DCE INTEROPERABILITY ASSURANCE PROGRAM — ILAB AND IFEST SERVICES

1. INTRODUCTION

DCE, as an interoperability technology, must do more than ensure conformance to APIs and protocols. It must also ensure the existence of practical distributed computing functionality. This ''real world'' functionality can be proven only through heterogeneous multi-vendor testing. For this reason, OSF has developed interoperability testing activities that will allow vendors to test their DCE implementations interactively with one another. These are known as ILAB (Interoperability Lab) and IFEST (Interoperability Festival).

2. ILAB

ILAB is an engineering environment continuously available to establish a close relationship between OSF engineering and DCE implementors.

For companies that would like additional assistance or the advantage of OSF-supported testing environment, the ILAB provides a multi-vendor heterogeneous client/server network environment, engineering support for porting and running tests provided by OSF, problem isolation and tracking, an interface to the OSF development organization, and the ability to work cooperatively with others to achieve product interoperability. As an example, the ILAB can be used to prepare DCE implementations for demonstration, IFEST, and/or Challenge style events.

This service is available year round and is designed to be used as an interactive engineering environment providing the opportunity for participating organizations to acquire a better understanding of the level of interoperability of their DCE implementations. Year round availability will allow participants to come into the Lab based on their own product schedules as often and whenever they want to test with the other DCE implementations in the lab.

Remote connections for this service will initially be provided to those companies whose physical requirements prevent them from being in the ILAB. Over time, these remote connections will be extended to all participants.

Initially, DCE implementations will be the main focus of testing in the Lab. Over time, the Lab is expected to be used to certify DCE implementations, provide an environment to develop and test distributed applications, increase the coverage of DCE implementation testing, and support the DME technology.

OSF offers an equipment loaner program for the ILAB enabling vendors to loan equipment to OSF after meeting the acceptance criteria for this program. This program provides the opportunity for this equipment to remain in the OSF facilities and be tested by other ILAB participants. In 1993, any fees associated with use of the ILAB or participation in a 1993 IFEST event are waived for any company lending a platform to OSF for the ILAB facility.

3. IFEST

IFEST is an engineering event at which vendors test their products interactively with one another and address interoperability issues cooperatively in real time.

IFEST is a large-scale, multi-vendor engineering workshop focused on facilitating open testing and development of DCE implementations with other participating companies to evaluate and demonstrate interoperability. IFEST is planned to be an annual event as of April, 1993.

IFEST will offer vendors an opportunity to tackle the large task of heterogenous interoperability testing in an open development environment. Vendors will allocate the required machine configurations as well as qualified engineering resources capable of addressing source-level issues and bugs encountered in multi-vendor testing. Interoperability issues will be addressed immediately at IFEST in real time. Typically these issues might otherwise be encountered initially at user sites, increasing the risks to DCE adoption.

The goal of IFEST is to foster an open, ''real-time'' testing and engineering environment to encourage the interoperability of as many DCE implementations with each other as possible. This event differs from ILAB in that it is an annual event, it makes available more implementations during the scheduled time period, and it is targeted towards testing and achieving interoperability between as many implementations as possible.

4. QUALIFICATIONS FOR PARTICIPATION

IFEST and ILAB will be open to all vendors developing DCE products, regardless of whether they are OSF DCE licensees. These events will also assist implementors in identifying implementation problems sooner than in the marketplace. ILAB and IFEST will be logical steps in the vendors' independent development and testing activities.

The general public will not be invited to IFEST and ILAB so that full vendor cooperation can be achieved. As IFEST and ILAB are not marketing events, vendors are expected to send engineers only. The information that is disclosed during these events is considered confidential by all participants, and is protected by the OSF ILAB/IFEST Confidentiality Agreement.

5. TESTING PROCEDURES

At the ILAB and IFEST, vendors will be expected to execute a number of tests designated by OSF in a specific order and in specific configurations of clients and servers (as is applicable based on the functionality provided by the implementation). The configurations and the test sequences are chosen to minimize the time spent doing required testing. Vendors should spend the remaining time testing with additional platforms, working with their own interoperability tests, and testing application software with other vendors. It is this phase of testing that will exercise DCE most rigorously.

For the 1993 IFEST event which took place in April, the primary objective was to test interoperability of the ''core'' DCE components on as many platforms as possible. Testing involved running four tests designated by OSF and provided as part of the DCE 1.0.1 release: `cds_abuse`, `dcesx`, `dcethrpc`, and `dceseacl`. Participants were required to run all tests as a DCE client and, if supported by the implementation, as a DCE server. In the case of the `dcethrpc` test, each configuration was also required to run the test both authenticated and unauthenticated. All testing was done within a specific cell, with the core component servers running on a single machine. Participants initially concentrated on running tests against the DCE 1.0 Reference Platforms (the Decstation 3100 running OSF/1 and the IBM RS6000 running AIX 3.2), and subsequently against other participants' test platforms. Some participants completed this testing early, and made an extra effort to examine other testing options; others used the entire three-week IFEST period to concentrate on the OSF-designated testing.

In addition to the original goal of core component testing, IFEST '93 included a preliminary effort by some companies to peform some interoperabilty testing of DFS. Future IFEST events are expected to expand this effort to a more formal and complete component of the testing event.

A detailed document explaining the current testing process and tests used is available from OSF Lab Services as part of the ILAB/IFEST Registration Kit. Interested parties should contact OSF Lab Services.

For details of the 1993 test process, see Appendix A.

6. ENTRANCE GUIDELINES

It is recommended that vendors be able to meet or exceed these guidelines before attending IFEST or ILAB:

(a) Provide a ''Secure Core'' implementation of DCE (RPC, Security, Directory, Threads, Time).

(b) Successfully execute the functional tests as provided in the DCE release on which their implementation is based.

(c) Successfully configure the platform to be tested into a one-machine cell (client and server, as applicable) using `dce_config` (or some similar vendor-provided tool).

7. EXIT CRITERIA

Vendors participating in ILAB will be expected to use their participation as a tool in acquiring a better understanding of the level of interoperability of their DCE implementations. There are no specific requirements regarding interoperability testing associated with ILAB participation.

Vendors participating in IFEST will be expected to successfully execute the interoperability tests designated by OSF with the OSF DCE Reference Platforms and at least two other implementations of the participant's choice. Tests will be performed on client and server implementations in multiple configurations of mixed-vendor services.

8. ILAB/IFEST FUTURES

OSF is continuing to evaluate and evolve the Interoperability programs as both the services and the technology on which they are based mature. Currently under consideration in terms of interoperability testing are:

(a) Improvement of coverage for currently tested components.

(b) Addition/extension of testing:
  (i) GDS testing.
  (ii) Enhanced DFS testing.
  (iii) Split-server testing.
  (iv) Inter-cell testing.
  (v) Replicated server testing.
  (vi) X.500 testing.
  (vii) DCE application testing.

(c) Connection of other lab sites.

(d) Planning for future events.

9. CONTACT INFORMATION

Organizations wishing to provide input to OSF or seeking information regarding the status of the Interoperability programs may do so by contacting:

OSF Lab Services
Phone: (617) 621-8869
Email: lab-services@osf.org

APPENDIX A. 1993 ILAB/IFEST TEST PROCESS

NOTE: This Appendix reproduces (with some re-editing) the document entitled ''OSF ILAB/IFEST Test Process'' that was used for the 1993 ILAB/IFEST. As noted in it, the testing process it describes is subject to change in subsequent years as the DCE technology and ILAB/IFEST itself gain in maturity.

This document presents the details of the test process to be used for participation in the OSF Interoperability Lab (ILAB) and Interoperability Festival (IFEST) programs. The following outlines the tests to be used, the manner in which each test is to be run, and the scenario in which all of the tests are to be executed. The information presented here is based on the current ILAB/IFEST offerings. It is expected that as DCE matures over time, the ILAB/IFEST and the associated testing process will mature to improve and expand the services provided. Information in this version of the document pertains to ILAB and IFEST activities in 1993.

(NOTE: Since the details of this test process may change with the maturation of these services and the technology they are designed to support, this document will always specify the period of time for which the information is applicable.)

A.1. ILAB/IFEST Test Descriptions

The following tests will be used for interoperability testing in the ILAB/IFEST facility. All tests are available on the DCE 1.0.1 release tape. Companies wishing to participate in ILAB or IFEST, but who are not OSF DCE licensees shall be provided with test source code with the stipulation that all copies of such code are returned or destroyed upon completion of their ILAB/IFEST participation. (Such companies should contact OSF Lab Services for details.) Any modifications or enhancements made to the tests by OSF subsequent to the 1.0.1 release, and deemed necessary for ILAB or IFEST will be made available to participants from OSF.

A.1.1. dcethrpc

(a) tests marshalling/unmarshalling of data representation
   (i) create threads, bind to server, pass data between client and server, compare received data for accuracy, log outcome, free resources

(b) tests server's ability to handle multiple different-platform clients and multiple simultaneous requests for service

(c) authenticated RPC via command line arguments

A.1.2. dceseacl

(a) test permissions via attempts to:
   (i) change ACLs on registry objects
   (ii) create principals, groups and organizations
   (iii) add/delete members of groups and organizations
   (iv) create accounts
   (v) change principal's, group's and organization's full name
   (vi) change management, authentication and user info for accounts
   (vii) change management info for principal, group and organization
   (viii) change authentication and management info for registry policies and procedures
   (ix) delete accounts, principals, groups and organizations

(b) verify invalid ACL entry types and permissions for all manager types

A.1.3. cds_abuse

(a) test CDS via various commands
   (i) create/list/show/delete directory
   (ii) create/show/delete object
   (iii) create/show/delete link
   (iv) set dir to skulk

(b) various scenarios re. order of commands (ex. create/show/ delete/show — verify delete worked)

(c) test uses `cdscp` (CDS control program)

A.1.4. dcesx

(a) test NSI control of threads via attempts to:
   (i) create/terminate/join multiple threads
   (ii) create/set/wait/delete mutex

(b) test NSI control of RPC via attempts to:
   (i) use connectionless and datagram protocols when possible
   (ii) register/unregister server bindings endpoints in the RPC runtime

(c) test NSI control of CDS via attempts to:
   (i) request/shutdown skulking
   (ii) export/remove server bindings to the directory service
   (iii) add/delete groups/profiles
   (iv) locate servers through group/profile entries

(d) test NSI control of security via attempts to:
   (i) set up a secure login

(e) test uses the following NSI APIs:
   (i) `rpc_ns_binding_export` — Establishes a name service database entry with binding handles or object UUIDs for a server.
   (ii) `rps_ns_profile_elt_add` — Adds an element to a profile; if necessary, creates the entry
   (iii) `rpc_ns_mgmt_set_exp_age` — Modifies the application's global expiration age for local copies of name service data

(iv) `rpc_ns_binding_unexport` — Removes the binding handles for an interface or object UUIDs from an entry in the name service database
(v) `rpc_ns_profile_elt_remove` — Removes an element from a profile
(vi) `rpc_ns_binding_import_begin` — Creates an import context for an interface and an object in the name service database
(vii) `rpc_ns_binding_import_next` — Returns a binding handle of a compatible server (if found) from the name service database
(viii) `rpc_ns_binding_import_done` — Deletes the import context for searching the name service database
(ix) `rpc_ns_group_mbr_add` — Adds an entry name to a group; if necessary, creates the entry
(x) `rpc_ns_group_mbr_remove` — Removes an entry name from a group

A.1.5. Other software needed for testing

(a) various control programs
  (i) `cdscp`
  (ii) `rpccp`
  (iii) `dtscp`
  (iv) `acl_edit`
  (v) `rgy_edit`
  (vi) `dce_config`

(b) systest tools
  (i) `chkproc`
  (ii) `expmachinfo`
  (iii) `getproc`
  (iv) `killproc`
  (v) `ropend` (server):
    [a] `ropen_idl.idl`
    [b] `remoteopen.c`
  (vi) `ropen` (client):
    [a] `ropen_idl.idl`
    [b] `ropen_main.c`
  (vii) `rshsp`
  (viii) `su.dce`
  (ix) *xx*`_begin` (where *xx* is a substitutable)
  (x) *xx*`_error`
  (xi) *xx*`_examine`
  (xii) *xx*`_fail`
  (xiii) *xx*`_log`
  (xiv) *xx*`_pass`
  (xv) *xx*`_severe`
  (xvi) *xx*`_warning`

A.2. Running the Tests

The following information is presented in two sections: general instructions which must be followed for running all tests, and specific instructions related to each of the individual tests.

A.2.1. General instructions for all tests

(a) Before running tests, DCE must be installed and configured.

(b) All required tests must exist locally on the machine to be tested.

(c) The default shell for ''root'' must be set to `/bin/sh`.

(d) There must exist a `/.rhosts` file on each machine to be used in testing, and the file must include the hostname of each machine to be used in testing.

(e) All tests and associated tools must be located in a ‘‘systest’’ directory which should be set up as follows:

```
                          systest
                             |
  +--------+------+-----+----+--+-----------+------------+
  |        |      |     |       |           |            |
profile.dcest  |  dcethrpc  |   directory  security      threads
           |            |       |           |            |
       dceseacl       tools     |      (dceseacl-   (dcethrpc-
                                |       related      related
                          +-----+-+---+  files)       files)
                          |       |   |
                     cds_abuse    |  log
                                dcesx
```

(f) STLIB must be set appropriately in *systest_dir*/`profile.dcest` based on the location of the systest directory.

(g) *systest_dir*/`profile.dcest` must be sourced from `/bin/sh` by the test operator in order to set up the proper environment.

A.2.2. Specific instructions for individual tests

A.2.2.1. dcesx

The options available for running dcesx are as follows:

(a) `-D` enable debug information
(b) `-K` perform skulks
(c) `-L` lock around imports
(d) `-M` enable multiple client threads
(e) `-Q` quiet mode
(f) `-R` re-randomize search order
(g) `-S` single system
(h) `-V` verbose
(i) `-H` help (display this information)
(j) `-i ifnum` specify # of interfaces
(k) `-m max_calls` specify # of call threads
(l) `-p passes` specify # of passes
(m) `-r retries` specify # of retries
(n) `-t delay` specify delay time

Before running the test, the operator must execute the following command:

```
dce_login cell_admin -dce-
```

The test should be run with the following command:

```
dcesx -K -M -R -i 10 -m 10 -p 1
```

There is no automatic logging mechanism for the `dcesx` executable. Output from `dcesx` is sent to `stderr`. As a result, *the operator must set up any logging desired*.

A.2.2.2. cds_abuse

The directory `./log` must be available within the directory from which the shell is being executed. (NOTE: Given the above description of the systest directory structure, it may be helpful to set up the test operator with systest as its home directory.)

The test may be executed by any login. However, the login id must have write rights to the `./log` directory.

Although the test automatically executes a `dce_login` to the `cell_admin` principal during certain sections of the test, the operator, if not root, should `dce_login` to `cell_admin` as a general precaution. This will ensure that certain other commands done by the test (such as klist) do not fail because of improper access/execution rights.

The test should be run with the following command:

```
cds_abuse 1
```

where the 1 indicates that the test should run for one hour.

Test logs are located in the ./log directory.

A.2.2.3. dceseacl

Remote machines and local machines must have an operator which has the systest directory as its home directory and includes the *systest_dir*/tools directory in its default path.

The options available for running dceseacl are as follows:

(a) −c Do not clean up the workstation log files (default: TRUE)
(b) −e Exclude testcases that are found on the command line
(c) −h show detailed usage for the workstation
(d) −l<LOOPS> Times to execute the workstation (default: 1)
(e) −t<HOURS> Hours to execute the workstation
(f) −f<File> name of file to use for environment variables

and

case_nums is a list of the testcases to execute or exclude (default: ALL)

The test should be run with the following command:

```
sh dceseacl  -l 1 -f dceseacl.data
```

NOTE: The default is to run using the file *systest_dir*/security/dceseacl.data as the data file.

Logs should be located in:

```
${STROOT}/current/{fail,pass,working,pass_fail_log}
```

NOTE: If test fails, some manual steps may be required to clean up the security database.

A.2.2.4. dcethrpc

Remote machines and local machines must have an operator which has the systest directory as its home directory and includes the *systest_dir*/tools directory in its default path.

The options available for running dcethrpc are as follows:

(a) −l<LOOPS> Times to execute the workstation (default: 1)
(b) −t<HOURS> Hours to execute the workstation
(c) −P<PROTOCOL> Protocol sequence (default: ncadg_ip_udp)
(d) −X<endpoint> Port number (default: 1062)
(e) −n<THREADS> Number of threads to create (default: 100)
(f) −m<MACHINE> Name of machine on which to start dcethrpc server (can specify multiple −m's)
(g) −s<CHUNK SIZE> Size of array chunks (default: 100)
(h) −C<MAX CALLS> Maximum number of calls server can handle concurrently

(i)	`-v`	Toggle verbose mode (default: TRUE)
(j)	`-p<principle name>`	User to authenticate with
(k)	`-A<auth level>`	Authentication level (default: 2)
(l)	`-V<auth service>`	Authentication service (default: 1)
(m)	`-Z<authz service>`	Authorization service (default: 1)
(n)	`-N`	Use unauthenticated RPC (overrides `-p`, `-A`, `-V`, and `-Z` flags)

and

`case_nums`	is a list of the testcases to execute or exclude (default: ALL)

There are two parts to the `dcethrpc` test: an RPC client (`thd_client`) which is referred to as the initiator, and an RPC server (`thd_server`) which is referred to as the receiver. In addition, the test may be run using authenticated RPC or unauthenticated RPC.

The tool `ropend` permits the start of a process from a remote machine. This is the reason for the `/.rhosts` file requirement described earlier. The receiving machine must exist in the `/.rhosts` file. In the case of `dcethrpc`, the `ropend` is necessary in order to permit the `thd_server` process to be run on the remote machine. The `thd_client` process then passes the appropriate commands into `thd_server`. `ropend` must be run on the receiver and should be run with the following command:

```
ropend &
```

The following combinations of `dcethrpc` must be run for two machines, A and B, between which testing is to be done:

(a) With machine A as the Initiator (I):
   (i) Run `dcethrpc` on machine A as RPC client using unauthenticated RPC (run `ropend` on machine B)
   (ii) Run `dcethrpc` on machine A as RPC client using authenticated RPC (run `ropend` on machine B)

(b) With machine B as the Initiator (I):
   (i) Run `dcethrpc` on machine B as RPC client using unauthenticated RPC (run `ropend` on machine A)
   (ii) Run `dcethrpc` on machine B as RPC client using authenticated RPC (run `ropend` on machine A)

For each run, it must be verified that both the `thd_client` and the `thd_server` have run successfully. (Review of associated log files should provide this information.)

To run `dcethrpc` using unauthenticated RPC, one of the following commands should be used:

```
sh dcethrpc -l 1 -m {appropriate_host} -N
```

or

```
sh dcethrpc -l 1 -m {appropriate_host} -N -X {port number}
```

if one wishes to specify the port to be used to contact the `thd_server`.

Running `dcethrpc` using authenticated RPC requires that a principal be added to the security database. In addition, the principal should have a password associated with it. *For the purpose of this test, the password should be the same as the principal's name.* To add the principal with password to the registry, the following should be done:

```
dce_login cell_admin -dce-
rgy_edit
domain principal
add {principal name}
domain account
add {principal name} -g none -o none -pw {principal password} \
                     -mp -dce-
quit
```

The principal must then be added to the `ktlist` on both the RPC client machine and the RPC server machine (each has its own ticket cache file):

```
ktadd -p {principal name} -pw {principal's password}
```

To run `dcethrpc` using authenticated RPC, one of the following commands should be used:

```
sh dcethrpc -l 1 -m {appropriate_host} -p {principal name}
```

or

```
sh dcethrpc -l 1 -m {appropriate_host} -p {principal name} \
            -X {port #}
```

if a specific port is to be assigned.

Logs should be located in:

```
${STROOT}/current/{fail,pass,working,pass_fail_log}
```

NOTE: If the test fails, some manual steps may be required to clean up the security database.

A.3. ILAB/IFEST Test Process

The following describes step by step the process by which interoperability testing in the ILAB or IFEST environment should take place. While there are no formal testing requirements associated with ILAB, success at IFEST for the purpose of attending OSF's Challenge event does require successful testing of an implementation against both the designated reference platforms and at least two other test platforms not provided by the participant organization.

The following four steps should be done *prior to* ILAB/IFEST participation:

(a) Acquire appropriate versions of tests to be used. Tests are available on the DCE 1.0.1 release tape which has been provided to all DCE 1.0 source licensees. Any modifications or enhancements made to the tests between the time DCE 1.0.1 is distributed and the time a participant comes to the lab will be made available to the participant. Companies wishing to participate in ILAB or IFEST, but who are not OSF DCE licensees shall be provided with test source code with the stipulation that all copies of such code are returned or destroyed upon completion of their ILAB/IFEST participation.

(b) Verify configuration of system in stand-alone environment. This should be done for both client and server as dictated by availability in the implementation to be tested.
  - (i) install DCE
  - (ii) configure DCE cell
    - [a] verify ability to use `dce_config` to create single-machine cell with both client and server (if applicable) for core components

(c) Verify ability of implementation to pass recommended DCE functional tests.
  - (i) Participants will be responsible for ensuring proper functionality of any implementation with which they wish to participate in the lab.
  - (ii) Such testing should be done *prior to* arrival at the ILAB/IFEST facility.
  - (iii) Recommended DCE functional tests include the following as provided in OSF DCE 1.0.1:
    - [a] `perf`
    - [b] `cp_killer`

[c] security functional tests

(iv) Test output should be compared with sample test output provided on DCE 1.0.1 tape for verification of passing.

(d) Tests acquired in the first step above should be *ported to and runnable on* the platform to be tested in a one-machine cell configuration *in advance* of coming to the ILAB/IFEST facility. This will facilitate the best use of time and resources in the lab environment.

(i) Preparation of the test platform to run required tests should include set up of the appropriate directories for tests, tools and log files

After completion of the above four steps, a participant should be prepared to begin interoperability testing of their DCE implementation within OSF's ILAB/IFEST environment.

For those participating in ILAB only, there are no specific testing requirements. Participants are free to run whatever interoperability tests they wish against whatever platforms are in the lab at the time of their participation. (However, the usual requirement of pre-registration to permit OSF to schedule such participation would still be enforced.) They may choose to run through the same testing scenario as will be required of IFEST participants, or they may choose to diverge from that scenario and run whatever tests best suit their needs at that time.

For those participating in IFEST, there is a set of exit criteria associated with successful IFEST participation. Those wishing to subsequently participate in OSF's Challenge event to demonstrate a DCE implementation must meet that set of criteria as a prerequisite of Challenge participation.

The following describes the testing scenario that will be used to facilitate participants' efforts to meet all IFEST exit criteria.

Configuration issues and testing requirements are as follows:

(a) Tests to be run include: `dcesx`, `cds_abuse`, `dceseacl` and `dcethrpc` (in four combinations of: (1) initiator, (2) recipient, (3) authenticated RPC, (4) unauthenticated RPC).

(b) All tests must be run against both OSF DCE 1.0.1 Reference Platforms (DECstation 3100 running OSF/1 1.0.4b8 (PMAX) and IBM RS6000 running AIX 3.2 (RS6000)), as well as at least two other test platforms of the participant's choice, other than their own.

(c) Testing must be done for client and/or server — whichever exists as part of the participant's implementation (i.e., client only, server only, or both client and server).

(d) Implementations and tests should be run using authentication (exception: `dcethrpc` will be run both authenticated and unauthenticated).

(e) At this time, OSF DCE currently supports both TCP and UDP protocols. Therefore, the ILAB testing environment will be configured to support these protocols.

(f) Testing configurations and scheduling are set up so as to minimize cell reconfiguration as much as possible as this can involve a significant time factor.

(g) Tests will be run simultaneously whenever possible.

(h) Split-server configurations will not be tested in this phase of the ILAB/IFEST offering.

(i) Intercell testing will not be done in this phase of the ILAB/IFEST offering.

Testing of all participants' test platforms (TPs) against the two reference platforms (RPs) will be addressed first. This phase of the testing will be referred to as ''Reference Platform Testing''. Following completion of this phase, testing of participants' TPs against other participants' TPs will be addressed. This phase of the testing will be referred to as ''Test Platform Testing''.

A.3.1. Reference platform testing

Every TP will test first as a client to each of the two RPs. This testing will be done in one large cell (two, if one becomes unmanageable) where all TP clients test simultaneously. Two scenarios (i.e., two different cell configurations) will be tested:

(a) Testing will begin with the RS6000 as the cell server. All TPs will run the required tests (`dcesx`, `cds_abuse`, `dceseacl` and the four variations of `dcethrpc`) configured as clients within this cell.

(b) The cell will then be reconfigured to use the PMAX as the cell server. All TPs will then run the required tests configured as clients within this cell configuration.

Upon completion of the ''TP as client'' testing, each TP will be tested as the cell server against the RPs as clients. This testing is expected to be done using multiple cell configurations which will make the most efficient use of all reference platform machines present in the lab as well as minimize the total time necessary for completing this phase of the testing. (Testing of TPs as servers against RPs as clients will require participants to ''wait their turn'' for short periods of time, and use of multiple cells will minimize the number of times this must occur.)

(a) Multiple cells will be configured, each with at least one of the two RPs included. The total number of TPs present will determine the number of TPs per cell (an even split will be desirable).

(b) For each cell including TP1 through TPn:

    TP1 will be configured as the cell server and each RP present in the cell will be configured as a client. All required tests will be run to ensure interoperability of the TP server with the RP client(s). Upon completion of these tests, the cell will be reconfigured such that TP2 will now be the cell server and the same testing will be run. This scenario will continue until TPn has completed its testing as a cell server to the RP client(s).

These two steps will be repeated as necessary to ensure that all TPs test as servers against both RPs as clients.

Upon completion of the ''TP as server'' testing, testing of TPs against one another will be addressed.

A.3.2. Test platform testing

TP testing will require that each TP has the opportunity to test as both a client and as a server against at least two other TPs. This means that every TP needs a chance to be configured as a server, and there will be between two and $n-1$ clients testing against it where n is the total number of TPs present (depending on the number of participants who wish to test their TP against the one that is currently a server).

Configurations for this testing phase will depend largely on the scheduled participants' requests for testing against certain platforms. Cell configuration will be done in such a way as to make the best use of all resources and time available while striving to meet all participants' testing objectives. Efforts will be made to maximize testing opportunities for all participants whenever possible. This is expected to result in one of two possible scenarios:

(a) One large cell is used where the cell is reconfigured n times to allow for each TP to act as the cell server and any/all client TPs in the cell may test against it. While this may take longer, it will maximize testing opportunities for all.

(b) Multiple cells are used where the various cells are comprised of TPs which participants have specifically requested be tested against each other. Each cell is then reconfigured x times (where x is the number of TPs in the cell) to allow for each TP to act as the cell server while client TPs test against it. In other words, if there is a group of participants who wish to test their TPs against each other, but have no interest in testing against the other TPs present in the lab, a separate cell may be configured to accommodate this. This may reduce the time necessary to complete all required testing (less time spent waiting for a particular TP to become server), but will also reduce testing opportunities.

A.3.3. Time required for IFEST testing

It is expected that testing will progress at the rate of two to three cell configurations per day, barring any problems which cause significant delays. Problems which would delay the testing schedule will be examined in relation to other cell members' progress. A determination will be made whether to hold up further testing to resolve problems at the end of that particular configuration testing.

The number of TPs present, as well as the number of cell configurations used to accomplish each phase of the required testing will directly effect the total time necessary for IFEST testing. For an environment of approximately 20 TPs, it is expected that the total time will be approximately three weeks.

A.3.4. Definition of success

All tests results should be maintained in appropriate logs, and successful testing (i.e., passage of each test) should be indicated by the contents of these logs. Participants will be expected to make their test results available to OSF. Success of each test will be verified by OSF Lab Services staff. Sample logs of successful tests run by OSF may be made available for participants' use in evaluating their results.

A.3.5. DTS testing

DTS NOTE: OSF is currently evaluating various methods of testing the interoperability of the DTS component of a DCE secure core implementation. Upon completion of this evaluation, an appropriate test will be provided to all ILAB/IFEST participants. It is anticipated that such a test will not be available in time for inclusion as a requirement of IFEST. It may, however, be made available for IFEST participants' use as a reference for DTS interoperability evaluation.

A.4. Future Considerations in the Test Process

The following are examples of accomplishments which would be pursued as part of subsequent phases of the ILAB offering:

(a) Enhanced test suites for core components
(b) Split-server testing
(c) Intercell testing
(d) DFS testing
(e) GDS testing
(f) Support for ISVs

OSF will continue to examine these and any other suggestions as they are made for ways to enhance and improve the ILAB offering over time.

AUTHOR'S ADDRESS

Maryanne Paratore
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
USA

Internet email: mp@osf.org
Telephone: +1-617-621-8957