

This is going straight to the pool room

This is going straight to the pool room

Source: <http://unix.derkeiler.com/Newsgroups/comp.os.vms/2007-08/msg02033.html>

- *From:* "Richard Maher" <maher_rj@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 28 Aug 2007 08:04:30 +0800
-

Hi Chuck,

If you're still there, FWIW here is how I'd do it (example code below).

Please search the attached code for `sys$sndjbc`. This code is called when the web-browser-based client has chosen to delete one of the VMS queue entries that was dynamically populated into their HTML DOM `<SELECT>` list. You might notice that the server code has not had to do *any* checking as to whether or not the client has privilege to do what they're trying to do, and is more than happy to allow VMS to sort that out. This is achieved by the deployment of the `T3$PERSONA_ASSUME` system service in the `USER_LOGON` routine. NB: The username that you have chosen to run these servers under requires *no* additional privileges to call `t3$persona_assume` and become the client for that unit of work!

The server environment is this: –

- 1) You have supplied the business logic for the application in the form of 6 3GL User action Routines that Tier3 will execute on your behalf during the life of the server.
- 2) You have selected a Username that you wish the Execution Servers to run under and registered the application in the Tier3 Configuration file. (If you wish your code to perform functions that the client does not have privileges for, then it is this Username that you would grant additional privileges, such as `CMKRNL`, to.)
- 3) You can specify a minimum number of servers that Tier3 must maintain for each application so that overhead, such as process-creation, image-activation, and database-attach, are out of the way before a client request arrives.
- 4) When Tier3 receives a client request, it allocates an available server from the System Manager configured pool, and will create additional servers if necessary.
- 5) Once an association between client and server is formed, it is up to your code to decide when the conversation, or message exchange, has completed. In the meantime you can call the `T3$SEND` service as often as you'd like and Tier3 will keep delivering you that client's request until it sees the `t3$m_close` or `t3$m_disconnect` flags. You can also attempt to interrupt the server with an OOB character if you wish to implement something like a hot-abort key or perhaps a Control-T.

This is going straight to the pool room

This is going straight to the pool room

The client environment is anything you want it to be! But for my Web-client examples (and the source is in the T3\$EXAMPLES directory) I have done this: –

- 1) Java Applet that gets loaded up to the client and whose init() method gets executed automatically when the web-page is displayed. (NB: *No* client installation of software reqd)
- 2) Java Sockets are used to connect back to the codebase (Your VMS server)
- 3) You *do not* need to be running Java, Tomcat, CGI, PHP or even a Web-Server on VMS *at all*
- 4) The Applet pops up a dialog box and gets your Username/Password and passes it to Tier3 for authentication.
- 5) If the credentials are invalid an "Access Denied" page is displayed
- 6) If successful, a network connection has been established that will automatically be torn-down when the browser changes pages or refreshes the current page. (This is optional and the Socket can be made to survive any number of page changes, but I haven't had the time to explore the possibility of Socket-Hijacking and with the use of HTML Frames I don't think it's necessary)

Anyway that is just one option for the client and you're free to choose whatever method you like. One reason why you might choose to follow my example is that 99% of the code is re-usable and will work for any Web application.

Cheers Richard Maher

```
*****
*****
*
*
* COPYRIGHT (c) BY TIER3 SOFTWARE LTD. ALL RIGHTS RESERVED.
*
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
ONLY *
* IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF SUCH LICENSE AND WITH
THE *
* THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY
OTHER *
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO
ANY *
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS
HEREBY *
* TRANSFERRED.
*
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
```

This is going straight to the pool room

This is going straight to the pool room

AND *

* SHOULD NOT BE CONSTRUED AS A COMMITMENT BY TIER3 SOFTWARE LTD.

*

*

*

*+

* Routine Name: USER_INIT

*

* This routine is called by Tier3 as part of the execution server's
initialization

* procedures.

*

* In this routine you would normally open files or connect to any
database(s)

* required by your application. In the DEMO application no files or
databases are

* accessed directly, but an exit handler is declared to listen for an
abnormal

* termination.

*

* As a CLI has been loaded you are also free to spawn a sub-process via
LIB\$SPAWN

* if that is a requirement of your application, and it should be noted that
there

* is no restriction on this execution server becoming a client of different
Tier3

* application(s) on other node(s).

*_

identification division.

program-id. user_init.

data division.

working-storage section.

01 ss\$_normal pic s9(9) comp value external

ss\$_normal.

01 sys_status pic s9(9) comp value external

ss\$_normal.

01 exit_desblk.

03 pic s9(9) comp.

03 pic s9(9) comp value external

exit_handler.

03 pic s9(9) comp value 1.

03 pointer value

reference exit_status.

01 exit_status pic s9(9) comp.

*

01 demo_context.

03 system_name pic x(8).

03 buffer_size pic 9(9) comp.

03 session_user pic x(12).

This is going straight to the pool room

This is going straight to the pool room

```
03 timer_context pic 9(9) comp.
03 session_count pic 9(9) comp.
03 message_count pic 9(9) comp.
03 lostlnk_count pic 9(9) comp.
03 persona_context pic 9(9) comp.
03 cancel_request pic 9(9) comp.
*+
* DEBUG area. TIER3 sets the "language" to COBOL by default, so the
following
* debugger commands are included just to show how.
*_
\d01 ss$_debug pic s9(9) comp value external
ss$_debug.
\d01 debug_commands.
\d 03 cmd_len pic x(1) value x"25".
\d 03 pic x(37) value "set
language cobol; show process/full".
*
linkage section.
01 t3_system_name pic x(8).
01 t3_buffer_size pic 9(9) comp.
procedure division using t3_system_name, t3_buffer_size giving sys_status.
00.
*+
* In this example, for the VMS debugger to be invoked at run-time then
* the /conditionals=d qualifier needs to be applied at compile-time.
*
* To assist a run-time decision as to whether or not to invoke debug
* TIER3 has defined a local DCL symbol T3$DEBUG. This symbol is set
* to the value specified by the system manager in the Tier3 Configuration
* file ie: "Y" or "N".
*_
\d call "lib$signal"
\d using by value ss$_debug, 1
\d by reference debug_commands.
*+
* Record some startup information in the execution server's log file.
*_
display "Initializing application " t3_system_name.
display "Buffer size is " t3_buffer_size with conversion.
*+
* These example routines are designed specifically to handle requests for
* the DEMO application only and require a buffer size of 510 bytes, but a
* single shareable image, or set of user action routines, could be designed
* to service the requests of multiple applications. Such routines could use
* the t3_system_name argument to determine the appropriate code path to
* execute at run-time. Similarly, by designing your user action routines to
* be able to cope with a variable buffer size you afford the system manager
* greater flexibility in tuning your application.
*
* We'll remember the Tier3 application parameters as they may come in handy.
```

This is going straight to the pool room

This is going straight to the pool room

*_

move t3_system_name to system_name.

move t3_buffer_size to buffer_size.

*+

* By passing the address of the data structure DEMO_CONTEXT as the context

* argument to the routine T3\$SETCTX, the data is made available to all

* subsequent user routines by becoming the first argument in each call.

*_

call "t3\$setctx" using demo_context giving sys_status.

if sys_status not = ss\$_normal go to fini.

*+

* DEMO does not need any application specific cleanup but we'll check the

exit

* status anyway.

*_

call "sys\$dclcxh" using exit_desblk giving sys_status.

*

fini.

exit program.

*

end program user_init.

*+

* Routine Name: USER_LOGON

*

* This routine is called by Tier3 when a client request has been allocated

to

* the execution server for processing. From this point on an association is

said

* to exist between the client and execution server. The association is

terminated

* by specifying either the t3\$m_close or the t3\$m_disconnect modifier in a

call to

* the T3\$SEND routine. The communication server may also cancel an

association at

* any time if the link to the client task is lost.

*

* Note: The output channel to the client will not be opened until the

USER_LOGON

* routine has returned control to Tier3. Therefore any attempt to call

* T3\$SEND from this routine would result in the error T3\$_CHANCLOSE

being

* returned.

*_

identification division.

program-id. user_logon.

data division.

working-storage section.

01 t3\$k_decnet pic s9(9) comp value external

t3\$k_decnet.

01 t3\$k_tcp_ip pic s9(9) comp value external

t3\$k_tcp_ip.

This is going straight to the pool room

This is going straight to the pool room

```
01 lib$_strtru pic s9(9) comp value external
lib$_strtru.
01 ss$_normal pic s9(9) comp value external
ss$_normal.
01 sys_status pic s9(9) comp value external
ss$_normal.
*
01 remote_node_name pic x(31).
01 remote_user_name pic x(12).
01 remote_user_socket redefines remote_user_name.
03 remote_host_addr.
05 rha_1 pic x.
05 rha_2 pic x.
05 rha_3 pic x.
05 rha_4 pic x.
03 remote_port_num pic 9(9) comp.
*
01 tcpip_rem_user_len pic 9(4) comp.
01 tcpip_rem_user pic x(21).
*
linkage section.
01 demo_context.
03 system_name pic x(8).
03 buffer_size pic 9(9) comp.
03 session_user pic x(12).
03 timer_context pic 9(9) comp.
03 session_count pic 9(9) comp.
03 message_count pic 9(9) comp.
03 lostlnk_count pic 9(9) comp.
03 persona_context pic 9(9) comp.
03 cancel_request pic 9(9) comp.
*
01 transport_type pic 9(4) comp.
*
01 remote_node_desc pic x(8).
*
01 remote_user_desc pic x(8).
*
01 local_user_desc.
03 lud_class_len.
05 lud_len pic 9(4) comp.
05 lud_class pic 9(4) comp.
03 lud_addr pic 9(9) comp.
*
01 local_persona pic 9(9) comp.
*
procedure division using demo_context,
transport_type,
remote_node_desc,
remote_user_desc,
local_user_desc,
```

This is going straight to the pool room

This is going straight to the pool room

```
local_persona giving sys_status.
00.
*+
* Initialize the timers and counts for session statistics.
*_
call "lib$init_timer" using timer_context giving sys_status.
if sys_status not = ss$_normal go to fini.

move zeros to cancel_request.
add 1 to session_count.
*+
* COBOL cannot receive arguments by descriptor transparently so
* the Run-Time Library routine LIB$SCOPY_DXDX is used to copy
* the remote_node, remote_user and local_user arguments to fixed
* length strings in local working-storage.
*_
call "lib$scopy_dx dx"
using by reference remote_node_desc
by descriptor remote_node_name
giving sys_status.
if sys_status not = ss$_normal and lib$_strtru go to fini.

call "lib$scopy_dx dx"
using by reference remote_user_desc
by descriptor remote_user_name
giving sys_status.
if sys_status not = ss$_normal and lib$_strtru go to fini.
*+
* The return status lib$_strtru is treated as a serious error for
* the local username. ie 12 bytes is the maximum length for a VMS
* username.
*_
call "lib$scopy_dx dx"
using by reference local_user_desc
by descriptor session_user
giving sys_status.
if sys_status not = ss$_normal go to fini.
*+
* Report the client association in the execution server's log file.
*_
display "Association with ", session_user(1:lud_len), " established".
display "Transport = " transport_type with conversion.
display "Rem node = " remote_node_name.
display "Rem user = " no advancing.

evaluate transport_type
when t3$k_decnet display remote_user_name
when t3$k_tcp_ip call "sys$fao"
using by descriptor
"!@UB.!@UB.!@UB.!@UB:!ZL"
by reference
```

This is going straight to the pool room

This is going straight to the pool room

```
tcpip_rem_user_len
by descriptor
tcpip_rem_user
by reference rha_1,
rha_2, rha_3, rha_4
by value
remote_port_num
giving sys_status
if sys_status not = ss$_normal go to fini
end-if
display tcpip_rem_user(1:tcpip_rem_user_len)
end-evaluate.
*+
* By assuming the persona of the VMS username that the remote client has
* been authorized to use on local node we have removed the need to perform
* additional security checking such as $check_access. IE: This server will
* be running with only those privileges and access rights that the client
* is entitled to assume. (Let VMS do all the work :-)
*
* Normally you would require DETACH privilege in order to be able to create
* a persona for the requesting client. But, as TIER3 has already authorised
* client access, we can provide you with the T3$PERSONA_ASSUME service so
* that a completely unprivileged server username is able to assume the
* guise of the requesting client.
*
* NB: Again, this is only a suggestion and not a requirement of Tier3.
*
* We'll hang on to the Persona Id in case someone like Rdb wants it.
*_
move local_persona to persona_context.
call "t3$persona_assume" giving sys_status.
*
fini.
exit program.
*
end program user_logon.
*+
* Routine: USER_RECV
*
* The USER_RECV routine is called by Tier3 for each message, or
* fragment of a message, that is received from a client with which
* the execution server is associated.
*
* In the DEMO example we are receiving one message per association
* with no fragmentation.
*
* If the status value T3$_CHANCLOSE is returned from the T3$SEND
* routine it is assumed that the link to the client has been lost
* so we simply return control to Tier3 after resetting our exist
* status to SS$_NORMAL to avoid server crash.
*_
```

This is going straight to the pool room

This is going straight to the pool room

identification division.
program-id. user_recv.
data division.
working-storage section.
01 t3\$_chanclose pic s9(9) comp value external
t3\$_chanclose.
01 sjc\$_delete_job pic s9(9) comp value external
sjc\$_delete_job.
01 jbc\$_nosuchque pic s9(9) comp value external
jbc\$_nosuchque.
01 jbc\$_nosuchjob pic s9(9) comp value external
jbc\$_nosuchjob.
01 jbc\$_nosuchent pic s9(9) comp value external
jbc\$_nosuchent.
01 jbc\$_nomoreque pic s9(9) comp value external
jbc\$_nomoreque.
01 jbc\$_nomorejob pic s9(9) comp value external
jbc\$_nomorejob.
01 jbc\$_nomoreent pic s9(9) comp value external
jbc\$_nomoreent.
01 jbc\$_normal pic s9(9) comp value external
jbc\$_normal.
01 ss\$_bufferovf pic s9(9) comp value external
ss\$_bufferovf.
01 ss\$_abort pic s9(9) comp value external
ss\$_abort.
01 ss\$_normal pic s9(9) comp value external
ss\$_normal.
01 sys_status pic s9(9) comp.
*
01 now_close pic 9(9) comp value external
now_close.
01 t3\$m_close pic 9(9) comp value external
t3\$m_close.
01 t3\$m_disconnect pic 9(9) comp value external
t3\$m_disconnect.
01 t3\$m_now pic 9(9) comp value external
t3\$m_now.
01 t3\$m_more pic 9(9) comp value external
t3\$m_more.
01 t3\$m_oob pic 9(9) comp value external
t3\$m_oob.
01 qui\$_display_queue pic s9(9) comp value external
qui\$_display_queue.
01 qui\$_display_entry pic s9(9) comp value external
qui\$_display_entry.
01 qui\$_display_job pic s9(9) comp value external
qui\$_display_job.
01 qui\$_cancel_operation pic s9(9) comp value external
qui\$_cancel_operation.
01 qui\$m_search_wildcard pic s9(9) comp value external

This is going straight to the pool room

This is going straight to the pool room

```
qui$m_search_wildcard.  
01 search_all_jobs pic s9(9) comp value external  
search_all_jobs.  
*  
01 getmsg_flags pic 9(9) comp value 3.  
01 send_flags pic s9(9) comp.  
01 local_flags pic s9(9) comp.  
01 sts_index pic s9(9) comp.  
01 on_or_off pic s9(9) comp.  
01 entry_count pic 9(9) comp.  
01 entry_limit pic 9(9) comp.  
01 qui_iosb.  
03 qui_cond pic s9(9) comp.  
03 pic s9(9) comp.  
01 sjc_iosb.  
03 sjc_cond pic s9(9) comp.  
03 pic s9(9) comp.  
*  
01 que_status_table.  
03 pic s9(9) comp value external  
qui$m_queue_available.  
03 pic x(10) value  
"AVAILABLE".  
03 pic s9(9) comp value external  
qui$m_queue_busy.  
03 pic x(10) value "BUSY".  
03 pic s9(9) comp value external  
qui$m_queue_disabled.  
03 pic x(10) value  
"DISABLED".  
03 pic s9(9) comp value external  
qui$m_queue_idle.  
03 pic x(10) value "IDLE".  
03 pic s9(9) comp value external  
qui$m_queue_paused.  
03 pic x(10) value  
"PAUSED".  
03 pic s9(9) comp value external  
qui$m_queue_pausing.  
03 pic x(10) value  
"PAUSING".  
03 pic s9(9) comp value external  
qui$m_queue_resuming.  
03 pic x(10) value  
"RESUMING".  
03 pic s9(9) comp value external  
qui$m_queue_stalled.  
03 pic x(10) value  
"STALLED".  
03 pic s9(9) comp value external  
qui$m_queue_starting.
```

This is going straight to the pool room

This is going straight to the pool room

```
03 pic x(10) value
"STARTING".
03 pic s9(9) comp value external
qui$m_queue_stopped.
03 pic x(10) value
"STOPPED".
03 pic s9(9) comp value external
qui$m_queue_stopping.
03 pic x(10) value
"STOPPING".
01 que_status_array redefines que_status_table.
03 que_status_item occurs 11.
05 que_value pic s9(9) comp.
05 que_text pic x(10).
01 que_flags_table.
03 pic s9(9) comp value external
qui$m_queue_printer.
03 pic x(10) value
"PRINTER".
03 pic s9(9) comp value external
qui$m_queue_batch.
03 pic x(10) value "BATCH".
03 pic s9(9) comp value external
qui$m_queue_generic.
03 pic x(10) value
"GENERIC".
03 pic s9(9) comp value external
qui$m_queue_terminal.
03 pic x(10) value
"TERMINAL".
01 que_flags_array redefines que_flags_table.
03 que_flags_item occurs 4.
05 flags_value pic s9(9) comp.
05 flags_text pic x(10).
01 job_status_table.
03 pic s9(9) comp value external
qui$m_job_aborting.
03 pic x(15) value
"ABORTING".
03 pic s9(9) comp value external
qui$m_job_executing.
03 pic x(15) value
"EXECUTING".
03 pic s9(9) comp value external
qui$m_job_holding.
03 pic x(15) value
"HOLDING".
03 pic s9(9) comp value external
qui$m_job_inaccessible.
03 pic x(15) value
"INACCESSIBLE".
```

This is going straight to the pool room

This is going straight to the pool room

```
03 pic s9(9) comp value external
qui$m_job_pending.
03 pic x(15) value
"PENDING".
03 pic s9(9) comp value external
qui$m_job_refused.
03 pic x(15) value
"REFUSED".
03 pic s9(9) comp value external
qui$m_job_retained.
03 pic x(15) value
"RETAINED".
03 pic s9(9) comp value external
qui$m_job_stalled.
03 pic x(15) value
"STALLED".
03 pic s9(9) comp value external
qui$m_job_starting.
03 pic x(15) value
"STARTING".
03 pic s9(9) comp value external
qui$m_job_suspended.
03 pic x(15) value
"SUSPENDED".
03 pic s9(9) comp value external
qui$m_job_timed_release.
03 pic x(15) value "TIMED
RELEASE".
01 job_status_array redefines job_status_table.
03 job_status_item occurs 11.
05 job_value pic s9(9) comp.
05 job_text pic x(15).
*
01 item_list_addr pointer.
01 qui_function pic s9(9) comp.
*
01 entry_item_list.
03 item_search_entry.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_search_number.
05 pointer value reference search_number.
05 pic s9(9) comp.
03 item_queue_flags.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_queue_flags.
05 pointer value reference que_flags.
05 pic s9(9) comp.
03 item_que_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_queue_status.
05 pointer value reference que_status.
```

This is going straight to the pool room

This is going straight to the pool room

05 pic s9(9) comp.
03 item_job_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_job_status.
05 pointer value reference job_status.
05 pic s9(9) comp.
03 item_que_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_queue_name.
05 pointer value reference que_name.
05 pointer value reference que_name_len.
03 item_job_name.
05 pic s9(4) comp value 39.
05 pic s9(4) comp value external qui\$_job_name.
05 pointer value reference job_name.
05 pointer value reference job_name_len.
03 item_job_username.
05 pic s9(4) comp value 12.
05 pic s9(4) comp value external qui\$_username.
05 pointer value reference job_username.
05 pointer value reference job_username_len.
03 item_account_name.
05 pic s9(4) comp value 8.
05 pic s9(4) comp value external qui\$_account_name.
05 pointer value reference account_name.
05 pointer value reference account_name_len.
03 item_uic.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_uic.
05 pointer value reference job_uic.
05 pic s9(9) comp.
03 item_priority.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_priority.
05 pointer value reference que_priority.
05 pic s9(9) comp.
03 item_job_size.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_job_size.
05 pointer value reference job_size.
05 pic s9(9) comp.
03 item_pid.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_job_pid.
05 pointer value reference job_pid.
05 pic s9(9) comp.
03 item_job_copies.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_job_copies.
05 pointer value reference job_copies.
05 pic s9(9) comp.

This is going straight to the pool room

This is going straight to the pool room

```
03 item_log_queue.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui$_log_queue.
05 pointer value reference log_que_name.
05 pointer value reference log_que_name_len.
03 item_sub_time.
05 pic s9(4) comp value 8.
05 pic s9(4) comp value external qui$_submission_time.
05 pointer value reference sub_time.
05 pic s9(9) comp.
03 pic s9(9) comp.
*
01 wild_item_list.
03 item_search_flags.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_search_flags.
05 pointer value reference qui$m_search_wildcard.
05 pic s9(9) comp.
03 item_queue_flags.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_queue_flags.
05 pointer value reference que_flags.
05 pic s9(9) comp.
03 item_job_entry.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_entry_number.
05 pointer value reference job_entry.
05 pic s9(9) comp.
03 item_que_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_queue_status.
05 pointer value reference que_status.
05 pic s9(9) comp.
03 item_job_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui$_job_status.
05 pointer value reference job_status.
05 pic s9(9) comp.
03 item_que_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui$_queue_name.
05 pointer value reference que_name.
05 pointer value reference que_name_len.
03 item_job_name.
05 pic s9(4) comp value 39.
05 pic s9(4) comp value external qui$_job_name.
05 pointer value reference job_name.
05 pointer value reference job_name_len.
03 pic s9(9) comp.
*
```

```
01 job_item_list.
```

This is going straight to the pool room

This is going straight to the pool room

03 item_search_flags.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_search_flags.
05 pointer value reference search_all_jobs.
05 pic s9(9) comp.
03 item_job_entry.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_entry_number.
05 pointer value reference job_entry.
05 pic s9(9) comp.
03 item_job_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_job_status.
05 pointer value reference job_status.
05 pic s9(9) comp.
03 item_que_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_queue_name.
05 pointer value reference que_name.
05 pointer value reference que_name_len.
03 item_job_name.
05 pic s9(4) comp value 39.
05 pic s9(4) comp value external qui\$_job_name.
05 pointer value reference job_name.
05 pointer value reference job_name_len.
03 pic s9(9) comp.
*
01 que_details_list.
03 item_search_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_search_name.
05 pointer value reference search_name.
05 pic s9(9) comp.
03 item_que_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_queue_name.
05 pointer value reference que_name.
05 pointer value reference que_name_len.
03 item_que_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_queue_status.
05 pointer value reference que_status.
05 pic s9(9) comp.
03 item_que_priority.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_base_priority.
05 pointer value reference base_priority.
05 pic s9(9) comp.
03 item_form_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_form_name.

This is going straight to the pool room

This is going straight to the pool room

05 pointer value reference form_name.
05 pointer value reference form_name_len.
03 item_stock_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_form_stock.
05 pointer value reference stock_name.
05 pointer value reference stock_name_len.
03 item_scsnode_name.
05 pic s9(4) comp value 6.
05 pic s9(4) comp value external qui\$_scsnode_name.
05 pointer value reference scsnode_name.
05 pointer value reference scsnode_name_len.
03 pic s9(9) comp.
*
01 que_summary_list.
03 item_search_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_search_name.
05 pointer value reference search_name.
05 pic s9(9) comp.
03 item_search_flags.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_search_flags.
05 pointer value reference qui\$m_search_wildcard.
05 pic s9(9) comp.
03 item_que_name.
05 pic s9(4) comp value 31.
05 pic s9(4) comp value external qui\$_queue_name.
05 pointer value reference que_name.
05 pointer value reference que_name_len.
03 item_queue_flags.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_queue_flags.
05 pointer value reference que_flags.
05 pic s9(9) comp.
03 item_que_status.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external qui\$_queue_status.
05 pointer value reference que_status.
05 pic s9(9) comp.
03 pic s9(9) comp.
*
01 sjc_item_list.
03 item_search_entry.
05 pic s9(4) comp value 4.
05 pic s9(4) comp value external sjc\$_entry_number.
05 pointer value reference search_number.
05 pic s9(9) comp.
03 pic s9(9) comp.
*
01 search_number pic s9(9) comp.

This is going straight to the pool room

This is going straight to the pool room

01 search_name pic x(31).
01 que_name pic x(31).
01 que_name_len pic 9(4) comp.
01 que_status pic s9(9) comp.
01 que_flags pic s9(9) comp.
01 base_priority pic s9(9) comp.
01 job_name pic x(39).
01 job_name_len pic 9(4) comp.
01 job_status pic s9(9) comp.
01 job_entry pic s9(9) comp.
01 account_name pic x(8).
01 account_name_len pic 9(4) comp.
01 job_uic pic s9(9) comp.
01 que_priority pic 9(9) comp.
01 job_size pic 9(9) comp.
01 job_pid pic 9(9) comp.
01 job_copies pic 9(9) comp.
01 log_que_name pic x(31).
01 log_que_name_len pic 9(4) comp.
01 sub_time pic s9(11)v9(7) comp.
01 job_username pic x(12).
01 job_username_len pic 9(4) comp.
01 form_name pic x(31).
01 form_name_len pic 9(4) comp.
01 stock_name pic x(31).
01 stock_name_len pic 9(4) comp.
01 scsnode_name pic x(6).
01 scsnode_name_len pic 9(4) comp.
*
01 show_entry_reply.
03 se_rec_type pic x(2).
88 entry_summary_resp value "11".
88 entry_detail_resp value "21".
03 se_summary_info.
05 se_entry_number pic z(9)9.
05 se_job_name pic x(39).
05 se_job_status pic x(15).
05 se_que_name pic x(31).
05 se_que_type pic x(10).
05 se_que_status pic x(10).
03 se_detail_info.
05 se_username pic x(12).
05 se_account_name pic x(8).
05 se_uic pic x(9).
05 se_priority pic z(3).
05 se_copies pic z(3).
05 se_size pic z(10).
05 se_pid pic x(8).
05 se_log_queue pic x(31).
05 se_submission_time pic x(23).
*

This is going straight to the pool room

This is going straight to the pool room

```
01 show_entry_len pic 9(9) comp value 117.
01 get_details_len pic 9(9) comp value 224.
01 entry_msg_len pic 9(9) comp.
*
01 show_que_reply.
03 sq_rec_type pic x(2).
88 que_summary_resp value "41".
88 que_detail_resp value "51".
03 sq_que_name pic x(31).
03 sq_que_status pic x(10).
03 sq_scsnode_name pic x(6).
03 sq_form_name pic x(31).
03 sq_stock_name pic x(31).
03 sq_base_priority pic z9.
*
01 show_que_len pic 9(9) comp value 33.
01 que_details_len pic 9(9) comp value 113.
*
01 error_msg.
03 pic x(2) value "00".
03 error_len pic 9(3).
03 error_text pic x(505).
01 error_msg_len pic 9(4) comp.
*
01 eof_msg.
03 pic x(2) value "99".
03 closure_type pic x(3).
*+
* The following field is used to induce a delay in reselect-delivery
* back to the client, to such an extent that the "CANCEL" button in the
* HTML/JavaScript code has a chance to interject. Set it to something
* like 1.50, if you're debugging the USER_INT routine or "Working. . ."
* message functionality.
*_
01 debug_delay comp-1 value 1.75.
*
linkage section.
01 demo_context.
03 system_name pic x(8).
03 buffer_size pic 9(9) comp.
03 session_user pic x(12).
03 timer_context pic 9(9) comp.
03 session_count pic 9(9) comp.
03 message_count pic 9(9) comp.
03 lostlnk_count pic 9(9) comp.
03 persona_context pic 9(9) comp.
03 cancel_request pic 9(9) comp.
*
01 msg_buff.
03 msg_type pic xx.
88 show_entry value "10".
```

This is going straight to the pool room

This is going straight to the pool room

```
88 get_details value "20".
88 delete_entry value "30".
88 show_que value "40".
88 que_details value "50".
88 show_jobs value "60".
03 pic x(508).
01 show_entry_msg redefines
msg_buff.
03 pic xx.
03 show_entry_number pic z(9)9.
03 show_entry_max pic 9(5).
01 get_details_msg redefines
msg_buff.
03 pic xx.
03 details_number pic z(9)9.
01 del_entry_msg redefines
msg_buff.
03 pic xx.
03 delete_number pic z(9)9.
01 que_info_msg redefines
msg_buff.
03 pic xx.
03 que_info_name pic x(31).
*
01 msg_size pic s9(9) comp.
*
01 msg_flags pic s9(9) comp.
*
procedure division
using demo_context,
msg_buff,
msg_size,
msg_flags
giving sys_status.
kick_off section.
00.
add 1 to message_count.
*+
* If the client has sent a message that is greater than the buffer size
* for this application, then Tier3 will set the bit T3$V_MORE in the
* msg_flags argument. The remaining fragments of the message are not lost,
* and will be delivered to user_recv routine in subsequent invocations.
* This example does not contain all of the code necessary to properly
* handle fragmented messages. The following "IF" statement is included
* merely to illustrate how your routine would be notified of fragmentation.
*_
call "mth$jiand" using msg_flags, t3$m_more giving local_flags.
if local_flags not = zeros display "Fragmented message – More to come".
*+
* NOTE: All Tier3 parameters passed to your routines are maintained in
* user mode read-only storage. Any attempt to modify their contents will
```

This is going straight to the pool room

This is going straight to the pool room

- * result in an access violation. Take a copy of any input parameter that
- * needs to be modified to local working-storage.
- *_

move low-values to error_text.
move spaces to show_entry_reply,
show_que_reply.

```
evaluate true
when show_entry if msg_size not = function length
(show_entry_msg)
or show_entry_number is not numeric
or show_entry_max is not numeric
perform break_link
else
set entry_summary_resp to true
move t3$m_now to send_flags
move show_entry_len to entry_msg_len
move show_entry_max to entry_limit
if show_entry_number = zeros
perform get_entry_wild
else
perform get_entry_info
end-if
end-if
when get_details if msg_size not = function length
(get_details_msg)
or details_number is not numeric
perform break_link
else
set entry_detail_resp to true
move now_close to send_flags
move get_details_len to
entry_msg_len
perform get_entry_info
end-if
when delete_entry if msg_size not = function length
(del_entry_msg)
or delete_number is not numeric
perform break_link
else
perform delete_it
end-if
when show_que if msg_size not = function length
(que_info_msg)
perform break_link
else
set que_summary_resp to true
perform get_que_names
end-if
when que_details if msg_size not = function length
(que_info_msg)
```

This is going straight to the pool room

This is going straight to the pool room

```
perform break_link
else
set que_detail_resp to true
perform get_que_details
end-if
when show_jobs if msg_size not = function length
(que_info_msg)
perform break_link
else
set entry_summary_resp to true
move show_entry_len to entry_msg_len
move t3$m_now to send_flags
move zeros to entry_limit
perform get_que_jobs
end-if
when other perform break_link
end-evaluate.

if sys_status = t3$_chanclose move ss$_normal to sys_status.

exit program.
*
get_entry_wild section.
00.
set item_list_addr to reference wild_item_list.
move qui$_display_entry to qui_function.
perform get_que_info.

if sys_status = jbc$_nosuchent
perform send_error
go to fini.

if sys_status not = jbc$_normal go to fini.

move zeros to entry_count.
move "EOF" to closure_type.
perform process_entry until sys_status not = jbc$_normal or closure_type
= "CAN".
if sys_status not = ss$_normal and jbc$_normal and jbc$_nomoreent go to
fini.

call "t3$send"
using by reference eof_msg
by value 5, t3$m_close
giving sys_status.

if sys_status = ss$_normal or t3$_chanclose
perform free_context.
*
fini.
*
```

This is going straight to the pool room

This is going straight to the pool room

get_entry_info section.

00.

move show_entry_number to search_number, job_entry.

move qui\$_display_entry to qui_function.

set item_list_addr to reference entry_item_list.

perform get_que_info.

if sys_status = jbc\$_nosuchent

perform send_error

else

if sys_status = jbc\$_normal

perform send_entry

if sys_status = ss\$_normal and show_entry

call "t3\$send"

using by reference "99EOF"

by value 5, t3\$m_close

giving sys_status.

*

delete_it section.

00.

move show_entry_number to search_number.

call "sys\$sndjbcw"

using by value 0, sjc\$_delete_job, 0

by reference sjc_item_list, sjc_iosb

by value 0, 0

giving sys_status.

if sys_status = ss\$_normal move sjc_cond to sys_status.

if sys_status not = jbc\$_normal

perform send_error

else

call "t3\$send"

using by reference "31"

by value 2, t3\$m_close

giving sys_status.

*

get_que_details section.

00.

move que_info_name to search_name.

move qui\$_display_queue to qui_function.

set item_list_addr to reference que_details_list.

perform get_que_info.

if sys_status not = jbc\$_normal

perform send_error

go to fini.

*

move que_name(1:que_name_len) to sq_que_name.

move scsnode_name(1:scsnode_name_len) to sq_scsnode_name.

move form_name(1:form_name_len) to sq_form_name.

This is going straight to the pool room

This is going straight to the pool room

```
move stock_name(1:stock_name_len) to sq_stock_name.  
move base_priority to sq_base_priority.  
perform get_que_status.
```

```
call "t3$send"  
using by reference show_que_reply  
by value que_details_len, now_close  
giving sys_status.
```

```
go to fini.
```

```
*
```

```
get_que_status.
```

```
*
```

```
move zeros to sts_index, on_or_off.  
perform until sts_index = 11 or on_or_off not = zeros  
add 1 to sts_index  
call "mth$jiand" using que_status,  
que_value (sts_index)  
giving on_or_off  
end-perform.  
if on_or_off = zeros  
move "UNKNOWN" to sq_que_status  
else move que_text (sts_index) to sq_que_status.
```

```
*
```

```
fini.
```

```
*
```

```
get_que_names section.
```

```
00.
```

```
call "str$trim"  
using by descriptor search_name, que_info_name  
by reference que_name_len  
giving sys_status.  
if sys_status not = ss$_normal go to fini.
```

```
if que_name_len < 31  
add 1 to que_name_len  
move "*" to search_name(que_name_len:1).
```

```
move qui$_display_queue to qui_function.  
set item_list_addr to reference que_summary_list.
```

```
perform get_que_info.  
if sys_status not = jbc$_normal  
perform send_error  
go to fini.
```

```
*
```

```
perform until sys_status not = jbc$_normal  
move que_name(1:que_name_len) to sq_que_name
```

```
call "t3$send"  
using by reference show_que_reply
```

This is going straight to the pool room

This is going straight to the pool room

```
by value show_que_len, t3$m_now
giving sys_status
if sys_status = ss$_normal
perform get_que_info
end-if
end-perform.
```

```
if sys_status not = jbc$_nomoreque go to fini.
```

```
call "t3$send"
using by reference eof_msg
by value 2, t3$m_close
giving sys_status.
*
fini.
*
```

```
get_que_jobs section.
00.
set item_list_addr to reference que_summary_list.
move qui$_display_queue to qui_function.
move que_info_name to search_name.
```

```
perform get_que_info.
if sys_status not = jbc$_normal
perform send_error
go to fini.
*
move zeros to entry_count.
move "EOF" to closure_type.
perform until sys_status not = jbc$_normal or closure_type = "CAN"
```

```
move qui$_display_job to qui_function
set item_list_addr to reference job_item_list
perform get_que_info
```

```
perform process_entry until sys_status not = jbc$_normal or
closure_type = "CAN"
if sys_status = jbc$_nomorejob or jbc$_nosuchjob
set item_list_addr to reference que_summary_list
move qui$_display_queue to qui_function
perform get_que_info
end-if
```

```
end-perform.
```

```
if sys_status not = ss$_normal and jbc$_normal and jbc$_nomoreque go to
fini.
```

```
if entry_count = zeros
move jbc$_nosuchjob to sys_status
perform send_error
```

This is going straight to the pool room

This is going straight to the pool room

```
else
call "t3$send"
using by reference eof_msg
by value 5, t3$m_close
giving sys_status.

if sys_status = ss$_normal or t3$_chanclose
perform free_context.
*
fini.
*
process_entry section.
00.
*+
* The following two lines have been included for the purposes of
* demonstrating the GUI "Abort" key processing and "Working. . ."
* functionality.
*_
call "lib$wait" using debug_delay giving sys_status
if sys_status not = ss$_normal call "lib$stop" using by value
sys_status.

add 1 to entry_count.
perform send_entry.

if sys_status = ss$_normal
if cancel_request not = zeros
or entry_count = entry_limit
move "CAN" to closure_type
else
perform get_queue_info.
*
free_context section.
*+
* The maximum number of entries may have been reached before all $GETQUI
information
* had been retrieved for the wildcard search, or the association with the
client may
* have been terminated prematurely by the communication server, therefore we
must
* be sure to tell the job controller that we wish to terminate the wildcard
search.
*_
00.
call "sys$getquiw"
using by value 0, qui$_cancel_operation, 0, 0
by reference qui_iosb
by value 0, 0
giving sys_status.
if sys_status = ss$_normal and qui_cond not = jbc$_normal
move qui_cond to sys_status.
```

This is going straight to the pool room

This is going straight to the pool room

```
*
send_error section.
00.
call "sys$getmsg"
using by value sys_status
by reference error_msg_len
by descriptor error_text
by value getmsg_flags, 0
giving sys_status.
if sys_status not = ss_normal go to fini.

move error_msg_len to error_len.
call "t3$send"
using by reference error_msg
by value buffer_size, t3$m_close
giving sys_status.
*
fini.
*
send_entry section.
00.
move job_entry to se_entry_number.
move job_name(1:job_name_len) to se_job_name.
move que_name(1:que_name_len) to se_que_name.

move zeros to sts_index, on_or_off.
perform until sts_index = 11 or on_or_off not = zeros
add 1 to sts_index
call "mth$jiand" using job_status,
job_value (sts_index)
giving on_or_off
end-perform.
if on_or_off = zeros
move "UNKNOWN" to se_job_status
else move job_text (sts_index) to se_job_status
move zeros to on_or_off.

move zeros to sts_index.
perform until sts_index = 11 or on_or_off not = zeros
add 1 to sts_index
call "mth$jiand" using que_status,
que_value (sts_index)
giving on_or_off
end-perform.
if on_or_off = zeros
move "UNKNOWN" to se_que_status
else move que_text (sts_index) to se_que_status
move zeros to on_or_off.

move zeros to sts_index.
perform until sts_index = 4 or on_or_off not = zeros
```

This is going straight to the pool room

This is going straight to the pool room

```
add 1 to sts_index
call "mth$jiand" using que_flags,
flags_value (sts_index)
giving on_or_off
end-perform.
if on_or_off = zeros
move "UNKNOWN" to se_que_type
else move flags_text (sts_index) to se_que_type.
```

```
if entry_summary_resp go to fini.
```

```
move job_username(1:job_username_len) to se_username.
move account_name(1:account_name_len) to se_account_name.
```

```
if job_uic = zeros
move spaces to se_uic
else
call "sys$fao"
using by descriptor "!%U", omitted, se_uic
by value job_uic
giving sys_status
if sys_status = ss$bufferovf
move ss$normal to sys_status
else
if sys_status not = ss$normal go to fini.
```

```
move que_priority to se_priority.
move job_copies to se_copies.
move job_size to se_size.
```

```
if job_pid = zeros
move spaces to se_pid
else
call "ots$cvt_1_tz"
using by reference job_pid
by descriptor se_pid
by value 8, 4
giving sys_status
if sys_status not = ss$normal go to fini.
```

```
move log_que_name(1:log_que_name_len) to se_log_queue.
```

```
if sub_time = zeros
move spaces to se_submission_time
else
call "sys$asctim"
using by value 0
by descriptor se_submission_time
by reference sub_time
by value 0
giving sys_status
```

This is going straight to the pool room

This is going straight to the pool room

```
if sys_status not = ss$_normal go to fini.
*
fini.
if sys_status = ss$_normal or jbc$_normal
call "t3$send"
using by reference show_entry_reply
by value entry_msg_len, send_flags
giving sys_status.
*
get_que_info section.
00.
call "sys$getquiw"
using by value 0, qui_function, 0 item_list_addr
by reference qui_iosb
by value 0, 0
giving sys_status.
if sys_status = ss$_normal move qui_cond to sys_status.
*
break_link section.
00.
display "Badly formatted request received from client – Connection
closed.".

call "t3$send"
using by reference "00Badly formed message"
by value 22, t3$m_disconnect
giving sys_status.
*
end program user_recv.
*+
* Routine: USER_LOGOFF
*
* This routine is called when the association with the client has been
* terminated, either by the USER_RECV routine or the communication server,
* and the USER_RECV routine has returned control to Tier3.
*
* For the DEMO example we are logging resource usage by the client.
*_
identification division.
program-id. user_logoff.
data division.
working-storage section.
01 iss$c_id_natural pic 9(9) comp value external
iss$c_id_natural.
01 ss$_normal pic s9(9) comp value external
ss$_normal.
01 sys_status pic s9(9) comp.
*
linkage section.
01 demo_context.
03 system_name pic x(8).
```

This is going straight to the pool room

This is going straight to the pool room

```
03 buffer_size pic 9(9) comp.
03 session_user pic x(12).
03 timer_context pic 9(9) comp.
03 session_count pic 9(9) comp.
03 message_count pic 9(9) comp.
03 lostlnk_count pic 9(9) comp.
03 persona_context pic 9(9) comp.
03 cancel_request pic 9(9) comp.
*
procedure division using demo_context giving sys_status.
00.
display "Session completed with user ", session_user.
move spaces to session_user.
*+
* Return back to our natural persona to avoid an identity crisis :-)
*_
call "sys$persona_assume" using iss$c_id_natural, omitted giving
sys_status.
if sys_status not = ss$_normal go to fini.
*+
* Display the user's session statistics.
*
* The Run-Time Library performance reporting routines have been used
* as a very simple example of obtaining information on server resource
* usage by a client. If this is a requirement of your application then
* you should consider obtaining more detailed information via the VMS
* system services $GETTIM and $GETJPI and accumulating totals for each
* user in a disk file. This disk file could be processed at a later
* date as part of charge-back accounting procedures.
*_
call "lib$show_timer" using timer_context giving sys_status.
*
fini.
exit program.
*
end program user_logoff.
*+
* Routine: USER_FINI
*
* This routine is called by Tier3 when the server has been culled from
* the execution server pool due to inactivity timeout.
*
* In the DEMO application there are no files to close or databases to
* disconnect from so the only image rundown activity performed here
* is the logging of server statistics.
*_
identification division.
progr
```

This is going straight to the pool room