perceived by the user when files must be transferred or reformatted can be significantly reduced by performing such actions in background mode.

- Job chaining - A desirable extension is to allow multiple batch tools to be run in sequence. Such a sequence should not be limited to just one batch host.

- Device I/O - A variant of background file motion is to have WMO control input and output from devices local to a user.

- Support of small (or non-NSW) batch hosts - Some hosts may be too small to support a Batch Job Package. Also, some hosts may be desirable as batch hosts but may not have the required NSW components (MSG, File Package). The Works Manager Operator should be extended to use existing Arpanet protocols (FTP, RJE) to submit batch jobs to such hosts.

- File groups - Extend batch tool specification facilities so that groups of related files can be supplied to batch tools.

4.4 Operator Utility - OPRUTL

The operator utility program, OPRUTL, has become sophisticated enough to be mentioned as a core system component in its own right. It can operate either stand-alone or as an actual NSW component under MSG. It allows the NSW to perform some maintenance functions which are tedious with more primitive developer-oriented utilities. Its capabilities are:

- To clean all or specified LOGIN entries out of the Works Manager database, e.g., to recover from a core system host crash.

- To enter new tool descriptors into the Works Manager database. This is the only practical means of entering batch tools, which must be parsed and error-checked on entry.

- To stop the Checkpointer via an MSG alarm.

- To report on current NSW usage, i.e., who is logged in.

- To reset all internal database locks (as a cleanup operations).

- To operate the Fault Logger, displaying selected portions of the logs on file.

4.5 Central Fault Logger (FL)

NSW 4.0 contained a prototype implementation of a centralized fault logging component, which has been upgraded in NSW 5.0. This component is intended largely to be an operational aid, by providing a single component through which the operator can receive fault messages from any component in the system - remote or local.

The Fault Logger design provides for sophisticated facilities to filter incoming messages and route them to several alternative or multiple destinations - devices, terminals, or files. The prototype FL simply maintains a fault message database in Arpanet mail file format, allowing mail handling tools such as HERMES to be used for information retrieval. The Fault-Logger Operator commands can be accessed on-line through the Operater Utility (OPRUTL).

The only component which currently logs faults to the FL is the TOPS20 Foreman. The next NSW release will see expanded use of a more sophisticated FL.

4.6 Global Configuration Database

NSW 4.0 included initial use of a prototype configuration database which supports specification and control of site dependent configuration data; no further development of this usage has proved necessary since that time, although some design work has been done on a generalization of the idea. It consists of a specially formatted text file containing parameters which apply both to a whole NSW configuration - hosts used, MSG generic classes defined, etc - and to a specific host in the configuration - directories used, timeout values, logging parameters, etc. The database is designed to be maintained at a central site and then be broadcast to all hosts in a given configuration, giving NSW operations a centralized means of controlling an entire NSW configuration.

The current database is a prototype used by most of the core system components and the TENEX/TOPS20 File Package. Other components are expected to implement use of this database in future releases. Configuration data now used include:

- WM - system herald, remote call timeout, event logging.
- CHKPTR - remote call timeout, deleted file wait interval, checkpoint interval, event logging.

- WMO - remote call timeout, event logging.
- FLPKG - remote call timeout, event logging, filespace directory name.

CHAPTER 5: TOPS20 TBH COMPONENTS

The TENEX/TOPS20 TBH is the most advanced of the three TBHs. All components (MSG, Foreman, and File Package) are substantially complete and tested; however, major enhancements are planned for the Foreman as part of the Redesign Effort and the NSW 6.0 task-list (see Appendix). All components are transportable between TENEX and TOPS20. Since all the TENEX systems which take part in NSW have been converted to TOPS20 systems, we refer in this report simply to "TOPS20" TBH components.

5.1 MSG

The MSG specification was produced in January 1976. It was revised in December 1976 - primarily to resolve ambiguities in the earlier document. It was extended in April 1978 to allow for support of multiple, concurrent NSW systems. The TOPS20 MSG component implements the revised and extended specification with only two exceptions (which are noted below).

The TOPS20 implementation of MSG is a single executable module which will run under (TENEX, TOPS20 Version 101B, and) TOPS20 Releases 3A and 4. In addition to the communication functions supported for processes (and defined by the MSG-process interface specification) the TOPS20 implementation includes a powerful process monitoring and debugging facility, and comprehensive performance monitoring software.

The TOPS20 implementation does not perform MSG-MSG authentication. Message sequencing and stream marking are not implemented (however the underlying software structure exists to support both).

A recent modification to MSG supports rapid timeout of attempts to contact remote hosts which are down or where no central MSG is running; this markedly reduces the wait time imposed on a user who has attempted to use an unavailable resource.

The implementation has also been modified to enhance its performance, based on extensive performance measurements completed this year. Changes include elimination of network connections for local message traffic, data re-structuring, reduction of calls on expensive JSYSes, and improved strategies for memory allocation.

5.1.1 MSG -- Future Directions

Very little additional effort is required for TOPS20 MSG. There are still some outstanding (but not pressing) MSG design issues:

- Details of MSG-MSG authentication - The general mechanism is as specified in the MSG design document of December 1976. However, the details of the ARPANET protocol exchanges are being re-examined.

- Maximum message size - The maximum message size is specified to be 65536 bytes (2**16). No implementation will accept messages that large. At present there is informal agreement to limit message size to at most 2048 bytes.

- Process creation - This issue was skirted in the original specification. However, a satisfactory solution must be found which balances the dynamic cost of process initialization and the static cost of maintaining unused ready-to-run processes.

- Optimization techniques - Compound operations like "send then receive" should be added, and some MSG code could be included inside those processes run under MSG to reduce context switching.

- Reliability techniques - Allow for multiple hosts, process classes, or process instances to be considered as recipients of generically addressed messages (broadcasting), so that the system can function better in the presence of "downed" hosts. The NSW Fault Logger is an example of a process which could make good use of such a feature.

5.2 Foreman

The current TOPS20 Foreman (Version 1615) implements all scenario functions defined by the Interim NSW Reliability Plan in its most recent revision (1 March 1977). The Foreman only supports tools which run in encapsulated mode. It does not yet support the direct use of NSW functions by any class of tools. It currently supports approximately twenty TOPS20 tools in this encapsulated mode. Some of these tools have been extensively tested and used within NSW; others have merely been superficially exercised.

The current Foreman implementation handles the "saving" of tool sessions (the "LNDSave" mechanism) when it loses contact with the Front End, or when the Front End asks that the session be saved (because the Front End has lost contact with its user), by copying the files in the

Local Name Directory into an archive directory; this frees up the workspace (directory) which had been in use. When the user later RESUMEs the saved tool session, if he requests REBEGIN -- which the Foreman now supports, incidentally -- or TERMINATE (with delivery of changed workspace files), the Foreman restores the files from its archive directory and delivers them to NSW Filespace. If the user requests ABORT, the Foreman deletes the archived files.

The TOPS20 Foreman has been extensively modified as a result of the extensive performance measurements made in early 1978 and reported in BBN report No. 3847, "A Performance Investigation of the National Software Works System". Performance enhancement has been currently limited to reducing resource consumption by the Foreman, e.g. by minimizing use of expensive JSYSes, pre-allocating workspace directories, etc. Future work will address alternative system support configurations, and altered patterns of NSW communications.

Since NSW 4.0, the Foreman has incorporated improved reporting of user file delivery from the tool. The Foreman also reports faults to the Fault Logger.

5.2.1 Foreman -- Future directions

Although the TOPS20 Foreman substantially implements the present specification, the enhancements planned in the Redesign Effort and task-list (see Appendix) have perhaps a greater impact on the TOPS20 Foreman than any other component, perhaps excepting the Works Manager itself. The following list contains the most important of these, plus some incidental improvements which have long been in the minds of the designers:

- Permanent integration of the TOPS20 mountable structures interface
- Coordinated Works Manager/Foreman protocol design and implementation to have common data base items reflect local resource management decisions
- Implementation of tool-specific encapsulated tool interfaces to handle tool peculiarities and improve performance
- Direct tool interface to NSW functions - i.e., non-encapsulated tool interface

- Design and implementation of a Foreman modified for on-line tool debugging
- Design and implementation of Foreman extensions for tool kits.
- Incorporation of some of the File Package's functionality in order to optimize file fetching and delivery operations.

5.3 File Package

The TOPS20 File Package is now functionally complete. The task of writing Intermediate Language encode/decode for non-TENEX binary format files is now complete, and has been tested with the CCN/360 File Package for several representative binary file types. The current File Package version has the following characteristics.

- All specified File Package procedures are implemented and tested for local, family, and non-family network transfers. Unspecified procedures to support the obsolete IP mechanisms in WMO have been expunged.
- The Intermediate Language (IL) encode/decode package has been re-structured for greater efficiency and maintainability. Encode/decode has been partitioned into three classes - text files, sequenced text files, and binary files; there is an encode and a decode module for each class, totalling six. Code size has increased, but both efficiency and code comprehensibility have been greatly enhanced. The interface between the (BCPL) calling routines and the (MACRO10/20) service routines has been simplified. Implementation of binary file encode/decode is complete, and has been extensively tested both against itself (i.e. against a remote TOPS20 simulating a non-TOPS20 host), and against the CCN/360 File Package. We have confirmed correct transmission of CMS2M object files from CCN/360 to TOPS20.
- Performance enhancements have been implemented based on the results of BBN's performance investigation as reported in BBN report No. 3847, "A Performance Investigation of the National Software Works System", Draft Version, July 1978 by Richard E. Schantz. We have minimized the use of expensive JSYSes, notably the CNDIR (connect to directory) JSYS (average cost 220 ms per call). We have done so by specifying that the File Package must be able to create/read/delete files in its own filespace and Foreman workspaces without connecting to them, and letting it

stay connected to its LOGIN directory. This has had no practical effect on the operation of NSW, beyond requiring that these directories be accessible from the system LOGIN directory. These enhancements have resulted in a CPU usage reduction of up to 60% for delivery of a file from the Foreman workspace.

- The logging of messages sent/received via MSG is under control of a spec in the Configuration Database (as in WM, WMO and CHKPTR). When logging is disabled, CPU usage for typical FP calls is reduced 25% - 40%. For comparison, the FP retrieval calls analyzed in BBN report No. 3847, "A Performance Investigation of the National Software Works System", Draft Version, July 1978, by Richard E. Schantz, which averaged about 2.9 seconds, can be reduced to as low as 0.7 seconds with logging disabled.

The File Package is written primarily in BCPL (approximately 6.9K statements including utilities.) The IL encode/decode package is written in Macro-10 and consists of approximately 1.7K instructions.

5.3.1 File Package -- Future Directions

In future releases, the capability which should be added is direct output of files to the user terminal. Currently, an editing or display tool must be run; a Works Manager command like "TYPE <file-group>" should be added to allow transmission directly from File Package to Front End. Another task which should be pursued is optimization of cross-net file transfers. The baud rate of such transfers should be improved and automatic restart and backup procedures in case of file transmission errors should be designed and implemented.

CHAPTER 6: UCLA IBM TBH COMPONENTS AND TOOLS

This section summarizes the status of the UCLA IBM NSW system by component package as of the end of this contract period. It corresponds to NSW Release 5.0, which will be the last release to support the IBM MVT Operating System. The tools available on the system are also listed and discussed.

Documentation covering the components of the UCLA IBM NSW system, and the tools availble to NSW users through the system, is of two types: UCLA technical reports and UCLA NSW documents. UCLA Technical reports are the direct reports to DARPA from contract personnel. UCLA NSW documents are produced for the NSW development or user community. They are delivered to the NSW Operations Contractor, in machine-readable form, for distribution upon request. The reader is referred to UCLA Technical Report TR-27, the Final Report for their NSW contract, for a complete set of references to these documents. (Citations in [square brackets] in this chapter refer to references in that report.)

6.1 MSG Central

MSG central [TR-12] is a set of routines that execute as a part of the UCLA ARPANET Network Control Program (NCP). It contains all the machinery necessary to create and destroy jobs, to execute the processes that are required to service generic message requests, as well as that needed to switchboard messages between executing processes on the local and remote hosts.

MSG central is written in Assembler language, using a set of subsupervisor interface macros that are specific to the UCLA NCP. It is not directly executable outside this context; however, the UCLA NCP has been successfully exported, and MSG could be exported concurrently.

MSG communicates with actual NSW processes via the UCLA interprocess communication mechanism known as the Exchange [TR-5]. Exchange is an exportable package. The other end of this communications link is managed by subroutine packages, so its elementary characteristics are not known to the using process.

6.2 The BJP Package

The BJP package consists of the modules that perform the Batch Job Processor function [UCNSW-207]. It allocates and frees temporary workspaces as batch jobs are created and finalized by the Works Manager Operator (WMO) core component. It submits local control-language data sets provided by WMO, monitors the submitted jobs, responds to status queries, and notifies WMO ansynchronously when a job is completed.

The BJP is implemented on the UCLA system as a continuously available single MSG process executing as a swapped TSO job. The single process is initiated by MSG central whenever it is itself initialized. The process name "BJP" carries the UCLA local "queue-generic-messages" attribute, which causes a single process to service all incoming generic requests directed to that name.

Because the MVT operating system does not itself include mechanisms for submitting jobs from internal processes, the MVT implementation of the BJP is tied heavily to UCLA modifications to MVT. These modifications are complex, and are intertwined with other UCLA modifications to the point that the BJP should not be considered exportable in its present form. This will not be a problem in the future implementation for the MVS operation system.

The BJP lacks a routine to clean up the queues, jobs, and datasets that may be left lying around after a change in WMO cycle number. The BJP does not clean up a directory at BJP-ENDJOB time.

The routine that reads the BJP's parameterization data set does not function correctly. Fixing this will probably be tied to implementation of the NSW Uniform Operations Interface concept.

6.3 The FOREMAN Package

The Foreman package consists of routines that implement the NSW Interactive Foreman function [ACC-FM]. It is executed on the UCLA system as a swapped TSO job, of which instances are materialized by MSG central on response to incoming generic messages.

Basically, the Foreman implements a "begintool" transaction which initiates a user-requested tool session. This transaction carries a "program name" which directs the Foreman to a special program stored in a local data base. This program is written in a locally defined language to be interpreted by the "Encapsulator Command Interpretor" (ECI) subcomponent of the Foreman [UCNSW-206]. The Foreman calls the

ECI to execute this program, which defines the file setup and cleanup and the "personality" of the tool. At indicated points in this execution, the ECI returns control to the Foreman to attach an indicated program, typically the native-mode tool program itself, as a concurrent process. On completion of that program, the ECI resumes interpretation of its program.

The Foreman also responds to "endtool" transactions. Because the tool and the Foreman are concurrent processes, the Foreman remains receptive to such transactions from the NSW core system during tool execution; however, due to the blocking mechanisms of the MVT operating system, both the Foreman and the tool are blocked when the tool is waiting for keyboard input from the user. This causes operational problems, and makes it inadvisable to terminate a UCLA-mounted tool by any mechanism except the tool's own voluntary termination command.

In the current implementation, the ECI operates as a synchronous subroutine of the Foreman, with the unfortunate result that the Foreman is not receptive to core-system requests during ECI operation.

Due to the characteristics of the MVT supervisor, the Foreman can not be sufficiently isolated from the tool. The kind of tool that runs undebugged programs can destroy the Foreman. Because of this, Foreman processes are never reused. Once a Foreman instance has completed, it is destroyed, and any subsequent Foreman request must create a new instance of the executing Foreman program in order to materialize a new MSG process.

Only encapsulated tools are supported, technically, and then only through pre- and post-processing by the ECI. The Foreman does not actually monitor tool execution, and no Foreman/Tool interface for "new" tools is presently defined.

Due to critical main-storage constraints in MVT/TSO, the Foreman can not maintain an LND, so recovery across crashes is not supported. In keeping with this, the "savelnd" and "rebegintool" procedure calls to the Foreman are not properly supported. They are accepted, but they function more or less like the "endtool" and "begintool" requests. The more esoteric "starttool" and "stoptool" are unimplemented only due to lack of demand.

The routines that read the configuration data set do not function correctly. Fixing this will probably be tied to implementation of the NSW Uniform Operations Interface concept.

6.4 The "FILE PACKAGE" Package

The "File Package" package consists of routines that implement the NSW File Package function [UCNSW-204, UCNSW-203]. It is implemented on the UCLA system as a swapped TSO job, of which instances are materialized by MSG central in response to incoming generic messages. Unlike Foreman processes, File Package processes are reusable. That is, when an instance of a File Package is complete, the same program instance will rematerialize as a new NSW process which can be used to satisfy another incoming generic message.

The File Package responds to five basic procedure calls:

1) The "import" procedure: moving or copying data into local NSW filespace from an external directory, another host, or the temporary workspace assigned to an NSW batch or interactive tool.

2) The "export" procedure: copying data from local NSW filespace to an external directory or a tool workspace.

3) The "send" procedure: copying data from any local data set to another host.

4) The "transport" procedure: copying data from an external directory or another host into another local external directory.

5) The "delete" procedure: deleting a file copy from any local filespace.

In servicing any of these procedure calls, the File Package performs operations that consist of combinations of several almost-independent capabilities:

a) Making equivalent copies of files.

b) Translating files from one local representaton ("file type") to another.

c) Moving file data between hosts.

d) Encoding and decoding the standard NSW Intermediate Language (IL).

e) Deleting copies of files.

Due to restrictions in the MVT data security system, the File Package (as well as any other NSW component) can only access data sets stored under its own charge number. Passwords cannot be used to override this constraint.

ASCII-type format effectors are not supported in other than internal forms. The tab descriptor of the GFD is lost when the GFD is relayed to a foreign host in a SENDME call.

6.5 Tools Installed

During the course of the contract, various tool programs have been mounted on the UCLA NSW system. Some of these were installed under IP and found their way into the full NSW implementation by default. Others have been installed through a careful design, validation, and documentation procedure. All were installed prior to the availability of a comprehensive tool-mounting procedure from the newly-commissioned NSW Tool Manager Contractor.

6.5.1 Well-Documented Tools

These tools represent the beginnings of a comprehensive workbench system for program development on IBM systems under NSW. Each is represented by at least one user document.

- Editing

  The TSOEDIT tool [UCNSW-107] allows the NSW user to create and maintain text data files on the UCLA system interactively.

- Browsing

  The DISPLAY tool [UCNSW-106] allows the NSW user to browse existing text data files interactively. It is particularly useful for examining the outputs of batch tools.

- Library Maintenance

  The "Interim Library Management" (ILM) tool kit [UCNSW-101] allows the NSW user to create and maintain library files on the UCLA system. Library files are not supported by NSW explicitly, but are required use IBM programming systems. Therefore, we support them implicitly through NSW tools pending the

implementation of UCLA's recommendations for direct NSW library support [TR-16]. There are presently ten tools in this kit. Some are batch, but most are interactive. There is also a document on installing this kit [UCNSW-212].

- o PL/I Program Development

  The PL/I tool kit [UCNSW-103] allows the NSW user to compile, link, and execute (with some restrictions) programs written in PL/I for the IBM Optimizing compiler [IBM-PL/I]. There are presently four batch tools in this kit. There is also a document on installing this kit [UCNSW-211].

6.5.2 Older Tools

The tools that remain in the UCLA NSW system due to their conversion from the IP system do not have formal user documents, pending their re-installation using tool installation procedures to be specified by the NSW Tool Manager. These tools are:

FORTRAN: compiles and executes a FORTRAN program.

ASM80: a cross assembler for the INTEL 8080 MPU.

PLM80: a cross compiler for the INTEL 8080 MPU.

MACRO20: a cross assembler for the AN/UYK-20 computer.

CMS2-M: a cross compiler for the AN/UYK-20 computer.

SPPCOBOL: a structured-programming package for COBOL.

6.5.3 Future Tools

The PL/I tool kit is intended to be the first of a comprehensive set of compatible tool kits supporting program development using the native languages of the IBM System/360 and 370 families [UCNSW-102]. These kits will support FORTRAN, COBOL, Assembler, and any other languages for which there are compilers available and interest in the NSW user community.

6.6 IBM 3033 MVS TBH -- The Future System

The personnel implementing and maintaining the UCLA IBM NSW system happen also to be the people responsible for the basic Arpanet software on that system, and hence are obliged to first work to bring the new machine (IBM 3033) back onto the Arpanet running with the new operating system (MVS), before turning their attention to the NSW software. Thus, the future directions planned for the UCLA IBM NSW system can be summarized as follows:

> Restore the functionality which was present in the 360/91 MVT version of the NSW system, but as reprogramming is required, do it in conformance with the best redesign ideas which have arisen in the course of implementing the previous version, and in accordance with the extensions planned to the NSW protocols and Management Plan.

For the individual components, this amounts to:

- The BJP must be redesigned and written from scratch.
- The Foreman must be heavily modified to gain the capabilities deliberately left out of the MVT version; however, it will run without those capabilities with little conversion.
- The File Package needs little conversion; however, when the extended protocols for FP-WM-FM interactions are ready, they will have to be implemented.
- MSG needs conversion to use IBM virtual terminals instead of the UCLA-specific ones in MVT.
- All subroutine packages need minor revisions.
- The tools all need to be re-installed, this time using more proper installation procedures, in coordination with the Tool Manager contractor.

CHAPTER 7: MULTICS TBH COMPONENTS

During the present contract period, the MULTICS NSW system has been materially improved, both in terms of the reliability and correctness of the components, and in terms of the number of tools available. Considered by component:

7.1 MSG

Previously, the Multics MSG server depended on an ad hoc, unsupported "Tasking Software" extension to the operating system; that Tasking Software is now supported by Multics TBH Support and has already been upgraded.

MSG has been modified for support of extended-leader host addresses, as required by current Arpanet protocols.

Many MSG bugs have been fixed (See appropiate STRs).

7.2 Foreman

The original version of the Multics Foreman was implemented to support tools which were written (or rewritten) specifically for use within NSW; the QEDX-RM tool, for instance, is a specially tailored version of a standard Multics editor containing explicit calls on Foreman functions where needed. As the emphasis in NSW has shifted more in the direction of packaging existing tools for easy cross-net access through NSW, an encapsulation technique has been developed and implemented for the Multics Foreman, permitting a larger number of tools to be made rapidly available (10, at present). Effort during the present contract period has been in two directions: improving the Foreman program to correctly execute all the relevant scenarios and connection protocols; and improving the encapsulation of individual tools.

The robustness of the Foreman has been increased, especially in the areas of detecting and reestablishing broken direct connections to the Front End.

Previously, the only safe non-ABORT method of terminating a tool session was to use the tool's own internal termination command; but now, the Foreman is able to respond to unexpected specific messages, allowing the QUIT TERMINATE and FM-LND-SAVE scenarios to work properly.

New tools have been added since the last interim report:
ada, ada-lstat, run-ada
-- Ada T & E Test Translator
smite
run (version 35.2).

Version numbers now work correctly, but the user interface with tools cannot yet make a direct reference to a file with the version number in that reference. Version numbers are there for status information only.

Multics tools follow an approach to the file system which differs from that of its TOPS-20 counterpart in that a file write operation is considered to be a global one, not just to a local workspace copy. Therefore, in the event of a crash, the highest version local copy will be equivalent to the global copy. This accomplishes two things:

a. Read and write operations appear to the user to be NSW global operations with the workspace copy as transparent as possible.

b. LND-Saving is not necessary since file delivery is done as needed, not at the end of a tool session.

Many FOREMAN bugs were fixed (See appropiate STRs).

7.3 File Package

The Multics File Package is a fairly reliable component. It conforms closely to the specification, and supports file encodement into Intermediate Language about as well as do the other TBH File Packages. Binary file transfer to non-Multics hosts is not supported, but is not required by any currently installed tools.

Local file types supported include:

a. MTX-TEXT which is like the default TXT file and assumes a stream file of ascii bytes.

b. MTX-LIBRARY which is physically a Multics directory. This file type does not support transfers to other hosts, i.e., it is PRIVATE.

7.4 Multics TBH -- Future Work

- Multics components are very close to being fully documented.
- A Batch Job Package is scheduled to be added to the Multics TBH software.

CHAPTER 8: Front End Components

There exist two operational Front End components for the present NSW System: The UNIX Front End, produced by Bolt Beranek and Newman, and the TOPS20 Front End (the "COMPASS FE"), produced by Massachusetts Computer Associates. There has also existed the "SRI FE", Produced by SRI International, which is no longer maintained, although it did successfully work with earlier versions of the NSW.

8.1 UNIX Front End

This component is produced under an entirely separate contractual arrangement, and has not as yet been formally incorporated into the NSW User System; Massachusetts Computer Associates, as Architecture Control Contractor, has not had any official responsibility for testing or evaluating the program. But as a part of our ACC duties, we have maintained regular communication with BBN personnel working on the UNIX FE, and all protocol changes affecting FE operation have been coordinated with their effort. Informally, we have operated the UNIX FE in the Development System (a test NSW system separate from the User System), and find it very sturdy and pleasant to use.

8.2 TOPS20 Front End

A user of the NSW User System (which is normally accessible only through the TOPS20 FE) will note some differences in what he sees on his terminal, compared to the system as it was at the beginning of this contract period:

- The Front End has an "Are You There" response, elicited by typing Control-T, which assures the user that the FE program is running, gives the date and time, and lists the connection state of each of the tools which are available for RESUMEing.

- All changes in the state of the connection to a tool are reported to the user as they happen:

  - "Awaiting connection" is typed every five seconds if the tool connection is not yet open when the Works Manager's reply-authorization is received;

  - "Connecting to <tool-instance>" is typed when the connection initially opens, when the user RESUMEs the tool after having

slewed away, or after the reply to a Help-request message has been sent to the Works Manager.

- "Disconnecting from <tool-instance>" is typed when the user slews away from the tool (with Control-N), or when a Help-request message arrives from the Works Manager on behalf of the tool.

o The functionality of the RESUME command has been extended in the Foreman, Works Manager, and Front End, so that the user now has the option to REBEGIN a saved tool (a tool session which was previously interrupted by communication failure).

o The former FASTOUT and MOVELOG operations have been subsumed as sub-commands to the LOGOUT command.

The most important changes in the TOPS20 FE, though, have been invisible to the user, having to do with the handling of network connections. Numerous errors and misunderstandings have been fixed, in a layered fashion -- since it was only after some were fixed that others were able to appear. The most important of these repairs were:

o The FE now no longer ever waits for a Close-connection operation to complete before proceeding. Previously, if the Close was part of the cleanup when a Foreman host crashed, the FE-host MSG process would wait for an excessively long time for an acknowledging Close from the other host, and the FE process would appear to be hung. Secondarily, the out-of-line handling of Close-connection operations speeds up the LOGOUT FAST and Autologout procedures. (This problem became clear only when an earlier misdirection of the lost-connection interrupt signal was discovered and fixed.)

o The Autologout scenario has long been the shakiest part of the FE program. Once the tool-connection handling had become reliable, it turned out that the Autologout procedure was still failing because of a mistaken treatment of I/O to the lost terminal. This has now been repaired, and lost connections to the user's host now reliably initiate the Autologout scenario as specified in the Interim Reliability Plan.

On the whole, the shortcomings of the present version of the TOPS20 Front End program lie not in failures to perform adequately the

functions it purports to perform, but rather in the absence of desirable functions and modes of operation which could be included. Most of these are in consonance with the changes called for in the Redesign document, and are discussed below.

The principal weakness of the present version of the program is that, if the Works Manager crashes, the FE will wait indefinitely for a reply to any message it might have sent to the Works Manager, and the user has no choice but to hang up his connection. To repair this shortcoming will take a restructuring of the message-handling procedures in the FE, which would be helpful in other respects as well, allowing multiple WM commands to be in process simultaneously, for instance, as the UNIX FE now does.

8.3 TOPS20 Front End -- Future Plans

A number of the changes proposed for the next NSW System Releases involve changes to the Front End (see Appendix). Correspondingly, the principal modifications projected for the TOPS20 Front End program are those required for the proposed system changes; with the possibility of making a few desirable local improvements along the way.

These projected modifications are:

- o A number of new commands, or additions of arguments to current commands, which involve no change in FE programming other than the recognition and translation of the commands themselves:
  - Additional arguments to the USE (service) command (host, workspace, arguments of call to the service invocation, etc.).
  - System status-information commands (show descriptors, current users, host configuration and status, etc.).

  Any other new Works Manager commands (service registration, etc.).
- o A TYPE command for typing text files directly on the user's terminal, without having to start up a service (e.g., an editor tool) to perform the typing.
- o Provision for direct connections to registered services other than through a Foreman process -- the "native mode" services and

ICP contact-socket tools.

- A command for "detaching" from a Foreman session at the user's request, that is, voluntarily causing the "saved session" activity which now occurs only when the user's connection to the FE is lost.

- Additional communication conventions between the FE and Foreman, covering user signals for talking to the Foreman rather than the tool, and for notifying the user that some output is waiting for him to see from a tool he is not actively communicating with.

- "Local" commands (not forwarded to the WM) for allowing the user to set his terminal-type and other parameters to improve FE handling of the user-interface communication.

- Complete recompilation of the FE program with the improved BCPL compiler; this will improve the efficiency of the running code, and permit a number of localized improvements to be made. It must be done, however, only in synchrony with the recompilation of the TOPS20 utility packages used in the Works Manager, File Package, Checkpointer, Operator Utility, etc.

CHAPTER 9: QUALITY ASSURANCE

9.1 Introduction

The objective of Quality Assurance is to provide continual, reliable access to a product whose limitations are both known and removable over time. It is the assurance that whenever a user accesses the service, it has at least the quality of the product previously accessed, and that notification of any trouble which might arise can be dispatched to a chain of organizations which can address it in a timely and effective manner. Attainment of these objectives rests upon Configuration Management (CM) - to ensure that the approved system revision is currently operational - and Software Trouble Reports (STR) - to respond to and coordinate the removal of limitations.

In this section, we discuss Quality Assurance methods for the NSW as a whole. Most CM is performed as a supporting service by Data Technicians, but response to and removal of limitations requires coordinated participation of many persons: users, operations, management, and developers.

9.2 Configuration Management

CM is to warrant that at all times, the service being offered is that approved by the Policy Group. As NSW is a distributed group of heterogeneous systems, an auditing style of CM is employed. That is, periodic inspections of the names and revision level of service components are compared with the system "inventory" as of its official release. Such auditing should ordinarily produce no exceptions and is best viewed as a watchdog who seldom "barks". Should changes appear, management can trace them to reduced limitations, new features, etc. The day-to-day operation can be delegated to the specialist organization; management personnel need only perform periodic audits.

The "inventory" in the NSW context is really the configuration index of a distributed, heterogeneous system. We decided to produce on each host a local configuration index, and then to copy those indices to a single host in order to construct a system configuration index. Each operating system has a tool suitable for constructing a local configuration index with some useful properties: text files, each line of which identifies a file in a given context (directory, library, etc.), by name and by the time of creation or most recent modification. TOPS-20, TSO, and MULTICS offer DIRECTORY, PDS, and library-map, respectively. Procedurally,

the Data Technician can index an NSW system by successively logging into each component host, running its local index-producing tool, copying the result to a single site, and finally composing the system configuration index by concatenating the local indices.

Every time an approved change is made, an index is prepared and retained. This approved index is used as a comparand in subsequent configuration audits. The periodic audits are performed by constructing the (current) configuration index and using TOPS-20's FILCOM tool for comparison. This is the motivation for single lines of text for each item described: if differences are found, FILCOM will display them in a fashion that is directly usable in pursuing the reason for the change. An example index and change notice are shown below:

```
Configuration Index for host USC-ISIC as WMH

     Configuration files (C. Muntz)

 MSG-GENERIC-NAMES.;103 585(7)  4-Jun-80 05:19:13

 MSG-NETWORK-CONFIGURATION.;102 73(7)  6-May-80 08:14:30

 CONFIG.BAS;514 1001(7)    10-Sep-80 02:40:13

 FORCOMFILE.BASE;3 512(0)  12-Mar-80 11:55:06

     Msg (R. Thomas)

 MSG.SAV;108161 48640(36)  10-Sep-80 02:33:46

     Front End (K. Sattley)

 FE .SAV;65201  33280(36)   8-Sep-80 05:11:33

 FETHDL.EXE;66300 36864(36)  8-Sep-80 05:23:25

 UNTLNT.EXE;3202 19456(36) 30-Jul-80 06:07:38

     Dispatcher (R. Schantz)
```

```
DSPCHR.SAV;1310 6656(36)  17-Apr-79 13:55:22
NSWROOT.SAV;2310 6144(36) 17-Apr-79 19:07:55
     File Package (C. Muntz)
FLPKG.SAV;25700 46592(36) 22-May-80 05:05:47
LOGUTL.SAV;4000 19456(36) 11-Mar-80 07:11:11
     Foreman (S. Swernofsky)
FOREMAN.SAV;1613 26112(36) 18-Jul-80 13:11:00
MKCOM.SAV;1611 6144(36)   18-Jul-80 13:25:20
LOGRED.SAV;1509 4608(36)  13-Aug-79 09:38:18
     Works Manager (C. Muntz)
WM .SAV;29602  120832(36) 17-Aug-80 15:18:17
CHKPTR.SAV;3401 86528(36) 20-Aug-80 14:47:07
WMO.SAV;16900  33792(36)  11-Mar-80 06:59:01
OPRUTL.SAV;1640 83456(36) 23-Jun-80 04:48:28
WMOUTL.SAV;14700 30208(36) 11-Mar-80 06:58:07
SIMWMT.SAV;5800 85504(36) 25-Jun-80 10:53:27
DMPUTL.SAV;1100 64000(36) 11-Mar-80 07:23:11
SIMINF.SAV;7300 58368(36) 25-Jun-80 10:55:01
DBSTAT.SAV;600 65536(36)  11-Mar-80 07:24:07
SIMWTF.SAV;1400 64512(36) 11-Mar-80 07:19:26
     Fault Logger (F. Ulmer)
FL .SAV;2010   36635(36)  18-Jul-80 13:09:18
```

FLOPER.EXE;2010 34304(36) 18-Jul-80 13:58:47

FLTEST.SAV;2000 25088(36) 20-Jun-80 10:19:18

... A Sample Change Notice

;CONFIG.INDEX;2 & CONFIG.WORK;56 9-Jun-80 0811 PAGE 2

```
LINE 35, PAGE 1
1)    FLPKG.SAV;25500 46592(36) 11-Mar-80 06:56:56
1)
LINE 35, PAGE 1
2)    FLPKG.SAV:25700 46592(36) 22-May-80 05:05:47
2)

LINE 41, PAGE 1
1)    FOREMAN.SAV;1610 26112(36) 10-Mar-80 12:20:35
1)
LINE 41, PAGE 1
2)    FOREMAN.SAV;1612 26112(36) 3-Jun-80 03:35:37
2)

LINE 49, PAGE 1
1)    WM.SAV;29300  120832(36) 11-Mar-80 07:12:15
1)
LINE 49, PAGE 1
2)    WM.SAV;29500  120832(36) 23-May-80 12:50:09
2)
```

Our original goal was CM auditing which could be performed by a Data Technician, with other personnel involved only if changes were found. If differences like the previous example are found, they may be forwarded to configuration managers for disposition; if FILCOM displays "NO LINES CHANGED" the CM audit has terminated with the watchdog silent.

9.3 Software Trouble Reports

If limitations are to be known and removed over time, each problem must be centrally reported and new ones carried in a journal for the life of the product. Consider the needs of different organizations:

- o User - Where can I report my problem? How can I work around it? When will it be fixed?
- o Developer - Are there any problems with my pieces? Here is a component which fixes problems a and b.
- o Managers - Which problems will be addressed by the next system release? What is the workload for each of the developers? Rank the outstanding problems by severity.

As contrasted with the CM watchdog who seldom barks, STR processing requires direct participation by nearly every project person. Each STR will be in a different state according to its progress and the subset of project staff affected.

Clearly, no existing tool could meet such needs, so MONSTR was specially designed and built - based largely on the Works Manager's database system, but operating over a computer-based problem journal instead of NSW's database of names, permissions, and file catalogue entries. This system is used by project personnel in all categories to report and deal with limitations.

9.4 Testing

The large number of interacting components found in a network operating system dictates a requirement for extensive testing. Each of these components must be unit tested, of course, and the set of components on a single host can be tested by usual methods. Once each host can function locally, the combinatorics of integration testing immediately arise. For example, the combination of hosts involved when an interactive tool needs to get a (possibly remote)

NSW file is roughly cubic in the number of hosts in the network: FE*FM*FP, where the three variables are respectively the number of hosts offering Front Ends, Foreman, and File Packages.

Our approach towards dealing with such a large number of test scripts is random testing by our Data Technician on a daily basis. Using NSW tools, we have written a program which - using a random number generator and tables of probabilities - chooses a different sequence of host locations and tool choices on each execution of the generator. (As the random seed is the time since midnight in hundredths of a second, duplicate runs are acceptably impossible.) The table of probabilities and two resulting test scripts are shown below. These scripts are then run by the Data Technician, but given availability of a testing tool, the script could be run automatically.

TEST CONFIGURATION IS AS FOLLOWS:

HOST NAME AND CHOICE PROBABILITY

USC-ISIC	18
USC-ISIE	18
UCLA-CCN	22
MULTICS	22
RADC-20	20

HOST NAME OFFERING TOOL NAME AND PROBABILITY

USC-ISIC	SOS-IC	15
USC-ISIE	SOS-IE	15
UCLA-CCN	FTN-UC	25
MULTICS	QEDXRM	30
RADC-20	SOS-R2	15

HOST NAME OFFERING FTP AND AND PROBABILITY

USC-ISIC	FTP-IC	30
USC-ISIE	FTP-IE	30
RADC-20	FTP-R2	40

HOST NAME OFFERING FRONT END AND PROBABILITY

USC-ISIC	30
USC-ISIE	35
RADC-20	35

```
  GENERATE TEST SCRIPT WITH 20 STEPS

LOG INTO NSW USING FE AT RADC-20
 USE FTP-IC TO SEND FILE TO USC-ISIE
 TRANSPORT FILE FROM USC-ISIE TO UCLA-CCN
 USE FTP-R2 TO GET FILE FROM UCLA-CCN
 EXPORT FILE TO UCLA-CCN
 TRANSPORT FILE FROM UCLA-CCN TO RADC-20
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 IMPORT FILE AT RADC-20
 EXPORT FILE TO UCLA-CCN
 IMPORT FILE AT UCLA-CCN
 USE FTP-IC TO SEND FILE TO UCLA-CCN
 USE FTP-R2 TO GET FILE FROM UCLA-CCN
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 USE FTP-IE TO SEND FILE TO MULTICS
 TRANSPORT FILE FROM MULTICS  TO UCLA-CCN
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 IMPORT FILE AT UCLA-CCN
 EXPORT FILE TO UCLA-CCN
 USE FTP-R2 TO GET FILE FROM UCLA-CCN
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
 EXPORT FILE TO MULTICS
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 IMPORT FILE AT MULTICS
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 USE FTP-R2 TO SEND FILE TO USC-ISIC
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 USE FTP-IE TO GET FILE FROM USC-ISIC
```

SCRIPT IS NOW COMPLETE

```
  GENERATE TEST SCRIPT WITH 20 STEPS

LOG INTO NSW USING FE AT RADC-20
 EXPORT FILE TO MULTICS
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIC
 USE FTP-IE TO GET FILE FROM MULTICS
LOG OUT OF FRONT-END AT USC-ISIC
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 EXPORT FILE TO UCLA-CCN
 USE FTP-IC TO GET FILE FROM UCLA-CCN
 USE FTN-UC TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 USE SOS-IE TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 EXPORT FILE TO USC-ISIC
 USE FTP-IE TO GET FILE FROM USC-ISIC
 EXPORT FILE TO UCLA-CCN
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIC
 TRANSPORT FILE FROM UCLA-CCN TO UCLA-CCN
 USE FTP-IE TO GET FILE FROM UCLA-CCN
LOG OUT OF FRONT-END AT USC-ISIC
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
 USE FTN-UC TO MAKE ANOTHER NSW FILE COPY
 USE SOS-R2 TO MAKE ANOTHER NSW FILE COPY
 EXPORT FILE TO USC-ISIC
 IMPORT FILE AT USC-ISIC
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
 USE SOS-IE TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - - -
```

```
LOGIN TO FRONT-END AT USC-ISIE
 EXPORT FILE TO RADC-20

   SCRIPT IS NOW COMPLETE
```

At the current time, only a few such test scripts have been run. During the following contract periods, we intend to run such tests on a daily basis.

CHAPTER 10: MONSTR -- A MONitor for Software Trouble Reporting

MONSTR, the MONitor for Software Trouble Reporting, was a response to the chaotic situation which existed in the area of bug reporting, tracking, and correction within the NSW project. During this contract period, work on MONSTR proceeded as follows:

- o April 1979 - November 1979: Writing and critiquing the A-level System Specification and the User's Reference Manual.
- o December 1979 - February 1980: Design of MONSTR as described in the User's Reference Manual.
- o March 1980 - July 1980: Implementation.
- o June 1980: Release of MONSTR 1.1 to NSW project personnel.
- o October 1980: Release of MONSTR 1.2 with increased reliability.
- o January 1981: Release of MONSTR 1.3 with archival capability.

Currently, MONSTR manages nearly 400 active STRs and is used daily to produce status reports which, prior to the existence of MONSTR, were virtually impossible to obtain.

The following commands are available in MONSTR version 1.3:

LOGIN	ACCEPT	ASSIGN-RESPONSIBILITY
MOVELOG	READ	STATUS
LOGOUT	HISTORY	SUMMARIZE
HELP	IN-BOX	CLASSIFY
CREATE	OUT-BOX	RETRIEVE
DISPATCH	RESPOND	REPORT
ARCHIVE		

MONSTR has most of the functionality described in the User's Reference Manual. The chief deficiency is the inability to FREEZE transactions, MODIFY or CANCEL them, and then THAW them to continue their progress.

MONSTR has exhibited some serious, unforeseen operational problems, due almost entirely to the fact that its database support is the Works Manager Table Facility and the Information Retr eval System. The WMTF and IRS are both well-written pieces of code, but MONSTR is exercising them at their design limits. At this time, the MONSTR

database is ten times the size of the Works Manager database (in number of pages), holds many more entries, and receives far greater daily use. Most importantly, MONSTR is an interactive program, which means the user can type control-C to abort at any time. The Works Manager, of course, is not interactive. This difference is crucial, because a control-C at an inopportune time can cause the WMTF or IRS to leave the database in a locked state, causing all MONSTR users to wait forever on an operation. MONSTR version 1.2 went a long way towards correcting this problem, but the database is still able to get into a locked state. This reflects the fact that the NSW database machinery was not designed to be used by an interactive process.

Another database problem has to do with the fact that the high volume of MONSTR transactions, combined with the WMTF's lack of a garbage-collection facility, leads occasionally to the situation where MONSTR "runs out of space", and the database must be reorganized before MONSTR can be used again.

To address these problems two steps have been taken. First, a number of changes have been applied to the WMTF and IRS (especially the former) to bring them more in line with the pattern of use MONSTR places on them. Second, a MONSTR operator performs a daily database dump to check for a locked database and to have a "copy" of the database in case the database must be restored. The latter event has only occurred two or three times in seven months.

Further MONSTR work should progress along a number of fronts:

- Database enhancements -- As described above, the database support for MONSTR is its weakest link. Modest changes in the Works Manager Table Facility will greatly increase MONSTR's overall reliability.

- Operator tools -- Currently the MONSTR operator has four tools: MONOPR, a very simple operator utility; and the three database simulation tools SIMWMT, SIMWTF, and SIMINF. The latter three lack key functions (e.g., printing the status of certain types of locks, resetting certain locks, etc.). The MONOPR tool (akin to the NSW OPRUTL tool) must be further developed to allow such things as STR deletion, adding new persons and organizations, etc.

- Addition of sets and working groups -- MONSTR must offer the ability to deal with named sets of STRs, sets which can exist for the duration of a session and across sessions. When a set exists

across sessions, it is generally of concern to a number of people who want to comment on the contents of the set. This capability we call "working groups", i.e., groups of people who wish to discuss some named collection of STRs (e.g. "all STRs to be fixed in the next release", "all File System STRs").

- User interface changes -- The user interface to MONSTR is via a collection of terminal handling routines used by the TOPS20 Front End. This has led to a slightly clumsy interface, with a lot of typing to produce a simple effect. MARGOT, a macro-based command-language interpreter-generator, is a tool which could be moved to the TOPS20 to provide a more flexible command interface to all TOPS20 components (Front End, OPRUTL, LOGUTL, MONSTR, etc.).

Appendix 1 -- ANNOTATED TASKLIST

This Appendix contains our conclusions about the importance and difficulty of implementing various of the features proposed in the NSW Redesign Study. It is especially directed toward providing a tasklist for the next NSW System Release. Appendix 2 will go into explanatory detail on the nature and effect of making the changes listed here.

We are now in the process of planning and designing for NSW Release 6.0. We currently anticipate the following changes/features to be part of that major system release. Task numbers following the enhancements refer to the RADC approved task list. Section numbers refer to appropriate sections in the accompanying design document (Appendix 2) which specifies the details of the new functionality.

A. Revised and Integrated NSW Resource Catalog Design and Implementation; Sections 1-9, this includes:

   - o improved and consistent naming, lookup and entry functions;
   - o revisions to support single integrated object space, including full support for file and service type objects (Task 4);
   - o revisions to the definition of semaphores (Task 9);
   - o revisions to the scoping mechanism (Task 17);
   - o addition of own space automatically created with new nodes;

B. Decentralized Protocols for NSW File Movement and Service Activation (Section 10.1).

   - o WM procedures for file/service entry, lookup and access control (Task 1);
   - o FM procedures for directly participating in file movement (Task 1);

C. Protocol Modifications to Avoid Timeout on Long Operations (Task 2) (Section 10.3).

D. Automatic Cleanup of Old Session on Re-Login Attempt (Task 3) (Section 10.6).

E. Limited user I/O commands (Task 5) (Section 11).

- o TYPE command to view text file on user's terminal.
- o PRINT command to send that file to line printer (UNIX FE and UNIX line printer only for this release).

F. Extended Services and Service Session Support (Task 7) (Section 10.8).

- o standard WS command interpreter (at minimum TOPS-20 TBH);
- o native mode services (at minimum TOPS-20 TBH);
- o limited single host tool kits and chained tools (at minimum TOPS-20 TBH);
- o revisions to support direct file access;
- o ICP contact socket tools;
- o detachable service sessions (at minimum TOPS-20 TBH);
- o automated service registration (TOPS-20 TBH minimum);
- o additional parameter collected and passed to the TBH on service initiation;
- o character string arguments collected as part of "USE" command line;
- o session id, user login name, service name, etc., available to TBH service;

G. Status Commands for the User (Task 11) (Section 10.2, 10.4)

- o ability to view static descriptors of file and service system objects, node and session records
- o list of logged-in users
- o status of configuration hosts (use "Inf NSW"?)
- o dynamic status of a user's active services and long transactions

H. TBH/WM Data Base Synchronization Improvements (Section 10.5)

WM comes up -> TBH FM (from config.bas)

I. Programmable Controlled Communication Between FE-TBH to Support Service Sessions, (Section 10.9).

- o selectable TELNET control sequences
- o selectable MSG alarms in place of control characters
- o selectable help alert signals

J. Other System Changes/Bug Fixes

- o MSG large host-address mode
- o avoiding user password recording in event logging
- o directly runnable (non-dispatched) TOPS-20 FE
- o TOPS-20 FM using GFT parameters
- o upgrade WM-BATCH-ENDJOB to propagate accounting list
- o Make WMO batch monitoring capabilities available as a service invocable from a TBH.
- o New BCPL
- o Additional compiler FE status Queries

Appendix 2 -- DRAFT SPECIFICATIONS FOR FUTURE RELEASES

This Appendix contains a November 1980 version of a developing draft specification for the changes to be made in NSW Release 6 (plus forethoughts about Release 7). The evolving document was originated by BBN, as is maintained by them, on the basis of frequent discussions with COMPASS personnel. This Appendix includes sections written by COMPASS personnel, and has been somewhat modified to reflect changes which were agreed to since this particular version was created. Some of the detailed specs given here are still subject to modification.

1. GENERAL FORM OF OBJECT NAME

The NSW resource catalog supports a single, uniform name space for naming all retained NSW objects. Objects are named as an ordered sequence of name components, separated by the special character '.'. It is useful to view an object's name as describing a path through NSW name space to the object. A complete pathname refers to a name beginning at the implied root of the NSW hierarchical name space and uniquely naming a single terminal object through the sequence of ordered name components.

2. KEYS

NSW access control is based on permissions (capabilities) held by the accessing agent. For controlling access to resource catalog objects, a permission (a key) may refer to either a single unique object in NSW name space, or all objects in an entire region of NSW name space.

Syntactically we write:

ABC.DEF.G as indicating a permission referring to the object uniquely designated by ordered name components ABC, DEF, and G. The object need not exist at the time the permission is created.

Or ABC.DEF.* as indicating a permission referring to the entire region of name space containing objects whose complete pathname begins with ABC.DEF. The region is evaluated at each access and includes all objects in the region at the time the right is exercised.

A key may include any number of '*'s, as long as it does not begin with one, and no two '*'s are consecutive. Thus the key ABC.*.DE.FG.*.H gives access to all names in the catalogue which start with ABC,

contain DE and FG consecutively somewhere, and end with H.

There are different keys for governing different kinds of access privilege. Three kinds of keys are defined for reading and writing the catalog itself. These keys are:

- o LOOKUP keys, required to do catalog object lookup operations in a given region (somewhat equivalent to directory read access on some systems),
- o ENTER keys, required to create a new object name in a given region (i.e. directory write),
- o DELETE keys, required to remove a current catalog object name in a given region (a specialized form of directory write),

These keys are applicable to all catalog operations regardless of the type of the object being manipulated.

Keys are stored with the node record to which the permission applies. In addition to these private keys, there is a system table recording public keys. Public keys are keys which system administrators have determined to be available for all user's of the system. A user's rights are determined by the union of his private keys with any public keys. (Public keys are merely a simple mechanism to both avoid replication and support convenient update of keys common to all users. The regions covered are part of the "system". For now, the public key table is modified only via system administrative means).

3. OBJECT ATTRIBUTES

All objects in the NSW resource catalog have attributes including (but not limited to) "type" and "site", where "type" is currently envisioned to be one of file, device, workspace, or service, and "site" indicates the host on which the object resides (site may be a list of locations in special cases). Every NSW object must have a unique name independent of its attributes. Objects cannot differ only in their attributes. Syntactically we can write:

A.B.C.D/type=device to refer to an object with name A.B.C.D which is of type device or

A.B.C.D.E/type=file;site=ISIE to refer to an object of type file and which is stored on host ISIE.

4. NAME LOOKUP

In general, manipulation of NSW resource catalog objects proceeds in three phases:

- o name and attribute lookup
- o general disambiguation (optional)
- o typed access control check

We can view the phases of object manipulation as a process of refining the set of objects which meet the specified requirements. Name and attribute lookup produces a set of one or more possible objects meeting the name and attribute constraints. This set can be reduced to a single object via a search rule or user help. This single object is the result of the lookup procedure, which is then checked for appropriate access control based on the type of object found and the type of manipulation requested.

4.1 FULL PATH LOOKUP

All accesses to the NSW object catalog requiring an existing object use the same lookup procedures, which are as follows for the case where a full pathname is specified:

(1) Verify that some lookup key is inclusive of the full-name object specifier) if none, lookup fails.

(2) Lookup the named object and return its catalog entry handle; if no such object, lookup fails.

(3) If attributes are specified, succeed only if object found has the required attributes.

When referencing an NSW catalog object, a user need not always give a full path name for the object. Two mechanisms exist which allow some name components to be omitted when referencing an object. The mechanisms are: scoping to support relative naming, and ellipses to support omitted component names. Object names employing either or both of these mechanisms are referred to as object specifiers (or object specs, or just specs).

4.2 ELLIPSES

When specifying an object name, the special symbol '...' may be used to indicate 0 or more components may be missing from the name as specified. Missing components may be before the first specified component, between specified components, or after the last specified component, with as many instances as required.

For example:

A.B...E, meaning 0 or more missing component names between the B and E component names.

A.B..., meaning possible missing component names after the B component, etc.

...A.B.C

...A.B...C.D...

A...B...C.D...E

are all legal object specifiers.

When an object specifier includes ellipses, the areas of NSW object space to which the user has lookup keys are searched for possible matching entries. The set of matching name entries is then reduced by any attribute requirements associated with the lookup; in particular, if the lookup specified a certain object type, only names with that object type are considered a match. If the set of matching objects has no members, the lookup fails. If the set of matching objects has exactly one member, that member is used in completing the operation (i.e. the operation specific access control check and subsequent object manipulation). If the object matching set has at least two members, user or program help can be invoked (if requested) to select a single member, which is then processed as above. If no help is provided when there are multiple matches, the lookup fails as it is an ambiguous object reference.

We refer to a name which includes ellipses as an incomplete name specifier, whereas a name which does not contain ellipses is a complete name specifier.

4.3 SCOPING

The resource catalog lookup function allows users (programs) to reference objects relative to user-selectable regions of NSW resource

space. These regions which are used as part of the lookup function are called scopes. A scope is a sequence of name components that designates the region of NSW name space consisting of names beginning with the scope's sequence of name components. Naming an object relative to a scope is much the same as directory-relative naming common on many systems. There is a single collection of scopes for each user, which supports all lookup operations regardless of the NSW operation being performed (Note 1). Active scopes for a user session are initialized from a permanent node record (Note 2). They can be modified either permanently in the node record or temporarily in the session record. All NSW object specs are looked up using scoping rules unless instructed not to by means of an explicit "unscope" request. Such a request is made by beginning the spec with the special symbol '$', indicating that the name is to be looked up in the context of the entire catalog. The convention to be used throughout this document and throughout the NSW implementation is that names which begin with a $ are relative to the root of the catalog, whereas names which do not have a leading $ are relative to the scopes in effect at the time. We sometimes refer to a name which employs scoping rules as a partial pathname specifier, whereas a name which does not employ scoping is a full pathname specifier. Scope regions are specified like keys, except that a scope must have only a single '*' special symbol, and only as its final component. A scope cannot be incompletely specified (i.e. contain ellipses).

Note 1: We would like to extend the design to support multiple collections of scopes which would be nameable and dynamically selectable for referencing NSW objects. However, in order to keep things simple for the present, we refrain from doing so at this time.

Note 2: When a new node is created, the scope field in the node record is automatically set to the own space which is associated with and created for the new node. For example, $ABC.DEF.* denotes the scope covering the region of NSW object space with initial name components ABC.DEF.

At any point in time the scoping set for a user can include:

(a) a single scope region

(b) a sequence of scope regions in which ordering is important

(c) a set of scope regions in which ordering is unimportant

(b) is referred to as a serial set of scopes, whereas (c) is a parallel set of scopes.

Functionally, scoped lookup (spec does not start with '$') proceeds as follows:

a. If there is a single active scope region in effect, look up the spec relative to the scope. This is accomplished by replacing the '*' of the scope with the given spec to form the full name specifier. Catalog lookup proceeds as described in sections 4.1 and 4.2 for names without and with ellipses respectively.

example 1: if scope = $A.B.*

and object specifier was C.D, scoped lookup would be equivalent to lookup of $A.B.C.D

example 2: if scope = $A.B.*

and object specifier was C.D.../type=file, then scoped lookup would be equivalent to lookup of $A.B.C.D.../type=file

example 3: if scope = $A.B.*

and object specifier was ...C.D, then scoped lookup would be equivalent to lookup of $A.B...C.D

b. If there are serial active scopes, lookup the user/program supplied object specifier relative to the first scope in the sequence,

if a single matching object results, the lookup succeeds and searching ceases.

if a multiple matches occur within the single scope, perform user disambiguation if provided; if disambiguation succeeds, the lookup succeeds and searching ceases; if disambiguation fails, the lookup operation fails.

if no object matches occur within the single scope, get next scope and repeat the above steps.

if no more scopes in the sequence, lookup fails.

c. If there are parallel active scopes, look up the supplied specifier as above relative to each of the scopes simultaneously. Create the set of objects which can be looked up with all of those full path specifiers. If no matching objects are found, lookup fails. If a single object is found, use it for satisfying the operation. If multiple matching objects are found, proceed as directed by user help to resolve to a single object. Then use it to satisfy the operation. If unable to resolve to a single object, lookup fails.

A single active scope is somewhat equivalent to the concept of a working directory common to most systems. Serial scopes are a slight generalization of search rule lookup, available on some systems. As specified here, NSW would support a single, unnamed scoping set, either parallel or serial. Obvious extensions would be to have more than one (named) scoping set, and/or allow user specifiable combinations of serial and parallel searches.

5. ENTERING CATALOG OBJECT NAMES

A set of rules similar to those for lookup apply when entering objects into the catalog. As with lookup there are various modes for doing this: scoped or unscoped names, with or without ellipses.

Ellipses in the name of an object being entered in the catalog serves to indicate that the name is to be "completed" in the context of the current catalog using the incomplete name lookup mechanism prescribed earlier. Scoped names with ellipses are resolved within the context of the current scopes only. Any incompletely specified name can only be used to replace an existing object (possibly creating a new version if that were available). It can not be used to create a new object name.

The catalog enter function which determines the complete name for an entry spec is a slightly modified version of the NSW object lookup just described, and works as follows:

Find a set of candidate complete full path object names by applying the basic NSW lookup mechanism to the entry spec.

1. If the resulting set contains a single object, the complete name of the existing object is used for the entry operation.

2. If the resulting set has no objects then

   - o if the spec contained ellipses the entry operation fails
   - o if the spec was already a full path specifier (i.e. beginning with '$') use the spec as the complete entry name
   - o if the spec required scoping and there was a single scope region, the complete name is the spec appended to the scope
   - o if the spec required scoping and multiple scope regions are being used in series, the complete name is the spec appended to the first scope region
   - o if the spec required scoping and multiple scope regions are being used in parallel, the complete name is determined by user disambiguation if available; else the operation fails

3. If the resulting set has no objects, the operation fails. If the resulting set has multiple objects, perform user disambiguation.

An Enter right to the appropriate region is required as well as a Delete right if an object is being replaced, in addition to a lookup right. At the catalog lookup level, the type of the object can be required to match or not depending on whether the attribute "type" is included with the specifier or defaulted by the operation. Actual replacement of catalog objects can be confirmed or not based on parameters of the operation. For entering new objects into the catalog (as with accessing existing objects) the catalog object name phase may be followed by type and operation dependent access control checks before the operation can complete successfully.

[It will be important for users to understand that with serial scopes their first (or only) scope should include enter rights to allow the system to automatically create objects on their behalf (e.g. a temporary workspace name, a profile, etc.). Having the system automatically find a region to which the user has ENTER access seems like a bad idea. Another alternative might be to add the concept of a "login" region, as distinct from a "connected" region which scoping represents. We do not plan to pursue this further at this time.]

NSW 6.0 will employ all of the preceding naming and lookup conventions for all catalog operations. In addition, existing user interfaces and system commands will be upgraded to reflect the modified operation of the catalog software.

System changes:

- Modify WM catalog maintenance procedures to support new naming lookup and scoping conventions.
- Modify (as necessary) ALL NSW components associated with the user interface to support the naming conventions when displaying NSW catalog names and regions.
- Front End commands which must be modified are:

  ALTER (scopes) : to support augmented scope types
  SHOW SCOPES : to remove access type
  SHOW FILES : to remove access type

- Modify WM software to support public access regions.
- Modify WM software to create private own space with node creation.
- Add commands for permanently modifying scopes in the node record.

6. NAMING GROUPS OF OBJECTS

The name lookup and entering conventions discussed so far are intended to identify and name a single catalog entry. At times, there is also a need to succinctly name a group of catalog objects (a plural name), usually in conjunction with some form of processing common to the entire group. The SHOW OBJECTS command is perhaps the most prominent example of where a designator is often used to denote a group of matching objects instead of being resolved to a single object. The syntactic form of a plural name is equivalent to that of a partial name specifier using ellipses, with the exception that the special character '*' is used instead of the special character '...', to denote objects with zero or more missing component names in place of the '*'. We have already indicated an important special case of the plural naming convention when referencing key and loop regions. A general plural name can have multiple '*' components, including preceding and trailing any specified name parts. '*' can appear only in the name part of a

plural object specifier. '*' is implied for all unspecified attribute fields. The matching set can be reduced by including particular attribute values with the object specifier.

example $A.B.*.C/type=file

would refer to the collection of file objects to which the user had lookup access, which began with component names A.B and terminated with component name C. If the name in the example did not have a leading '$', the lookup would be with respect to the scope setting prevailing at the time. Parallel scoping rules applied to a plural name generates a collection of objects which is the union of the plural name applied to each scope region. Serial scoping rules applied to a plural name generates the set of objects matching the name for the first scope which has a non-empty lookup result.

example: Serial Scope in effect:

$A.B.* $A.C.* $A.D.*

Catalog contains objects {$A.C.E, $A.C.F.E, $A.D.E, $A.B.G}. The plural name *.E would refer to the set {$A.C.E, $A.C.F.E}. Had the scopes been parallel, the appropriate set would be {$A.C.E, $A.C.F.E, $A.D.E}. We defer further discussion of the use of plural names of this type within the context of NSW primitive operations.

NSW 6.0 will use and support plural naming conventions for all keys and scopes, and objects referenced with the SHOW command.

System changes:

- Modify WM software to process the new form of plural name for scopes, keys and object lookup.
- Modify WM software and (if necessary) Front End software to accept and display plural names in the commands which accept or display keys, scopes and sets of objects (currently only SHOW).

7. ASPECTS OF THE FILE SYSTEM MODEL

NSW supports a copy model for workspace file processing. WS-CI's always make local copies of NSW files to be used for supporting a service session. In addition, unless otherwise requested by the accessing service, a tool reference to an NSW file will result in a

nameable workspace copy of the file. A copy is a snapshot of an NSW file at a given instant in time, allowing for the possibility of some host or service dependent data transformations. The NSW system does not maintain the mutual consistency of file copies. Optimizations to support read only access where by the file is not actually copied when a locally accessible NSW file space image is already available, are supported; however the copy semantics must still prevail from the perspective of the accessing service. The copy semantics provides uniformity across translated (information lossy) and non-translated file movement, and across hosts which do not maintain local NSW file storage resources. A tool can specify a read only file access request, and to the extent enforceable or believable on the service host, NSW IBH Software can make use of an optimized non-copy access path to the NSW file space image for satisfying the file accesses. The optimization has no effect on the NSW access control mechanism.

[Comment: the workspace copy model serves as a mechanism for hiding some of the problem areas in the fundamental functionality of the NSW file system and file system transfer capabilities; in particular, lack of host independent access methods, some information-lossy file transfer modes, and automatic tool-specific file conversions make developing an integrated heterogeneous file system very difficult.]

7.1 FILE IMAGES

Users can request that an information-lossless image of an NSW file be MOVED from its current NSW storage host to another NSW storage host (if such a transfer is possible). The system understands the equivalence of the original physical copy and the new physical image. Files entered (imported) into NSW file space are always marked as "original." Images maintained by the system as a result of a MOVE operation are marked as a "user-directed-image." In addition, to support the copy semantics for workspace file access, images of NSW files are often transported from a storage host currently supporting a physical copy to one which needs to use the file. When this occurs, and when the destination host is an NSW File Bearing Host (FBH), the destination may at its discretion retain an information-lossless image of the original to serve as a cached copy. Images maintained by the system as a result of tool oriented file movement are marked as a "system-directed-image." The retention of a cached image must be coordinated with the central catalog.

The NSW file system maintains the equivalence of all physical file images, and can use any one to satisfy a user/tool file request. Users can direct the system to use a particular image by specifying the host

attribute when referencing the file. Users alone are responsible for managing (i.e. moving, deleting) the disposition of originals and user-directed images (within allocation limitations, of course). Users are "charged" for storage associated with originals and user-directed images. The system (the FBH) manages the collection of system-directed images that it has decided to cache. A FBH may at any time, assuming the proper coordination with the central catalog, delete a cached image in accordance with its cache management policies and current file space demands. The central catalog process (WM) may also initiate the deletion of cached images using the same mechanisms as for supporting user deletion of originals. Caching represents an area in which the system performance may be tuned; updated file access protocols to support FBH file caching are given in a later section. Users are never "charged" for system-directed-images.

On any lookup operation the user may limit the selection of an image through the host attribute. Without a specified host attribute, the system may select any image interchangeably. If a particular image is specified but unavailable, the operation will fail. Replacing or otherwise modifying a file catalog object invalidates all existing images of the file. The Delete operation defaults to all images of the file.

7.2 IMMOVABLE OBJECTS

When entering a file object in the NSW resource catalog (IMPORT, DELIVER), the object may be marked as "immovable." Existing files may also be subsequently so marked, according to appropriate access control and prevailing file state. An "immovable" file cannot be copied using NSW operations, nor can an image of it be made at other NSW file space supporting hosts. Such a file can be accessed only by services that run co-resident on the file host, and only via direct access methods.

7.3 DIRECT FILE ACCESS

A service can request direct access to an NSW image of an NSW file. Such access can be granted only if:

- the TBH can support this mode of access without violating the integrity of the NSW file system
- a file image exists in local NSW file space or can be moved to the TBH NSW file space (e.g. it is not marked immovable)

- o the file can be used by the service without any service specific transformations
- o appropriate catalog file locks can be set
- o user/tool has appropriate modify access rights

If any of these conditions are not met, the operation will fail. Granting direct access to an NSW file image invalidates all other images of the file, and causes the image which is directly accessed to be marked as the original. The NSW does not itself support the primitives for file access methods (i.e. for referencing internal file data). Direct file access is provided only through existing TBH specific file access methods where appropriate.

File catalog locks must be explicitly or implicitly released when the direct file access completes.

7.4 FILE SEMAPHORES (locks)

Users/services can request that a "lock" be set on an NSW file to control access to it as it undergoes modification. Locks can be set for exclusive access, multiple-reader/single-writer access, or as a warning, to support alternative patterns of shared access. The duration of the lock can be indefinite (until explicitly cleared), or can coincide with the lifetime of the setting activity (service, or user session).

When accessing an NSW file object, the operation proceeds in the following phases. First, the name is looked up in the catalog to identify the object. Then the appropriate object specific access control check is applied. Finally, and only after completing these initial phases, any requested locking specification is checked for consistency with the current lock state. If the use is consistent, the operation can be completed. If not, the operation fails.

NSW 6.0 will support read only file access, direct file access, immovable objects, and full-file semaphore functionality. It will also support multiple image file management functions.

System changes:

- o Support protocol addition 10.2 and and modify the GET and other additional parameters.

- o Augment (if desirable) TBH software to make use of protocol additions in support of enhanced file access procedures.

8. NEW OBJECT TYPES

In addition to the object type "file" and object type "service" (extended from the NSW 4.1 type "tool") which are currently supported by NSW, device and workspace objects are also to be entered in the common NSW resource space. All object types can appear anywhere in NSW name space. The system implementation uses two regions $SERVICES.* and $DEVICES.* as repositories for generally accessible user services and line printer devices. Users may have their active scopes set to reference regions within $SERVICES.* and $DEVICES.* to facilitate naming these system supported services and devices.

However, because these two regions represent important system wide functions, we provide special purpose mechanisms to make the expected common access patterns easier without unduly burdening the single scoping mechanism, or requiring support for multiple named scopes. The first mechanism provides for conveniently looking up system wide service objects within one or more regions of $SERVICES.*. The mechanism is to have the system automatically append in serial scope fashion the relevant regions of $SERVICES whenever a scoped request to instantiate a service fails to find a matching object. In effect, this merely causes the system to search designated system service areas when instantiating a service if it cannot be found in the user specified areas. It is the scope equivalent to public keys. In a similar fashion, operations which specifically refer to using device specs will search a public $DEVICES* region, if lookup using user scopes fails.

The second mechanism provides for the session record containing an optionally specifiable default line printer device name which is initialized from the node record and can be used in conjunction with the PRINT file command. The default line printer would be modifiable temporarily or permanently like scopes. Since we are unlikely to support anything beyond lineprinters, and since one typically accesses only a single line printer, this mechanism should be sufficient for quite some time.

Permanent workspaces (when supported) are cataloged anywhere in user accessible NSW space at the time of registration. Temporary workspaces are automatically cataloged by the system, under a designated name entered in the appropriate region according to the user's scope setting in effect at the time the workspace is assigned. When externally

referencing a file within a workspace, the NSW set file notation is used i.e. WORKSPACE-SPEC!WORKSPACE-FILE-NAME, where WORKSPACE is looked up under normal NSW name lookup conventions, and WORKSPACE-FILE-NAME is interpreted within the host workspace context.

When interacting with a service within a workspace, the naming context is automatically set to be the service workspace itself. There are two mechanisms for escaping from this implied "scope":

- An NSW wide convention for specifying names relative to the NSW context (where the NSW scoping mechanism previously discussed is in effect). The convention is the use of a leading special symbol ('^') when entering an object name. This feature may not be supported for all interactive services on all TBH's.

- Optional host specific escapes to reference non-workspace objects on the native host e.g. <directory> as a workspace escape for native mode on TOPS-20.

NSW 6.0 will support a system wide service region to be used when instantiating NSW services. This version will also support a standard "unscope from workspace" naming convention. Supported devices will be temporarily limited to line printers directly connected to UNIX-FE hosts, or TOPS-20 FE hosts, as devices private to their implementation (i.e. directly associated with).

System changes:

- o Add $SERVICES.* public region to lookup for the USE command in the WM.

- o Modify interactive NSW mode IBH software to recognize and handle "escape to NSW" convention.

- o Add TYPE and LIST commands to FE, defaulting always to users TTY and local LPT (if any); requires protocol change 10.2.

9. OBJECT TYPE SPECIFIC ASPECTS OF THE RESOURCE CATALOG

Although all object types are entered into a common name space, and are referenceable via a common mechanism, there are object type specific operations, permissions, etc., which have meaning only when referencing an appropriately typed object within an appropriate operation. (In some cases, operations and permissions may be defined over more than

one object type). In this section we attempt to enumerate a minimal set of operations and permissions for the anticipated NSW object types.

Note: the lookup, enter and delete permissions refer to the name space itself, regardless of object type.

For initial implementation, permissions to create and delete specific object types are subsumed by general catalog enter and delete permissions. Further control is provided by type dependent permissions governing the use of the cataloged objects.

Object Type = File

Appropriate Permissions:

COPY: access required to read a file and/or make a copy of it

UPDATE: access required to change the contents of an NSW file object (either through replacement, or direct read/write access)

Appropriate Operations:

(these operations default their object type attribute to "file").

COPY
RENAME FILE
DELETE FILE
GET/DELIVER
LOOKUP FILE
ENTER FILE
IMPORT/EXPORT
SHOW FILES
SHOW DESCRIPTOR OF FILES
MOVE
...

Object Type = Service

Appropriate Permission:

EXECUTE: access required to instantiate the service

Appropriate Operations:

(These operations default their object type attribute to "service").

USE
SHOW SERVICES
SHOW DESCRIPTOR OF SERVICES
LOOKUP SERVICE
ENTER SERVICE
RENAME SERVICE
DELETE SERVICE

Object Type = Device

Appropriate Permission:

APPEND: access required to print a file

Appropriate Operations:

(These operations default the object type attribute to "device").

TYPE
PRINT
SHOW DEVICES
SHOW DESCRIPTOR of DEVICES

Object Type = Workspace

Appropriate Permission:

ACTIVATE: access required to instantiate a service in a designated workspace. A temporary permission for the workspace is added to the node record for each NSW workspaces assigned to the user as a result of invoking a service.

Appropriate file permissions govern access to the workspace files for external manipulation via set file notation.

Appropriate Operations:

(These operations default the object type attribute to "workspace").

```
SHOW WORKSPACES
SHOW DESCRIPTOR OF WORKSPACES
ACTIVATE
```

OBJECT INDEPENDENT OPERATIONS:

Some operations can be invoked uniformly across all object types. These operations do not default the type attribute, and include:

```
SHOW OBJECT
SHOW DESCRIPTOR OF OBJECT
DELETE OBJECT
RENAME OBJECT
```

NSW 6.0 will include full support for object types file and service. It will also include support for object independent operations as they pertain to files and services. Generalized workspace and device object support will be deferred until Release 7.0.

System changes:

- Add object descriptors for service objects.
- Add a set of commands which support services in a manner similar to file, e.g., SHOW services; Rename services, etc. (in addition to existing USE command).
- Add/modify support for keys of TYPE COPY, Modify (files) and Execute (services), including in ASSIGN-RIGHTS.
- Add commands to support SHOW Descriptors of files and SHOW Descriptors of services.
- Add command to support Register Service objec

10. PROTOCOL CHANGES TO SUPPORT NSW RELEASE 6

In this section, we propose a number of protocol changes to support the file system enhancements mentioned, and promote other architectural enhancements to be included in the next major NSW system release (NSW 6.0 which is to include catalog reorganization, WS-CIs, non-FP file manipulation, file transfer timeouts, etc)

10.1 FILE ACCESS

The following three new WM calls are intended to support decentralization of the file access and file catalog update procedures. The primitives are to be used to support the inclusion of file access capabilities in Foreman components, and to allow FE's access to the NSW file system. The implementation approach illustrates an architectural trend whereby the WM is used for catalog lookup and access control, but the object manipulation is moved closer to the requesting agent. The addition of this functionality is completely backward compatible with the current file access protocols. The form of each primitive is identical, grouping the parameters into process identification data, exception handling data, logical object data, and (if appropriate) physical object data.

10.1.1 ENTER-FILE-OBJECT

Accepts:

Identification: session id or service id

Exception
Handling: help process

: disambiguation control parameter

: confirmation control parameter (what to do if there already is an object by this name i.e. use help confirmation);

Logical: file specifier (subject to complete name lookup rules; fully or partially specified;)

: attribute list for the new object; some attributes cannot be set from this list (e.g. time of creation) and are automatically inserted by the WM; attributes include possible marking as

immovable object; attribute type=file is implicit with the primitive;

: lookup hint

Physical: physical file descriptor (for the new object in NSW file space);

: gft

: size of file (bytes + byte size)

Returns:

Confirmation: success/failure

Logical: full path name object was put away as

: scope (region) which was used to create new entry (if any)

: lookup hint

NOTES:

The Enter-File-Object primitive is intended to support the catalog update part of the DELIVER (IMPORT) operation by processes within the "security kernel." Attribute type=file is implicit with the primitive. Requires enter access + update access if actually replacing a file. "Identification" will generally be service id.

gft and gfd are copied from existing WM/FP primitives and could possibly be subsumable by an attribute.

"help process" is one of:

(a) the calling process

(b) process designated as controller from the session record (usually the FE)

(c) an arbitrary process id

(d) no help process

Disambiguation control and confirmation control are parameters for allowing the calling process to control the use of help messages for ambiguous file references and operation confirmation. Both the full pathname of the object created and the scope (if any) which was actually used to create the full pathname are returned to support flexible service use of these individual name parts.

LOOKING AHEAD SOMEWHAT

At some point in the near future we are likely to try to support a modified form of the DELIVER scenario in which a component on a non-File Bearing host (e.g. UNIX FE) can directly provide a copy of a local file for inclusion in the NSW file system. We anticipate further modification to the Enter-file-object primitive to support this capability in the form of an extended PCD parameter. There seem to be two possible approaches. One is to have the PCD refer to a connection ID on which the donating process (i.e. the FE) is willing to send the appropriate file when requested by a SEND-ME message from a FP selected by the WM. The other is to have the FE component first interact with an FP on a FBH (which FBH is an issue here), and then have the PCD returned with the Enter-File-Object refer to the FBH copy. We defer further design on this topic for now.

10.1.2 ACCESS-TO-FILE-OBJECT

Accepts:

Identification: session id or service id

Exception
Handling: help process

: disambiguation control parameter (including how to handle lookup failure)

: confirmation control parameter

Logical: file spec (subject to general lookup rules);

: attribute list (those attributes,if any, which are to be used as disambiguation; attribute type = file is implicit with

the primitive);

: access type (copy, direct);

: locking data (type, duration);

: lookup hint (optional) valid only with full file name specifier; see note below;

Returns:

Confirmation: success/failure

Logical: full NSW file name of object looked up;

: scope employed to locate object (if any);

: attribute list including creation timestamp, or version (if we ever support that);

: new file specifier (if any was obtained);

: lookup hint (optional; can be used with any future reference to the file object; see below);

Physical: physical file copy list (including a marked original) gft gfd

NOTES:

The Access-to-File-Object primitive is intended to support the catalog lookup and access control check part of a file access procedure. It is intended that the FE or FM process use the returned object descriptors to interact directly with the appropriate FP to obtain a copy of the file (or access a local copy if accessible). When invoked from a FE, identification will typically be session id; from a FM, identification will typically be service id; alternatively, {user name + password} strings can be supplied as authentication data. In this case, a pseudo user session, lasting for the duration of the operation is created. If password is omitted, it can be obtained via user help, as specified. The file specifier and any listed attributes are used as augments to the lookup function. If a single object results, the access is checked based on the access type requested. If the accessing agent has the

appropriate permission, the locking data is checked for consistency with current usage. If {user + password} authentication is supplied, only indefinite duration locks can be set. A failure at any of of these steps results in a failure of the operation except for the case where lookup results in an empty set and the caller has requested via the disambiguation control parameter that the system gather an alternative file specifier (as requested by UCLA). In conjunction with this new primitive, each FP on every FBH must be modified to accept file movement requests from non-WM kernel components (if they do not already do so).

When this type of request is issued by non-kernel (FE) processes, additional authentication messages may be required to achieve a secure system. Alternatively, it seems possible to develop a scheme whereby for non-kernel processes the Access-to-File-Object returns an encrypted capability which can be presented to and validated by the appropriate FP for retrieving the file. This approach would not change the pattern of communication for non-kernel processes, but would introduce additional authentication overhead in all of the components. We defer further consideration of system security at this time in order to introduce new functionality without requiring system wide changes.

LOOKUP HINTS

The lookup hint is a means whereby the WM can provide an internal handle for a catalog entry in the WM data base to another NSW component when an object has been looked up in the catalog. The form of the hint might be an internal catalog identifier or virtual memory address. The idea is that when and if that object is subsequently subject to another lookup operation, the hint may provide a more efficient path to the appropriate catalog entry than would be the name specifier. Properties of the hint mechanism are that it is never required, and the lookup will logically yield the same result whether or not the hint was used.

The motivation for including a lookup hint type of parameter at this time is to solve the following problem related to supporting autonomous File Bearing Host (FBH) based file image management. By separating the file name lookup transaction from the physical copy manipulation transaction, we have also effectively removed the WM from the reply loop for indicating that an additional NSW file image is being retained at the FBH (in the case of an inter-host file reference). When the FBH software decides to cache a file image after an inter-host transfer, the physical descriptor data must be reported back to the WM for inclusion in the catalog so the image can be used to service subsequent references to the file.

There seem to be two ways to do this:

(a) conditionally including two additional specifically addressed messages (cached image data and confirmation) with the definition of the access-to-file transaction; or

(b) defining an independent file-image-update transaction which can be generically initiated when and if the cached image is retained;

The problem with the first approach is related to the decentralization of the caching decision. Since the decision is made by the FBH component, it must alert the WM of its intentions to cache an image along with the request for lookup so that the WM will continue the transaction beyond the reply to the lookup. This is made a little tricky since the desire to cache is predicated on there being a transfer to begin with. That decision lies with the referencing component and may not be made until it views the PCD list. We reject as too error prone an extended transaction based on characteristics of the FBH caching implementation matching against the PCD list returned to it. A scheme whereby the WM always waits some time out interval for a possible additional reply is also feasible, but would seem to be adversely effected by high load situations under which caching might be all the more important.

The problem with adding an independent file-image management transaction seems to be the high overhead of recreating the context for referencing the file in question at the catalog. One important use of a continuing specifically addressed transaction is that the context associated with handling the particular operation is maintained within a (WM) process state or "hot" data base entry, and need not be recreated with each phase of the activity. Generic message sending does not seem to be significantly more expensive than specific message sending. Also, the current inter-message processing interval is very large relative to internal virtual memory management intervals, so the appropriate pages and memory maps will likely be swapped independent of whether it is handled by a single continuing process or an independent process. The major impact of an independent transaction is expected to be the replication of the name to object lookup for processing the independent parts of the "logical" transaction. To alleviate this overhead, we introduce lookup hints as arguments returned when an object is looked up. That hint can then be optionally supplied as a subsequent transaction parameter to optimize access to the same object.

Some proposed details of lookup hints for the catalog object update operation:

- The hint itself is uninterpreted by the component which receives it; it is merely remembered and returned when the same object is to be referenced.

- When using a hint, only a full complete pathname is acceptable (i.e. including $ and no ellipses) for the file specifier.

- There are two somewhat distinct uses of hints which can be supported. In one case, the reference is to any version of the object in the catalog under the complete name. In this case, no further name qualifier is required. A second case is where only a particular version of a catalog object can satisfy the request (e.g. for reporting a new image of an existing version). To support this second case, there must additionally be some parameter by which the WM can validate that the object currently in the catalog under that name is the one to which the hint refers. Version numbers would appear to be suitable here, if they were part of our system. In their absence, we suggest using the object creation timestamp attribute (already maintained in the catalog) as the validating parameter. This would mean that whenever lookup hints are given out, the creation timestamp must be returned as an object attribute. When a lookup hint is used and intended to identify only a specific version of an object, the object specifier must include this attribute. If the lookup hint fails to refer to the object identified by the spec and any included attributes, the hint is ignored. If the catalog no longer contains an object with that name and those attributes (e.g. the version has been replaced), the operation fails. It is intended that the file-image-update operation include the creation (version) attribute, and that the operation fail if the object it is updating is not the one to which the hint applies. This is what would normally happen if the name and creation attribute did not match the current catalog entry under that name, even without the hint.

- The same hint and name specifiers would be usable for subsequent FBH to WM catalog updates to indicate that a cached image is to be deleted.

The primitive for initiating the transaction to add or remove cached file images from the catalog data base (used by "security kernel" processes) would be the following:

10.1.3 FILE-IMAGE-UPDATE

Accepts:

Identification: service id;

Exception
Handling: none

Logical: full file name specifier (only);

: attribute list (must include creation timestamp);

: lookup hint (optional, from a previous reference to the file object);

: type (add or remove a cached image PCD);

Physical: PCD to be added or removed;

Returns:

Confirmation: keep it, or discard it (default on error condition or timeout is to retain with the option of retrying a removal at some future time);

NOTES:

The File-Image-Update primitive is intended to support the coordination of independent file cache management by FBH hosts with the maintenance of a central catalog of physical file images. It is anticipated that the addition of images to the cache would be handled by the component coordinating the individual inter-host file transfer (e.g. FM or FP), whereas removal might be the responsibility of a background file space management daemon process which periodically purges unreferenced images.

These are the primitives that NSW host components must use to support read only file access, direct file access, system directed file caching, optimized NSW file copying for service support, and other file movement related aspects of Release 6.0 functionality.

System changes:

- Add these new primitives to WM.
- Write TBH/FE software to use the new primitives for interfacing to tool file requests/delivery, and FE file manipulation commands.
- Augment WM-GET, WM-DELIVER to include additional parameters needed for controlling file system modifications.

10.2 CHAINING TOOLS IN EXISTING WORKSPACE

We propose adding an additional lookup primitives of this time to support a form of tool chaining. The Access-to-Service-Object primitive would be a call on the WM to perform lookup and access control for an NSW service, in much the same way that Access-to-File-Object looked up NSW catalog files. Access-to-Service-Object would return a service descriptor, much like Access-to-File-Object returns a file descriptor. We envision that a Work space command interpreter would be able to use such a primitive to support a command for chained services in an existing workspace. At this time we anticipate this form of access to service to be able only for co-located services.

10.2.4 ACCESS-TO-SERVICE-OBJECT

Accepts:

Identification: session id; service id; or {user + password};

Exception Handling: help process;

: disambiguation control

: confirmation control

Logical: service spec (subject to general lookup rules);

: attribute list (implicit type = service; site = expected to be same as requestor);

: access type (execution);

: lookup hint

Returns:

Confirmation: success/failure

Logical: full NSW service name of object looked up;

: scope used (if any) to locate the object;

: lookup hint

Physical: physical service descriptor from catalog (similar to what gets transferred with BEGINTOOL TRANSACTION).

NOTES:

Service object lookup is performed in exactly the same manner as file object lookup. Once an appropriate object is identified, the accessing agent requires an execute key for the operation to succeed.

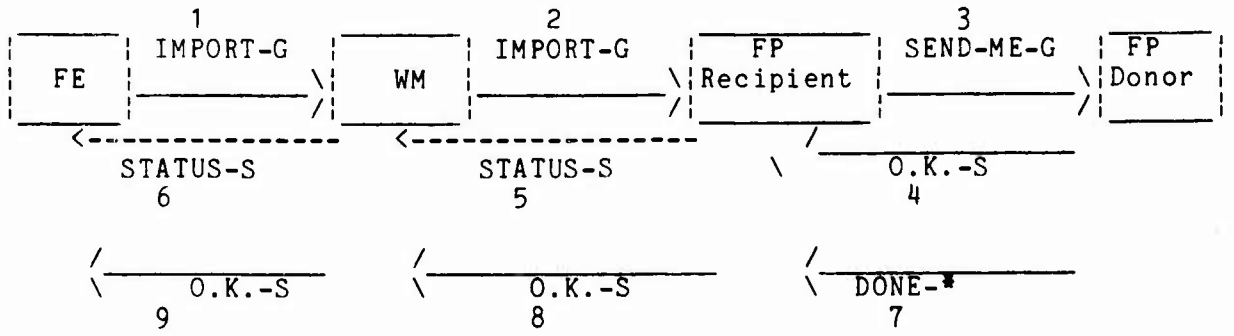
10.3 AVOIDING TIMEOUT ON LONG FILE TRANSFER

To avoid possible transaction timeout by initiating or intermediary processes due to an unexpectedly long event, such as transferring a large file under heavy load, we introduce the concept of a partial reply for long transactions. Transactions are designated as short, or long. Long transactions are those that may have a large variance beyond the expected variance of message transmission and simple processing. Currently, only interhost file transfer will be defined to be a long transaction, although others may be so designated as the need is identified. For long transactions, we break the replies into two phases: the first phase reply is an intermediate status response indicating only that all of the resources (hosts, processes, files, etc.) needed to complete the transaction are both currently available and committed to this transaction; the second phase reply is the normal termination of the transaction as currently supported. (See section 10.4 for a discussion of status querying during the second phase of a transaction). The responsibility for initiating the first phase status reply lies with the last process invoked in any designated long transaction chain. This process is identified in the documentation for

each relevant NSW scenario. Status replies are sent to the process designated in the invoking message as the one to which completion replies are to be sent. This is a process which is presumably timing out a final response, and is usually the invoking process.

It is the responsibility of all intermediary processes in a designated long transaction to relay (and possibly enhance) the status reply back to any process which may be awaiting its final response. (In some cases, a message serving as a status reply may already be part of the scenario). The issue in selecting and limiting those transactions which require intermediate status replies is one of avoiding the additional overhead in those cases where the status reply mechanism is of the same order of processing magnitude as the operation it is reporting. In general, an intermediate status reply should be acceptable (but not required) for any transaction using NSWTP conventions. For NSW version 6.0 we will specify and require its use in only the single case of file transfer to remedy the obvious system deficiency.

The immediate application of the status reply protocol will be in the case of NSW network file transfer. The transfer scenario involving network transmission would be upgraded to be:



Messages marked as "-G" are generically addressed. Messages marked as "-S" are specifically addressed. The message marked as "-*" is not currently an MSG message at all but rather an explicit data transmission over the direct connection between the FP processes to support the file transfer. It is a optimization of sorts, to incorporate existing signals which serve other functions as well (messages 4 and 7) into the general transaction protocol sequence.

By adding an intermediary status reply which cascades over all of the processes of the transaction, we accomplish two things (a) we provide

some relatively immediate feedback that the operation can be initiated and will proceed subject to loading factors on the hosts and the network (b) it provides initiating processes with the specific address of their generically addressed counterpart; this specific address can then be used to initiate further status queries, as discussed in then next section.

TIMEOUTS

Associated with the two phases of response for long transactions are two sets of timeout intervals. The short timeout interval (the one generally in use now) is intended to protect against waiting too long (tying up resources) before deciding that the operation is not likely to complete because of resource unavailability; the long timeout interval is a low overhead approach to protect against failure during an operation while allowing adequate time for operations to complete under variable load and size factors. A component would commit to the larger timeout because it has the prior knowledge that the operation has been successfully initiated, and the knowledge of the relevant operational parameters (e.g. size of file).

In general, a component in direct contact with the user can set very liberal timeouts provided the user is not locked out during the waiting period. If the user is prevented from any other parallel activity and from forcing the timeout to occur (i.e. aborting the operation) a shorter timeout interval, along with user help for renewing the timeout wait, seems advisable. A more complete solution to the timeout problem might include lower level guarantees of expedient delivery of timing signals. We will not pursue such an approach for NSW at this juncture. Rather, we will augment the intermediate response scheme with a general user oriented facility which allows one process to "poke" another process for its current status, given it has a specific address for the process. We will require that a long timeout be renewable if the communicating process responds to a status probe sent on expiration of the timeout period. This response will indicate that the component is still working on the request and that timeout may be premature.

NSW 6.0 will support the modified protocol which includes intermediate replies for all scenarios which have an NSW inter-host file transfer operation as a subscenario. This includes the IMPORT, EXPORT, GET, DELIVER, COPY. The FP processes are responsible for initiating the intermediate response at the minimum whenever a network file transfer is invoked. WM processes are responsible for relaying an intermediary response back to the original caller (FE or FM). In the case of direct FM/FE involvement with file transfer, there will not be an intermediary

WM, and the FM/FE will be the direct recipient of the intermediate reply. Processes receiving an intermediate reply must remember the process name associated with the reply (identifying the next process in the chain), and must attempt and be unsuccessful in a direct status probe (see next section) of that next process before completing a self-initiated transaction timeout. User interface software will be modified to confirm self-initiated transaction timeout.

System changes:

- o Augment FP's or all TBH's to send intermediate responses.
- o Modify WM to record intermediate response parameters and pass on to original caller.
- o Modify FE/FM/WM processes to initiate direct status probe whenever self-initiated timeout of transaction occurs, and the specific name of the process working on the transaction is known.

The form of the intermediate status responses is:

send specific ()

need new message type here???

10.4 DISTRIBUTED "ARE YOU THERE"

To allow FE users the ability to interrogate the status of long operations and on-going tool activity, we are specifying a status alarm to which all WM, FM and FP processes are expected to promptly reply with an indication of their current status. There are two forms of the status alarm. One form (direct status) requests the current status only of the receiving process. The other form (chained status) requests the status of an on-going transaction, and may involve processes relaying the status alarm to a process which it invoked as part of the transaction. A process receiving such a transaction alarm which is in the midst of a subtransaction with another process (e.g. a WM waiting for a file operation to complete) passes the status alarm down the transaction chain to its terminus and returns to the caller status derived from the process terminating the chain. This chain of events is similar to that for handling intermediate responses, except that it is invoked from the source instead of from the terminus.

We envision the user interface to such a facility allowing users to initiate "are you there" alarm signals directed at active service

sessions and active long transactions which have already reported intermediary status. This would be in addition to the local "are you there" (^T) now supported directly within the FE process. Users and NSW processes would be able to select either the direct or chained type of status invocation.

The reply to a status alarm serves two purposes:

(a) it serves to indicate that the appropriate processes are still active and (b) it provides a user text message indicating the current state of the transaction (to be defined later). Users may invoke "are you there" requests to verify that an on going long transaction is still progressing. System components may automatically invoke "are you there" signals (with a short response timeout) before initiating timeout procedures when a long timeout is about to expire.

Status replies which arrive after a final response to a transaction can be ignored. Status probes which refer to an unknown transaction return an indication of this condition.

Specifications for the form of the alarm and its response will be forthcoming.

NSW 6.0 will support a user initiated status probe against all active NSW service session and all active long transactions (i.e. those that have reported intermediate status). There will be FE commands/control sequences for initiating direct and chained status probes, and for reporting the results of the probe to the user.

System changes:

- Add FE commands/control sequences for status probe initiation (direct and chained), and facility to display results (similar to help message handling).

- Upgrade all NSW components (WM, FM, FP) to respond to a direct and chained status probe.

10.5 Other User Oriented Status Commands

NSW 6.0 will support commands for displaying tool and service object descriptors (or at least a subset of the data contained in those descriptors). The commands are SHOW FILE DESCRIPTORS, SHOW SERVICE DESCRIPTORS, and SHOW OBJECT DESCRIPTORS. Each takes a set of singular or plural object specifiers, and displays in text form the relevant

parameters of the descriptors (providing the user has appropriate access to the object in question). The SHOW NODE command will be augmented to display all appropriate node data, and a show session command will be added to display all pertinent data associated with current NSW user session data base. NSW 6.0 will also support new commands (SHOW USERS, SHOW HOSTS) for displaying a list of currently logged in users, and a list of hosts which are currently part of the NSW configuration at the time.

10.6 TBH/WM DATA BASE SYNCHRONIZATION

Currently, when a TBH restarts, the reliability scenarios call for it to report to the WM only new LND saved sessions (i.e. those not yet confirmed by the WM) which are rerunnable, and those recent sessions which the FM is discarding because they are not rerunnable. As currently formulated, the WM-LND-SAVED primitive will accept either a single tool-id or a list of tool-ids. In either case, the list is assumed to be incremental i.e. is in addition to whatever saved sessions are already in the data bases. This has led to situations where data base entries stored in one data base (e.g. WM) but not in the other (e.g. FM) required manual periodic deletion since they were not useable and were unreportable as deleted. In addition, hosts which did not save any sessions across TBH restarts had no simple way to automatically clean up invalid WM data base entries.

The modification we propose for NSW 6.0 is to add a saved session synchronization form of the WM-LND-SAVED call to allow a TBH to report ALL of the saved sessions it knows about (or is willing to save) at once, including those previously confirmed as rerunnable by the WM. This would allow the WM to purge its tables of all entries which the hosts have not indicated a willingness to save and are thus effectively useless. On completion of this transaction, the WM and TBH service session data bases should be synchronized. In particular, this form of the LND synchronization transaction will be required at TBH restart time. A host that does not support rerunnable sessions will report no saved sessions, and the WM will purge all current service activation records for that host.

There will now be two WM functions to support workspace reliability scenarios, WM-LND-SAVED and WM-LND-SYNCH. The WM-LND-SAVED message provides only for an incremental update to the WM's tables. Both the single-session and multiple-session variants of WM-LND-SAVED for continued use by an FM implementation. A new protocol message (WM-LND-SYNC) is provided for complete resynchronization of the FM and WM tool session data bases. The intent is that the WM-LND-SYNCH be

used at a minimum whenever the TBH restarts, and that the WM-LND-SAVE may be used during normal operation of the system to report isolated (e.g. single service session) updates without requiring a full resolution of the TBH database.

WM-LND-SAVE:

```
list:  ( string:  HERALD
         list   :  (integer:  TOOLID...)
         list   :  (integer:  TOOLID...)
       )
```

After the FM herald, the first list is of TOOLID's of tool sessions to preserve, the second list is of tool session to discard. Either list may be null. The most typical case called the single-session case, occurs when only one list is non-null, and contains only a single TOOLID. It is typically used to handle failures relating to a single tool instance (e.g. a broken FE connection) without incurring the overhead of a complete TBH data base scan.

<reply>:

```
list:  (integer:  TOOLID...)
```

The reply is a list of "exceptions." These sessions are unknown to the WM or have been previously discarded. Their entries should be discarded by the FM. Should an entry appear on both the "preserve" and "discard" lists of the WM-LND-SAVED message, the WM shall discard the session.

WM-LND-SYNC:

```
list:  ( string:  HERALD
         list   :  (integer:  TOOLID...)
       )
```

The list of TOOLID's specifies the entire set of recoverable tool sessions known to this TBH FM. By implication all other sessions associated with this TBH should be discarded. Up to about 500 sessions may be specified in a single MSG message under current size limitations. This primitive must be invoked by each TBH on every TBH restart.

<reply>:

list: (integer: TOOLID...)

The reply is a list of exceptions which the WM instructs the TBH to discard.

10.7 AUTOMATIC CLEANUP OF EXISTING SESSION RECORD ON RE-LOGIN ATTEMPT

This protocol modification is designed to support the ability to have a user request that an old inactive session activation record in the WM data base be purged so that a new login request can be processed. In addition this protocol change should support a way to transfer the service context from the old session to the new one. This will be done through the LND saving mechanism already implemented in the current system.

There follows a discussion of the changes to NSW 5.0 required to allow users to assert their right to clean up (in the SAVE-LND sense) a Node record which indicates a session in progress, and to effect a new Login to that Node.

User-Interface changes:

Instead of seeing an FE error-printout of the WM's text "Somebody is logged in to Node <proj>+<node>", the user will see a message more like:

NSWExec records indicate that <proj>+<node> is occupied with a session which started at <timestamp>. Type CR to save that session and continue your login, or ^X to abort the login.

Assuming the user types CR, his login will complete as usual; if there were in fact running Foremen with LNDs to save from the previous session, the user will see -- probably after some minutes -- messages announcing the RESUMEability of those saved sessions ("Records are available...").

Front-End changes:

The WM will send an Alarm to the FE, with a code indicating that it should do a Stopme, without sending any other messages or waiting for any replies. If the FE chooses not to accept this alarm, it will become disconnected from all its pending tools, and become

unknown to the WM (its Sessid (user id) will no longer retrieve a valid Active User Entry).

Data-Base changes:

+ Include an NSW-timestamp element in Node records and Active User entries. This allows the printout mentioned above, which helps the user recall whether the wedged session is actually one he had trouble with, or is someone else actively logged into the node. It also permits the normal LOGIN reply to give the time of last login, which is a small, but interesting security datum.

Protocol changes:

+ New FE Alarm command, mentioned above.

+ Foremen should be able to accept an FM-SAVE-LND call from Works Managers, as well as from FEs, and send their reply back to the sender, of course.

  - Foremen need not change their present practice of sending Generic WM-LND-SAVED messages during the SaveLnd scenario. The fact that some other instance of the WM will receive and process those messages is logically irrelevant, and will in fact simplify the modifications of the WM code.

  - As a future optimization, the Foremen might notice that the sender is a WM, and in that case, include the information they usually send in the Generic WM-LND-SAVED in their reply to the WM->FM:FM-SAVE-LND call, and skip the Generic WM call. We aren't ready to specify the form of that yet, since it would involve looking somewhat further into the WM code than we have done so far.

Works Manager changes:

+ At the outermost block level of the routine Wmlog, declare a flag to indicate peremptory overlogging, and space to contain the necessary information from the apparently-busy Node record and from the corresponding AUE, if available: At least Sessid, FE Process name, Semaphore list.

+ After the LOGIN parameters have been validated, and the user's Node record has been retrieved, the Sessid field (NODE>>node.ndsessid) is tested -- around line 139 of WMLOGS:Wmlog. The present code makes an error exit; it should be changed to set the overlogging flag, record the Sessid, and try to retrieve the AUE for that Sessid. If it can find the AUE, it should copy the essential information (or make a local copy of the whole AUE) and delete the old AUE from the Data base. Then it should return to the present code, setting up a new AUE.

+ Before writing the new AUE into the database, test the overlogging flag and, if set, append to the list of files with semaphores set (which has just been copied from the Node record into the new AUE) the names of any such files which were in the wedged AUE and are not already on the list. (Around line 177)

+ Return to the present code through sending the WM->FE:Reply (to the FE->WM:WM-LOGIN call). Then, before sending the WM->FE:FE-LND-SAVED calls for the old tools found in the Node record (around line 208), test the overlogging flag, and if it is set, generate a list of all ToolIDs which have the saved (wedged) Sessid as their user's id (using the Wmtess - Tfrrtv technique illustrated at lines 399-401 of WMLOGS:Wmflgo, for instance). Then, for each element of that list:

  - Fetch the Active Tool Entry for that ToolID to verify that it is (still) "active" and still points to the Sessid in question;

  - Construct a WM->FM:FM-SAVE-LND call to the Foreman process named in the ATE; close-and-unlock the ATE (so that the WM-LND-SAVED code can get at it); send the message, and wait for the reply. Not much can be done with the information in the reply, even if it is an error-reply.

  - Repeat the above steps for the next ToolID on the list, if any.

+ Then continue the remainder of the present code in Wmlog, sending WM->FE:FE-LND-SAVED messages for the older tools which had been in the Node record when it was originally fetched. (The normal processing of the FM->WM:WM-LND-SAVED messages will have caused WM->FE:FE-LND-SAVED messages for these newly-saved tools to be sent, already -- presumably these are of greatest interest to the user.)

+ Send the above-mentioned MSG Alarm to the FE Process named in the wedged AUE.

+ (Consider subroutinizing -- or replicating -- the appropriate portions of the code in Wmlnds to accept the FM->WM:WM-LND-SAVED information in the reply from the Foreman to the WM->FM:FM-SAVE-LND message, for future speeding-up.)

10.8 DETACH SERVICE SESSION

This protocol addition would allow a Foreman to request that a service connection be smoothly terminated, as if the session had ended but without the tool termination interaction with the WM which usually accompanies a message ending the conversation. The intent here is to allow a WS-CI on a TBH which provides rerunnable Service Sessions to support a SAVE command. When available, this command instructs the WS-CI to save the workspace context so that it may be resumed (reattached to) at some latter time. The current reliability scenarios support all phases of this except for the smooth cessation of the session without invalidating the session record. The FM can use the single tool instance of the LND save function to perpetuate the service session record. However, currently, the FM can not break the connection to the FE without invoking error recovery procedures (if it unilaterally breaks the connection) or invalidating the service entry with the WM (if it sends an End-tool message.)

The new FE function is:

<<to be supplied>>

10.9 WORKSPACE COMMAND INTERPRETERS and GENERAL SERVICE SUPPORT

NSW 6.0 will support Workspace Command Interpreters (WS-CI) to allow more direct use control over NSW service sessions, and as support for providing native node tool execution, tool kits, and detachable service sessions. When a WS-CI is provided by a TBH, it is activated in one of three ways:

- Through a designated well known name for the WS-CI entered in the NSW object catalog, and activated with the USE command.

- Through a designated NSW-wide control sequence issued while some other service is active.

- o Through the invocation of a tool kit (collection of tools to be run on a single host); whenever a tool kit is invoked, the user will find himself interacting with a WS-CI to coordinate the individual tool activations.

A WS-CI will use as its prompt the host name on which it runs. All WS-CI's will provide the following commands:

1. GET (NSW-file) as (Workspace file)
2. DELIVER (workspace file) as (NSW file)
3. RUN (program name)
4. RESUME (program name)
5. KILL (program name)
6. QUIT

In addition, WS-CI's may also support standard commands of the following form:

7. SHOW WORKSPACE-FILES ; to view the names of workspace files
8. SHOW TOOLS ; to view the names of tool kit members
9. SET file-access-mode
10. SAVE workspace-session ;to save the workspace context for later resumption

There may also be any number of host specific WS-CI commands, as well as interfaces to other NSW functions supported by the central NSW system.

Additional parameters which are to be added to the appropriate session setup messages to support more sophisticated services and user interfaces to services:

(a) an optional user-specified list of character strings collected by the FE and passed to the FM/SERVICE on startup (i.e. system can collect service parameters on command line; useful for non-TOPS-20 tools);

(b) full service name (character string);
scope (region) name was found in (character string);
user login name (character string) ;
user session id (integer);
NSW catalog name for workspace (character string
when supported);

all passed to FM as part of SERVICE startup to support the development of fancy tools and tool interfaces;

(c) Begintool handling a list of augmented tool descriptors to be processed by a WS-CI as possible arguments to a local RUN command;

(d) augment WM to FE Begintool response to support arbitrary cataloged ICP tools; the FE will do ICP to the host/socket returned form the Begintool.

(e) Host characteristics passed to FE on service session startup; (static through descriptor and/or dynamic through communication setup Augmented FE-Open Conn to support dynamic setting of host characteristics regarding communication with the particular host/service e.g. how to handle telnet control options, an alarm version of WS-CI "^C" if necessary, etc.

10.10 HELP MESSAGES ON THE DIRECT CONNECTION

To allow the use of direct connections for such things as user confirmation of service operations, while maintaining the ability to alert the user to needed help for a service instance he is not currently using.

addresses	number of copies	line number
Patricia Baskinger RADC/ISCP	10	
RADC/TSLD GRIFFISS AFB NY 13441	1	2
RADC/DAP GRIFFISS AFB NY 13441	2	3
ADMINISTRATOR DEF TECH INF CTR ATTN: DTIC-DDA CAMERON STA BG 5 ALEXANDRIA VA 22314	12	5
HQ ESC (XPZP) SAN ANTONIO TX 78243	1	12
HQ ESC/DOO SAN ANTONIO TX 78243	1	13
HQ USAF/XOKT WASHINGTON DC 20330	1	17
HQ USAF/RDST WASHINGTON DC 20330	1	20
HQ USAF/RDPV WASHINGTON DC 20330	1	21
PENTAGON USDR&E, RM 3D-139 ATTN: TSCO WASHINGTON DC 20301	2	26

HQ AFSC/TEVE ANDREWS AFB MD 20334	1	29
HQ AFSC/DLAE ANDREWS AFB MD 20334	1	30
HQ AFSC/DLWA ANDREWS AFB MD 20334	1	36
HQ AFSC/XRKK ANDREWS AFB MD 20334	1	37
HQ AFSC/XRKR ANDREWS AFB MD 20334	1	38
HQ SAC/NRI (STINFO LIBRARY) OFFUTT AFB NE 68113	1	39
AFATL/DLODL TECHNICAL LIBRARY EGLIN AFB FL 32542	1	44
HQ 3246 TW/TETW EGLIN AFB FL 32542	1	46
AFATL/DLODL EGLIN AFB FL 32542	1	47
ESMC/PM (STINFO) PATRICK AFB FL 32925	1	48

TAFIG/IIPE LANGLEY AFB VA 23665	1	50

HQ TAC/XPS (STINFO)] LANGLEY AFB VA 23665	1	51
HQ TAC/XPJC ATTN: Lt Taylor LANGLEY AFB VA 23665	1	52
HQ TAC/DRCA LANGLEY AFB VA 23665	1	55
HQ TAC/DRCG LANGLEY AFB VA 23665	1	56
HQ TAC/DRF LANGLEY AFB VA 23665	1	58
AFSC LIAISON OFFICE LANGLEY RESEARCH CENTER (NASA) LANGLEY AFB VA 23665	1	59
AFWL/NTYEE (C E BAUM) KIRTLAND AFB NM 87117	1	63
AFWL/SUL ATTN: TECHNICAL LIBRARY KIRTLAND AFB NM 87117	1	64
ASD/ENEGE ATTN: CAPT T CLELAND WRIGHT-PATTERSON AFB OH 45433	1	67
ASD/ENEGE ATTN: MR LARRY WEAVER WRIGHT-PATTERSON AFB OH 45433	1	68

ASD/TAMC WRIGHT-PATTERSON AFB OH 45433	1	74
ASD/XRE WRIGHT-PATTERSON AFB OH 45433	1	80
AFIT/LDE - TECHNICAL LIBRARY BUILDING 640, AREA B WRIGHT-PATTERSON AFB OH 45433	1	81
AFIT/ENE (MAJOR BORKY) WRIGHT-PATTERSON AFB OH 45433	1	82
ASD/AXT ATTN: CAPT CHARLES SMITH WRIGHT-PATTERSON AFB OH 45433	1	88
AFHRL/OTN WILLIAMS AFB AZ 85224	1	90
AFHRL/OA BROOKS AFB TX 78235	1	91
AUL/LSE 67-342 MAXWELL AFB AL 36112	1	96
HQ AFCC/DAPL BLDG P-40 NORTH, RM 9 SCOTT AFB IL 62225	1	98
HQ AFCC/EPE SCOTT AFB IL 62225	2	100
AFHRL/LRT LOWRY AFB CO 80230	1	101

3300 TTW/TTGX KEESLER AFB MS 39534	1	103
DEFENSE INTELLIGENCE AGENCY ATTN: RSE-2 (LT COL SCHWARTZ) WASHINGTON DC 20301	1	107
DEFENSE INTELLIGENCY AGENCY ATTN: RSM-1 WASHINGTON DC 20301	1	108
CODE R123 TECHNICAL LIBRARY DEFENSE COMMUNICATIONS ENGINEERING CENTER 1860 WIEHLE AVENUE RESTON VA 22090	1	110
DIRECTOR DEFENSE NUCLEAR AGENCY ATTN: TIL WASHINGTON DC 20305	1	111
CHIEF, C3 DIVISION DEVELOPMENT CENTER, MCDEC ATTN: R S HARTMAN QUANTICA VA 22134	2	112
AFLMC/LGY ATTN: MAJOR MORGAN GUNTER AFS AL 36114	1	116
DIRECTOR BML ADVANCED TECHNOLOGY CENTER ATTN: ATC-P, CHARLES VICK PO BOX 1500 HUNTSVILLE AL 35807	1	119
EOARD/CMI TECHNICAL LIBRARY FL 2878 BOX 14 FPO NY 09510	2	120
COMMANDING OFFICER NAVAL AVIONICS CENTER LIBRARY - CODE 765 INDIANAPOLIS IN 46218	1	123
NAVAL TRAINING EQUIPMENT CENTER TECHNICAL INFORMATION CENTER ORLANDO FL 32813	1	124

COMMANDER NAVAL OCEAN SYSTEMS CENTER ATTN: TECHNICAL LIBRARY, CODE 4473B SAN DIEGO CA 92152	1	125
SUPERINTENDENT (CODE 1424) NAVAL POSTGRADUATE SCHOOL MONTEREY CA 93940	1	127
COMMANDING OFFICER NAVAL RESEARCH LABORATORY CODE 2627 WASHINGTON DC 20375	1	128
REDSTONE SCIENTIFIC INFORMATION CENTER ATTN: DRSMI-RPRD US ARMY MISSILE COMMAND REDSTONE ARSENAL AL 35809	2	131
DOT/FAA TECHNICAL CENTER ARD-142 (ATTN: A R CIOFFI) ATLANTIC CITY NJ 08405	1	134
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH MESA LIBRARY PO BOX 3000 BOULDER CO 80307	1	135
AIR FORCE ELEMENT (AFELM) THE RAND CORP 1700 MAIN STREET SANTA MONICA CA 90406	1	140
DR RAYNER K ROSICH ELECTRO MAGNETIC APPLNS, INC C/O 7031 PIERSON STREET ARVADE CO 80004	1	142
AEDC LIBRARY (TECH FILES) ARNOLD AFS TN 37389	1	143
Director National Security Agency ATTN: T1213/TDL Fort Meade MD 20755	0	144
Director National Security Agency ATTN: W07 Fort Meade MD 20755	1	145

Director National Security Agency ATTN: W31 Fort Meade MD 20755	1	148
Director National Security Agency ATTN: S254 Fort Meade MD 20755	1	151
Director National Security Agency ATTN: R03 Fort Meade MD 20755	1	155
Director National Security Agency ATTN: R1 Fort Meade MD 20755	1	156
Director National Security ATTN: R2 Fort Meade MD 20755	1	157
Director National Security Agency ATTN: R5 Fort Meade MD 20755	1	158
Director National Security Agency ATTN: R6 Fort Meade MD 20755	1	159
Director National Security Agency ATTN: R7 Fort Meade MD 20755	1	160
Director National Security Agency ATTN: R8 Fort Meade MD 20755	1	161
Director National Security Agency ATTN: R9 Fort Meade MD 20755	1	162
HQ ESD/DCF-1 HANSCOM AFB MA 01731	1	163

HQ ESD/FAE, STOP 27 HANSCOM AFB MA 01731		1	164
ESD/DCKD (STOP 53) ATTN: LT COMBS HANSCOM AFB MA 01731		1	168
HQ ESD/YSEA HANSCOM AFB MA 01731		1	172
ESD/XRET HANSCOM AFB MA 01731		1	174
ESD/XRWS HANSCOM AFB MA 01731		1	178
ESD/XRWT HANSCOM AFB MA 01731		1	179
ESD/XRWW HANSCOM AFB MA 01731		1	180
ESD/XRTR HANSCOM AFB MA 01731		1	181
ESD/XR HANSCOM AFB MA 01731		1	184
ESD/DCR-3E HANSCOM AFB MA 01731		1	185
HQ ESD/YSM (STOP 18) HANSCOM AFB MA 01731	115	2	186

HQ ESD/DCR-1S Hanscom AFB MA 01731	1	187
HQ ESD/DCR-1I HANSCOM AFB MA 01731	1	188
Mr. Charlie Muntz Mass Computer Associates 26 Princess Street Wakefield MA 01880	5	3
AFLC/LOEC Attn: Capt Riski Wright-Patterson AFB OH 45433	1	4
WR-ALC/MMECV Attn: Mr. Palmer Craig Robins AFB GA 31098	1	5
SM-ALC/MMECF Attn: Mr.Van Jonnson McClellan AFB CA 95652	1	6
OC-ALC/MMECO Attn: Mr. Mike Parish Tinker AFB OK 73145	1	7
GSG, Inc. Attn: Mr. Doug Payne 51 Main Street Salem NH 03049	1	8
BBN, Inc. Attn: Dr. Rick Schantz 50 Moulton Street Cambridge MA 02138	2	9 10
Honeywell Information Systems 116 Attn: Mr. John Ata Bldg. 3, Room 35	1	11

Griffiss AFB NY 13441

UCLA
Attn: Mr. Neil Ludlam
Office of Academic Computing
Los Angeles CA 90024 — 2 — 12

SRI International
Attn: Ms. Jake Feinler
Menlo Park CA 94025 — 1 — 13

PAR
Attn: Mr. Thomas McGibbon
Freedom Mall
Rome NY 13440 — 1 — 14

UNIVAC
Attn: Mr. Paul Wood
PO Box 3525
Mail Station U2U22
St. PAul MN 55165 — 1 — 15

IIT Research Institute
Attn: Ms. Lorraine Duvall
The Beeches Carriage Suite
Turin Road
Rome NY 13440 — 1 — 16

Defense Advanced Research Proj Agency
Attn: Colonel Druffel
1400 Wilson Blvd
Arlington VA 22209 — 1

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.