Introduction

The *Compaq OpenVMS Galaxy Software Architecture on OpenVMS* introduces a new approach to multi-CPU configurations, built on the solid, proven foundations of existing OpenVMS Symmetric Multiprocessing and VMScluster technologies. This is an evolving technology.

APIs are provided, as well, to detect and affect CPU transitions in real time. This module describes these APIs and illustrates simple applications using these APIs.

Topics

- CPU Migration using DCL
- System Event Detection
- CPU Transitioning Interfaces

CPU Migration Using DCL

CPUs may be migrated from one instance to another using:

```
$ STOP/CPU[/ALL]/MIGRATE=[SCSNODE|PARTITION-ID] [cpu-id]
```

- The following CPUs cannot be stopped and migrated:
  - The Primary CPU
  - A CPU for which a process has affinity

Sample CPU migration using DCL:

```
jygal> show cpu

JYGAL, a AlphaServer 4100 5/300 2MB
Multiprocessing is ENABLED. Streamlined synchronization image loaded.
Minimum multiprocessing revision levels: CPU = 1

PRIMARY CPU = 00
CPU sets:
   Active          00 02 03
   Configure       00 02 03
   Potential       00 01 02 03
   Autostart       00 01 02 03
   Failover        None
jygal> stop/cpu/migrate=jygal2 2,3
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 2 after it reaches quiescent state
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 3 after it reaches quiescent state
```

Sample CPU Migration Using DCL (continued)

- View from instance JYGAL.

```
jygal> show cpu

JYGAL, a AlphaServer 4100 5/300 2MB
Multiprocessing is ENABLED. Streamlined synchronization image loaded.
Minimum multiprocessing revision levels: CPU = 1

PRIMARY CPU = 00
CPU sets:
   Active          00
   Configure       00
   Potential       00 01 02 03
   Autostart       00 01 02 03
   Failover        None
jygal>
```

- View from instance JYGAL2.

```
jygal2> show cpu

JYGAL2, a AlphaServer 4100 5/300 2MB
Multiprocessing is ENABLED. Streamlined synchronization image loaded.
Minimum multiprocessing revision levels: CPU = 1

PRIMARY CPU = 01
CPU sets:
   Active          01 02 03
   Configure       01 02 03
   Potential       00 01 02 03
   Autostart       00 01 02 03
   Failover        None
jygal2> stop/cpu/migrate=jygal 2,3
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 2 after it reaches quiescent state
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 3 after it reaches quiescent state
jygal2>
```

System Event Notification

Processes may register to be notified of system events in real time.

- These system events include the adding/deleting of members, active_cpus, and or configured CPUs.
- Event codes:

```
SYSEVT$C_ADD_MEMBER
SYSEVT$C_DEL_MEMBER
SYSEVT$C_ADD_ACTIVE_CPU
SYSEVT$C_DEL_ACTIVE_CPU
SYSEVT$C_ADD_CONFIG_CPU
SYSEVT$C_DEL_CONFIG_CPU
```

- Event services:

```
int  sys$set_system_event(unsigned int event,
void (*astadr)(__unknown_params),
            int astprm, unsigned int acmode,
unsigned int flags, struct _generic_64 *handle);

int  sys$clear_system_event(struct _generic_64 *handle,
unsigned int acmode, unsigned int event);
```

- The handle argument must be 0 to set up for unique event handling.
- Event notification is performed through AST routine calls.

System Event Notification Example

Program to distinguish CPUs involved in event notification.

```
$ type cpu_detector.c
/*  Program to illustrate the use of the sys$set_system_event
    system service to detect the adding/deleting of CPUs in instances
    of a galaxy.

*/
#include <stdio.h>
#include <starlet.h>
#include <ssdef.h>
#include <sysevtdef.h>
#include <unistd.h>
#include <syidef.h>
#include <iledef.h>
#define check(STS) if(!((STS)&1)) sys$exit(STS)
int     evt_ast(__int64 evt_type);
static  unsigned int last_active_mask;
static  unsigned int last_conf;
unsigned int cpu_info(int       itype);
int     main(void)
{
/* Set up event list. */
       __int64 events[] = {SYSEVT$C_ADD_GALAXY_MEMBER,
                           SYSEVT$C_DEL_GALAXY_MEMBER,
                           SYSEVT$C_ADD_ACTIVE_CPU,SYSEVT$C_DEL_ACTIVE_CPU,
                           SYSEVT$C_ADD_CONFIG_CPU,SYSEVT$C_DEL_CONFIG_CPU};
```

System Event Notification Example (continued)

```
        const   int     NUM_EVTS=sizeof(events)/sizeof(*events);
        int     i;
        int     status;
        __int64 handle = 0;

         last_active_mask = cpu_info(SYI$_ACTIVE_CPU_MASK);
         last_conf = cpu_info(SYI$_CPUCONF);
puts("*****************************************");
         for(i=0;i<NUM_EVTS;i++)
         {
/* Enable AST delivery for unique events with an ASTPRM that identifies
   which event we are detecting.
*/
                 status = sys$set_system_event(events[i],evt_ast,events[i],
                                 0,SYSEVT$M_REPEAT_NOTIFY,&handle);
                 check(status);
                 handle = 0;
         }
/* Go to sleep and let the ASTs handle the rest. */
         sleep(600);
         return(SS$_NORMAL);
}
void     disp_new(void);
void     disp_del(void);
/* AST routine to handle events. */
int      evt_ast(__int64 evt_type)
{
```

System Event Notification Example (continued)

```
/* Notify user of event occurrence and go away. */
        switch(evt_type)
        {
                case    SYSEVT$C_ADD_GALAXY_MEMBER:
                        puts("Community member added");
        /* Display configuration changes. */
                        disp_new();
                        break;
                case    SYSEVT$C_DEL_GALAXY_MEMBER:
                        puts("Community member deleted");
        /* Display configuration changes. */
                        disp_del();
                        break;
                case    SYSEVT$C_ADD_ACTIVE_CPU:
                        puts("Active CPU added");
        /* Display configuration changes. */
                        disp_new();
                        break;
                case    SYSEVT$C_DEL_ACTIVE_CPU:
                        puts("Active CPU deleted");
        /* Display configuration changes. */
                        disp_del();
                        break;
                case    SYSEVT$C_ADD_CONFIG_CPU:
                        puts("Instance CPU added");
        /* Display configuration changes. */
                        disp_new();
                        break;
```

System Event Notification Example (continued)

```
                case    SYSEVT$C_DEL_CONFIG_CPU:
                        puts("Instance CPU deleted");
        /* Display configuration changes. */
                        disp_del();
                        break;
                default:
                        puts("I am confused!");
        }
        return(SS$_NORMAL);
}

/* Function to list additions to cpu configuration database. */
#include <syidef.h>
void    disp_new(void)
{
        int     i;
        unsigned int new_active;
        unsigned int new_conf;
/* Determine the changes to the active cpu mask. */
        new_active = cpu_info(SYI$_ACTIVE_CPU_MASK) & ~last_active_mask;
        last_active_mask = cpu_info(SYI$_ACTIVE_CPU_MASK) ;
/* Determine the changes to the potential cpu mask. */
        new_conf = cpu_info(SYI$_CPUCONF) & ~last_conf;
        last_conf = cpu_info(SYI$_CPUCONF) ;
/* Display the new CPUs in the active list. */
        if(new_active)
        {
                printf("Newly active CPU ids: ");
```

System Event Notification Example (continued)

```
                for(i=0;i<SYI$_MAX_CPUS;i++,new_active >>=1)
                {
                        if(new_active&1)
                        {
                        printf(" %02d",i);
                        }
                }
                printf("\n");
        }
/* Display the new CPUs in the configured list. */
        if(new_conf)
        {
                printf("Newly configured CPU ids: ");
                for(i=0;i<SYI$_MAX_CPUS;i++,new_conf >>=1)
                {
                        if(new_conf&1)
                        {
                                printf(" %02d",i);
                        }
                }
                printf("\n");
        }
puts("*****************************************");
}
/* Function to list deletions from cpu configuration database.
   Design note, this was cut and modified from the previous
   routine.  They probably could have been combined. */
```

System Event Notification Example (continued)

```
#include <syidef.h>
void    disp_del(void)
{
        int     i;
        unsigned int new_active;
        unsigned int new_conf;
/* Determine the changes to the active cpu mask. */
        new_active = ~cpu_info(SYI$_ACTIVE_CPU_MASK) & last_active_mask;
        last_active_mask = cpu_info(SYI$_ACTIVE_CPU_MASK) ;
/* Determine the changes to the potential cpu mask. */
        new_conf = ~cpu_info(SYI$_CPUCONF) & last_conf;
        last_conf = cpu_info(SYI$_CPUCONF) ;
/* Display the new CPUs in the active list. */
        if(new_active)
        {
                printf("Newly deleted CPU ids: ");
                for(i=0;i<SYI$_MAX_CPUS;i++,new_active >>=1)
                {
                        if(new_active&1)
                        {
                        printf(" %02d",i);
                        }
                }
                printf("\n");
        }
/* Display the new CPUs in the configured list. */
        if(new_conf)
        {
                printf("Deleted configured CPU ids: ");
```

System Event Notification Example (continued)

```
                for(i=0;i<SYI$_MAX_CPUS;i++,new_conf >>=1)
                {
                        if(new_conf&1)
                        {
                                printf(" %02d",i);
                        }
                }
                printf("\n");
        }
puts("*****************************************");
}
/* Function to get SYI information on cpu masks and return as
        an unsigned int.
*/
unsigned int cpu_info(int       itype)
{
        int     status;
        unsigned int    ret_val;
        ile3    items[] = {{sizeof(ret_val),0},
                                {0,SYI$C_LISTEND}};
/* Set item code based on passed parameter. */
        items[0].ile3$w_code = itype;
        items[0].ile3$ps_bufaddr = &ret_val;
/* Get the info. */
        if((status = sys$getsyi(0, 0, 0, items, 0, 0, 0)) & 1)
        {
                return(ret_val);
```

System Event Notification Example (continued)

```
        }
        else
        {
                sys$exit(status);
        }
}

$
$
```

System Event Notification Example (continued)

Sample run of CPU detector.

```
$ show cpu

WFGLXA, a Compaq AlphaServer GS320 6/731
Multiprocessing is ENABLED. Full checking synchronization image loaded.

Primary CPU = 000
CPU sets:
   Active              0-3,8,12,13
   Configure           0-3,8,12,13
   Powered Down        None
   Potential           0-8,10-15
   Autostart           0-31
   Failover            None
$ spawn/nowait r cpu_detector
%DCL-S-SPAWNED, process SYSTEM_1 spawned
$
  ****************************************
```

Transition CPUs to another instance.

```
$ stop/cpu/migrate=wfglxb 12,13
%SMP-I-CPUTRN, CPU #12 was removed from the active set.
```

From CPU_DETECTOR

```
Active CPU deleted
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 12 after it reaches quiescent state
```

From CPU_DETECTOR

```
Newly deleted CPU ids: 12
%SMP-I-CPUTRN, CPU #13 was removed from the active set.
STEM-I-CPUSTOPPING, trying to stop CPU 13 after it reaches quiescent state
```

From CPU_DETECTOR

```
Deleted configured CPU ids: 12
```

System Event Notification Example (continued)

```
$
*****************************************
                                    From CPU_DETECTOR
Instance CPU deleted
Newly deleted CPU ids:  13
Deleted configured CPU ids:  13
*****************************************
Active CPU deleted
*****************************************
$

++++++++++++++++++++++++++++++++
                          From another instance, transition CPUs back to original.
$ write sys$output f$getsyi("scsnode")
WFGLXB
$ sh cpu

WFGLXB, a Compaq AlphaServer GS320 6/731
Multiprocessing is ENABLED. Full checking synchronization image loaded.

Primary CPU = 004
CPU sets:
   Active              4-7,10-15
   Configure           4-7,10-15
   Powered Down        None
   Potential           0-8,10-15
   Autostart           0-31
   Failover            None
```

System Event Notification Example (continued)

```
$ stop/cpu/mig=wfglxa 12,13
%SMP-I-CPUTRN, CPU #12 was removed from the active set.
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 12 after it reaches quiescent state
%SMP-I-CPUTRN, CPU #13 was removed from the active set.
%SYSTEM-I-CPUSTOPPING, trying to stop CPU 13 after it reaches quiescent state
$
++++++++++++++++++++++++++++++++++
```

From CPU_DETECTOR.

```
  Instance CPU added
Newly configured CPU ids: 12 13
%SMP-I-SECMSG, CPU #12 message:   START
%SMP-I-CPUTRN, CPU #12 has joined the active set.
%SMP-I-SECMSG, CPU #13 message:   START
%SMP-I-CPUTRN, CPU #13 has joined the active set.
$
****************************************
```

From CPU_DETECTOR.

```
Instance CPU added
Newly active CPU ids:  12 13
****************************************
Active CPU added
****************************************
```

System Event Notification Example (continued)

Sample Reboot of another instance.

```
SHUTDOWN message on WFGLXA from user SYSTEM at _WFGLXB$OPA0:    02:09:53
WFGLXB will shut down in 0 minutes; back up shortly via automatic reboot.  Pleas
e log off node WFGLXB.
Standalone
```

From CPU_DETECTOR

```
Community member deleted
*****************************************
%CNXMAN,  Lost connection to system WFGLXB
```

From CPU_DETECTOR

```
Community member added
*****************************************
%PBA0 CPU00:  21-NOV-2000 02:11:15 Port is Reinitializing.
%CNXMAN,  Received VMScluster membership request from system WFGLXB
%CNXMAN,  Proposing addition of system WFGLXB
%CNXMAN,  Completing VMScluster state transition
```

CPU Management Programming Interface

CPUs can be transitioned in a form similar to **$STOP/CPU/MIGRATE** under program control.

- To transition CPUs use:

```
SYS$CPU_TRANSITION[W]     tran_code, cpu_id, nodename, node_id, flags, efn, iosb,
    astadr_64, astprm_64, timout
```

- Transition codes:

```
CST$K_CPU_STOP
CST$K_CPU_MIGRATE
CST$K_CPU_START
CST$K_CPU_FAILOVER
```

- Flags:

```
CST$V_CPU_DEFAULT_CAPABILITIES
CST$V_CPU_ALLOW_ORPHANS
```

- Requires CMKRNL privilege.

CPU Transitioning Example

Program which runs as a foreign command and transitions CPUs to instance "WFGLXB" through migration.

```
$ type push_cpu.c
/* Simple program to illustrate the SYS$CPU_TRANSITIONW
   system service.  Emulates $STOP/CPU/MIGRATE=WFGLXB id
 */
#include <stdlib.h>
#include <starlet.h>
#include <ssdef.h>
#include <cstdef.h>
#include <descrip.h>
#include <stdio.h>
#include <iosbdef.h>
#define check(STS) if(!((STS)&1)) sys$exit(STS)
#define EXP_ARGS 2
#define CPU_ARG 1
int     main(int argc,char **args)
{
        int     status;
        int     CPU;
        $DESCRIPTOR(other_inst,"WFGLXB");
        iosb    stat;
        const   int     TMO = 10;

 /* Validate that we received a cpu id and convert it. */
        if(argc != EXP_ARGS) return(SS$_INSFARG);
        CPU = atoi(args[CPU_ARG]);
```

CPU Management Programming Interface

CPUs can be transitioned in a form similar to **$STOP/CPU/MIGRATE** under program control.

- To transition CPUs use:

```
SYS$CPU_TRANSITION[W]     tran_code, cpu_id, nodename, node_id, flags, efn, iosb,
    astadr_64, astprm_64, timout
```

- Transition codes:

```
CST$K_CPU_STOP
CST$K_CPU_MIGRATE
CST$K_CPU_START
CST$K_CPU_FAILOVER
```

- Flags:

```
CST$V_CPU_DEFAULT_CAPABILITIES
CST$V_CPU_ALLOW_ORPHANS
```

- Requires CMKRNL privilege.

CPU Transitioning Example

Program which runs as a foreign command and transitions CPUs to instance "WFGLXB" through migration.

```
$ type push_cpu.c
/* Simple program to illustrate the SYS$CPU_TRANSITIONW
   system service.  Emulates $STOP/CPU/MIGRATE=WFGLXB id
 */
#include <stdlib.h>
#include <starlet.h>
#include <ssdef.h>
#include <cstdef.h>
#include <descrip.h>
#include <stdio.h>
#include <iosbdef.h>
#define check(STS) if(!((STS)&1)) sys$exit(STS)
#define EXP_ARGS 2
#define CPU_ARG 1
int     main(int argc,char **args)
{
        int     status;
        int     CPU;
        $DESCRIPTOR(other_inst,"WFGLXB");
        iosb    stat;
        const   int     TMO = 10;

 /* Validate that we received a cpu id and convert it. */
        if(argc != EXP_ARGS) return(SS$_INSFARG);
        CPU = atoi(args[CPU_ARG]);
```

CPU Transitioning Example (continued)

```
/* Migrate that CPU to JYGAL2. */
        status =  sys$cpu_transitionw(CST$K_CPU_MIGRATE,CPU,
                        &other_inst,0,0,0,&stat,0,0,TMO);
        check(status);
        check(stat.iosb$w_status);
        return(SS$_NORMAL);
}

$
$ sh cpu

WFGLXA, a Compaq AlphaServer GS320 6/731
Multiprocessing is EABLED. Full checking synchronization image loaded.

Primary CPU = 000
CPU sets:
   Active              0-3,8-11
   Configure           0-3,8-11
   Powered Down        None
   Potential           0-15
   Autostart           0-31
   Failover            None
$
```

CPU Transitioning Example (continued)

Set up a foreign command to run the program.

```
$ push=="$sys$sysdevice:[ballen]push_cpu"
$ push 8
%SMP-I-CPUTRN, CPU #08 was removed from the active set.
$ push 9
%SMP-I-CPUTRN, CPU #09 was removed from the active set.
                                                    Active CPU deleted
$
```

Note that CPU Ids 8 and 9 have now been transitioned.

```
$ sh cpu

WFGLXA, a Compaq AlphaServer GS320 6/731
Multiprocessing is ENABLED. Full checking synchronization image loaded.

Primary CPU = 000
CPU sets:
   Active              0-3,10,11
   Configure           0-3,10,11
   Powered Down        None
   Potential           0-15
   Autostart           0-31
   Failover            None
$
```

Introduction

The Command Definition Utility supports the use of internal command parsing, similar to that done by the Authorize, Mail, Install utilities. This module describes the methods required to use this interface and provides an example. See the *OpenVMS Utilities Routines Manual* for more details.

Topics

- DCL Parsing Interfaces
- Sample DCL Parser

DCL Parsing Interfaces

To use the Command Definition Utility to support internal command parsing use the following steps:

1. Create a Command Language Definition (CLD) File that includes:

- A Module name that will be used to define a parse table for **CLI$DCL_PARSE**.
- For each verb:
  a. Define a routine to dispatch to on verb entry.
  b. Optionally, define parameters and qualifiers that may be processed in the action routine, using **CLI$GET_VALUE** and/or **CLI$PRESENT**.

2. Compile the CLD file using: **$ SET COMMAND/OBJECT *x*.CLD**.
3. Within your program loop, using **CLI$DCL_PARSE** to prompt for, read, and parse the user input.
4. Dispatch to the action routine associated with the verb, using **CLI$DISPATCH**.
5. In the action routine, process the rest of the command, using **CLI$GET_VALUE** and/or **CLI$PRESENT**.
6. Compile the program and associated routines.
7. Link the program and associated routines with the object module produced in step 2.

Sample DCL Parser

Command Language Definition File

```
$ type convert.cld
    MODULE  cvt_routines
    DEFINE VERB CONVERT
      ROUTINE convert
      PARAMETER P1, value(required)
      QUALIFIER HEX
      QUALIFIER OCTAL
      QUALIFIER DECIMAL, DEFAULT
    QUALIFIER BINARY
DEFINE VERB EXIT
    ROUTINE DONE
DEFINE VERB BYE
    ROUTINE DONE
$
```

Program using the parser.

```
$ type converter.c
    /* Sample of using DCL to parse command lines.
    */
#include <cli$routines.h>
#include <libclidef.h>
#include <descrip.h>
#include <lib$routines.h>
#include <stdio.h>
#include <climsgdef.h>
#include <starlet.h>
#include <ssdef.h>
#include <rms.h>
extern void *cvt_routines;
int  bin_cvt(char *str,int *val);
char *bin_2_str(char *str,unsigned int val);
int  convert(void);
```

Sample DCL Parser (continued)

```
int main(void)
    {

      $DESCRIPTOR(pmt,"Converter> ");
      int    status;

    /* Get commands until user terminates or enters EOF. */
      while((status = cli$dcl_parse(0,&cvt_routines,lib$get_input,
            lib$get_input,&pmt)) != RMS$_EOF)
    {
/* If we received a verb, dispatch to action routine. */
       if(status&SS$_NORMAL)
       {
             CLI$DISPATCH();
       }
       else if(status == CLI$_IVVERB)
       {
             fprintf(stderr,"Try, CONVERT, EXIT, or BYE\n");

       }
     }
     return(SS$_NORMAL);
}
/* Entered on verbs: EXIT or BYE. */
void done()
{
     sys$exit(SS$_NORMAL);
}
```

Sample DCL Parser (continued)

```
#define MAX_IN 255
#define ONE_CNV 1
/* Entered on CONVERT verb. */
int  convert()
{
     $DESCRIPTOR(value,"P1");
     $DESCRIPTOR(hex_qual,"HEX");
     $DESCRIPTOR(bin_qual,"BINARY");
     $DESCRIPTOR(oct_qual,"OCTAL");
     $DESCRIPTOR(dec_qual,"DECIMAL");
     static   char   val_str[MAX_IN+sizeof("\0")];
     struct   dsc$descriptor_s value_d = {MAX_IN,DSC$K_DTYPE_T,
                    DSC$K_CLASS_S,val_str};
     int      val_bin;
     char     c;
       int    status;

 /* Get parameter. */
      status = cli$get_value(&value,&value_d,&value_d.dsc$w_length);
      if(!(status&1))
      {
        return(status);
      }
 /* Make it a C string. */
        val_str[value_d.dsc$w_length] = '\0';
 /* Convert to hex, octal, binary, or decimal.
    Note: order of checking is important, since decimal
    will be defaulted.
 */
```

Sample DCL Parser (continued)

```
/* Hex */
if(cli$present(&hex_qual)&SS$_NORMAL)
{
  if(sscanf(val_str,"%x%c",&val_bin,&c)!=ONE_CNV)
  {
        printf("Bad input\n");
        return(SS$_NORMAL);
  }
}
/* Octal */
else if(cli$present(&oct_qual)&SS$_NORMAL)
{
  if(sscanf(val_str,"%o%c",&val_bin,&c)!=ONE_CNV)
  {
        printf("Bad input\n");
        return(SS$_NORMAL);
  }
}
/* Binary */
else if(cli$present(&bin_qual)&SS$_NORMAL)
{
  if(!bin_cvt(val_str,&val_bin))
  {
        printf("Bad input\n");
        return(SS$_NORMAL);
  }
}
```

Sample DCL Parser (continued)

```
/* Decimal */
   else if(cli$present(&dec_qual)&SS$_NORMAL)
   {
     if(sscanf(val_str,"%d%c",&val_bin,&c)!=ONE_CNV)
     {
           printf("Bad input\n");
           return(SS$_NORMAL);
     }
   }
   printf("%x (Hex)\n",val_bin);
   printf("%d (Decimal)\n",val_bin);
   printf("%o (Octal)\n",val_bin);
   printf("%s (Binary)\n",bin_2_str(val_str,val_bin));
   return(SS$_NORMAL);
}
```

Sample DCL Parser (continued)

```
/* Convert binary to string in binary. */
    #define BIN_DIG 32
    #define LOW_BIT 1
    char    *bin_2_str(char *str,unsigned int val)
    {
      int   i;
      for(i=0;i<BIN_DIG;i++)
      {
            str[i] = '0';
      }
      str[i] = '\0';
/* walk from high bit (right) to low. */
     while(val)
     {
       --i;
       if(val&LOW_BIT) str[i] = '1';
       val>>=1;
     }
     return(str);
}
/* Convert string in binary to binary. */
#include <string.h>
#define CNV_ERR 0
int bin_cvt(char *str,int *val)
{
     int     i;
     int     slen;
     *val = 0;
     if((slen=strlen(str))>BIN_DIG) return(CNV_ERR);
```

Sample DCL Parser (continued)

```
/* walk from low bit (left) to high. */
    for(i=0;i<slen;i++)
    {
      (*val) <<=1;
      switch(str[i])
      {
            case '0':
                  break;
            case '1':
                  (*val) |= 1;
                  break;
            default:
                  return(CNV_ERR);
      }
    }
    return(SS$_NORMAL);
}
$
```

Sample DCL Parser (continued)

```
$
    $
    $ set command/object convert
    $ cc converter
    $ link converter,convert
    $
    $ run converter
    Converter> convert 12
    c (Hex)
    12 (Decimal)
14 (Octal)
00000000000000000000000000001100 (Binary)
Converter> convert 1ff/hex
1ff (Hex)
511 (Decimal)
777 (Octal)
00000000000000000000000111111111 (Binary)
Converter> convert 1010/bin
a (Hex)
10 (Decimal)
12 (Octal)
00000000000000000000000000001010 (Binary)
Converter> convert 123d
Bad input
Converter> convert 12/bin
Bad input
Converter> xxx
```

Sample DCL Parser (continued)

```
%CLI-W-IVVERB, unrecognized command verb - check validity and spelling
 \XXX\
%TRACE-W-TRACEBACK, symbolic stack dump follows
  image    module    routine              line      rel PC           abs PC
                                             0 FFFFFFFF800609E4 FFFFFFFF800609E4
                                             0 FFFFFFFF800609E4 FFFFFFFF800609E4
                                             0 FFFFFFFF800609E4 FFFFFFFF800609E4
 CONVERTER                                   0 00000000000109FC 00000000000209FC
 CONVERTER  CONVERTER  main              28483 0000000000000184 0000000000020184
 CONVERTER  CONVERTER  __main                0 0000000000000068 0000000000020068
                                             0 FFFFFFFF83897474 FFFFFFFF83897474

Try, CONVERT, EXIT, or BYE
Converter> ^Z
$
$ link/notrace converter,convert
$
$ r converter

Converter> 123
%CLI-W-IVVERB, unrecognized command verb - check validity and spelling
 \123\
Try, CONVERT, EXIT, or BYE
Converter> convert 666/octal
1b6 (Hex)
438 (Decimal)
666 (Octal)
00000000000000000000000110110110 (Binary)
Converter> bye
$
```

Introduction

Version 7.3 of OpenVMS supports a programming interface for gathering performance statistics.

Topics

- Performance Management API

The Performance Management API

V7.3 of OpenVMS supports the SYS$GETRMI system service for use in obtaining performance management statistics.

- Format:

```
int sys$getrmi(unsigned int efn, unsigned int , unsigned int ,
void *itmlst, struct _iosb *iosb, void (*astadr)(__unknown_params), int
     astprm);
```

- Item list (itmlst) is of the form ile3.
- Ile3 structure format:

ile3$w_code	ile3$w_length
ile3$ps_bufaddr	
ile3$ps_retlen_addr	

- Item codes are documented in *OpenVMS System Services Reference Manual: A—GETUAI.*
- Item codes parallel fields displayed by the MONITOR utility and are of the form RMI$_*item*. Examples:
  - RMI$_DZROFLTS
  - RMI$_ENQCVT
  - RMI$_ENQNEW
  - RMI$_FAULTS
  - RMI$_PREADIO

SYS$GETRMI Example

```
$ type perf.c
/*****************************
Example of obtaining performance statistics using SYS$GETRMI

*****************************/
#include <starlet.h>
#include <rmidef.h>
#include <iosbdef.h>
#include <iledef.h>
#include <stdio.h>
#define check(S) if(!((S)&1)) sys$exit(S)
#define RMI_EF 32
#include <ssdef.h>

int     main(void)
{
        int status;

        static  int     new_enq;
        static  int     cvt_enq;
        static  int     tot_faults;
        static  int     hard_faults;
/*** $GETRMI item list. ***/
        ile3    rmi_items[] = {
                {sizeof(new_enq),RMI$_ENQNEW,&new_enq},
                {sizeof(cvt_enq),RMI$_ENQCVT,&cvt_enq},
                {sizeof(tot_faults),RMI$_FAULTS,&tot_faults},
                {sizeof(hard_faults),RMI$_PREADIO,&hard_faults},
                {0,0}};
        iosb    io_stat;
```

SYS$GETRMI Example (continued)

```
/*** Gather the stats. ***/
        status = sys$getrmi(RMI_EF,0,0,rmi_items,&io_stat,0,0);
        sys$waitfr(RMI_EF);
        check(status);
        check(io_stat.iosb$w_status);
/*** Display the stats. ***/
        printf("ENQs == %d\n",new_enq);
        printf("Converted ENQs == %d\n",cvt_enq);
        printf("Total faults == %d\n",tot_faults);
        printf("Hard faults == %d\n",hard_faults);
        return(SS$_NORMAL);
}
$
$ run perf
ENQs == 309235
Converted ENQs == 4222
Total faults == 45540
Hard faults == 4978
$
```

Introduction

The $SNDJBC system service tends to be one of the more difficult services to use. Often, all that is needed is a simple example of using it to get beyond the initial hurdles in comprehension. The example, included here, submits a simple batch job to the SYS$BATCH queue. The same mechanism would be used for print jobs, but you would use a print queue as your target.

Topics

- Sample Submit From a Program

Sample Submit From a Program

```
$ type my_submit.c
/* Sample call to $SNDJBC.
   Submits XXX.COM to SYS$BATCH with /NOPRINT and
   /NOTIFY equivalents.
*/

#include <starlet.h>
#include <ssdef.h>
#include <sjcdef.h>
#include <iledef.h>
#include <iosbdef.h>
#include <string.h>
#define QUEUE_ARG 0
#define FILE_ARG 1
#define check(STS) if(!((STS)&1)) sys$exit(STS)
int main(void)
{
    char    queue[]="SYS$BATCH";
    char    job[]="xxx.com";
    iosb    sjc_iosb;
    int     status;
    ile3    laundry_list[] = {{0,SJC$_QUEUE},{0,SJC$_FILE_SPECIFICATION},
                              {0,SJC$_NO_LOG_SPOOL,0,0},
                              {0,SJC$_NOTIFY,0,0}
                              ,{0}};
```

Sample Submit From a Program (continued)

```
/* Note:  this service is a little weird in that it does not use
   string descriptors, but actual strings.
*/
     laundry_list[QUEUE_ARG].ile3$w_length = strlen(queue);
     laundry_list[QUEUE_ARG].ile3$ps_bufaddr = queue;
     laundry_list[FILE_ARG].ile3$w_length = strlen(job);
     laundry_list[FILE_ARG].ile3$ps_bufaddr = job;

/* Enter the file. Use SYNCHRONOUS form. */
     status = sys$sndjbcw(0, SJC$_ENTER_FILE,0,laundry_list,&sjc_iosb,
              0,0);
/* Make sure that you check both the call and completion status. */
     check(status);
     check(sjc_iosb.iosb$w_status);
     return(SS$_NORMAL);
}
$
```

Sample Submit From a Program (continued)

```
$ cc my_submit
$ link my_submit
$ show queue sys$batch
Batch queue SYS$BATCH, idle, on ALLEN::
$ type xxx.com
$SHOW TIME
$
$ run my_submit
Job XXX (queue SYS$BATCH, entry 4) completed
$
$ type xxx.log
  14-DEC-1998 23:32:07
  SYSTEM       job terminated at 14-DEC-1998 23:32:07.43
  Accounting information:
  Buffered I/O count:                31      Peak working set size:       2032
  Direct I/O count:                  24      Peak virtual size:         167744
  Page faults:                      174      Mounted volumes:                0
  Charged CPU time:       0 00:00:00.48      Elapsed time:       0 00:00:00.90

$
```