Virtualization Standards for Business Continuity: Part 4

This is the fourth of a series of articles defining the policies, guidelines, standards, and procedures that provide the foundation of a virtualized environment, thus enabling business continuity, disaster recovery, and high availability, with an emphasis toward Return On Investment (ROI).

The focus of this article is to present a shell script that provides the HTML report generator for the web based reporting mechanism discussed in the previous article in this series, which presented the data gathering portion of the reporting mechanism.

The data gathering script "gethmcdata.ksh" captures frame, LPAR, slot, and virtual I/O information from an HMC and stores that information to a directory on the system where the script was executed. The HTML report generator script, "processdata.ksh", reads the data gathered from one or more HMC's, and automatically generates web based documentation. This documentation is saved to a local directory on the system where the script was executed.

These reports provide configuration documentation for all physical and virtual resources assigned to all systems managed by one or more HMC's. Together, these scripts provide system administrators with the ability to automate the gathering of configuration information, as well as automating the process of document generation. The existence of documents that provide detailed information for system configuration is essential to operating an organization under a business continuity mentality.

The reason for automating this process is again, business continuity. In order to ensure that an organization has the ability to conduct business, accurate and up-to-date configuration information must be available to system administrators, security administrators, management, auditors, and regulators. Automation ensures this information not only exists, but is up-to-date as well, and frees the system administrator to perform other tasks. The job description of any system administrator should include as a primary duty, to automate any task that can be automated, and documentation should be at the top of that task list.

```
#!/usr/bin/ksh93
################################################################
function usagemsg_processdata {
  print "
Program: processdata

Generates HTML documents from frame, LPAR, adapter, and slot
information gathered from one or more HMCs.

Usage: ${1##*/} [-?vV] [-d /localdir] [-w /htmldocs] [-M]

  Where:
    -d = Local Directory where HMC data is stored
         (default: /tmp/hmcdata)
    -w = Local Directory in which to store generated HTML reports
         (default: /tmp/htmldata)
    -M = Enable \"menugen\" web site generator compatibility mode
         See http://www.mtxia.com/css/Downloads/Documentation/
    -v = Verbose mode - displays processdata function info
    -V = Very Verbose Mode - debug output displayed

Author: Dana French (dfrench@mtxia.com) Copyright 2007
\"AutoContent\" enabled
"
}
################################################################
####
#### Description:
####
#### Retrieves frame, LPAR, adapter, and slot information
```

Virtualization Standards for Business Continuity: Part 4

```
#### from an HMC.
####
#### Assumptions:
####
#### This script assumes the "gethmcdata.ksh" script has
#### already generated the HMC data files.
####
#### Dependencies:
####
#### Products:
####
#### This script generates HTML formated web pages for
#### displaying frame, LPAR, adapter, and slot information
#### from an HMC.
####
#### Configured Usage:
####
#### Details:
####
#################################################################
function processdata_mgput {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  typeset DD="${2:?# ERROR: ${0}: HTML directory not specified}"
  typeset PA="${3:?# ERROR: ${0}: parent not specified}"
  typeset CH="${4:?# ERROR: ${0}: child not specified}"
  typeset DE="${5:?# ERROR: ${0}: description not specified}"

  exec 9>>"${DD}/menugen.dat"
  processdata_print 9 "${PA}\t\t${CH}\t\t${DE}"
  exec 9>&-

  return 0
}
#################################################################
function processdata_print {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  typeset PS="${2:?# ERROR: ${0}: print string not specified}"
  print -u ${FD} -- "${PS}"
  return 0
}
#################################################################
function processdata_footer {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  processdata_print ${FD} "\n<P><HR></P>"
  processdata_print ${FD} "<P>This page was generated on $( date ) from a Korn Shell Script written by"
  processdata_print ${FD} " <A Href=\"mailto:dfrench@mtxia.com\">Dana French</A>,"
  processdata_print ${FD} " <A Href=\"http://www.mtxia.com/css\">Mt Xia Inc.</A></P>"
  return 0
}
#################################################################
function processdata_titlebar {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  typeset T1="${2:?# ERROR: ${0}: Title line 1 not specified}"
  typeset T2="${3:?# ERROR: ${0}: Title line 2 not specified}"
  processdata_print ${FD} "\n<!-- ################################################################## -->"
  processdata_print ${FD} "\n<P><HR></P>"
  processdata_print ${FD} "\n<P><H3>${T1}</H3></P>"
  processdata_print ${FD} "\n<P><BLOCKQUOTE><H4>${T2}</H4></BLOCKQUOTE></P>"
  return 0
}
#################################################################
function processdata_tablehead {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  typeset TH="${2:?# ERROR: ${0}: Table header value not specified}"
  processdata_print ${FD} "    <TH Bgcolor=\"lightgrey\">"
  processdata_print ${FD} "${TH}"
  processdata_print ${FD} "    </TH>"
  return 0
}
#################################################################
```

Virtualization Standards for Business Continuity: Part 4

```
function processdata_tablecell {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  typeset TD="${2:?# ERROR: ${0}: Table data value not specified}"
  processdata_print ${FD} "    <TD Bgcolor=\"white\">"
  processdata_print ${FD} "${TD}"
  processdata_print ${FD} "    </TD>"
  return 0
}
################################################################
function processdata_rowbegin {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  processdata_print ${FD} "<P>"
  processdata_print ${FD} "  <TR>"
  return 0
}
################################################################
function processdata_rowend {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  processdata_print ${FD} "  </TR>"
  processdata_print ${FD} "</P>"
  return 0
}
################################################################
function processdata_tablebegin {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  processdata_print ${FD} "\n<P>"
  processdata_print ${FD} "<TABLE Border=\"1\">"
  return 0
}
################################################################
function processdata_tableend {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  processdata_print ${FD} "</TABLE>"
  processdata_print ${FD} "</P>"
  return 0
}
################################################################
function processdata_htmlbegin {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  typeset TT="${2:?# ERROR: ${0}: HTML page title not specified}"
  processdata_print ${FD} "<HTML>"
  processdata_print ${FD} "<HEAD>"
  processdata_print ${FD} "<TITLE>${TT}</TITLE>"
  processdata_print ${FD} "</HEAD>"
  processdata_print ${FD} "<BODY Bgcolor=\"white\" Text=\"black\" Link=\"blue\" Alink=\"yellow\" Vlink=\"red\">"
  processdata_print ${FD} "<P><H2>Managed System Resources</H2></P>"
  processdata_print ${FD} "<P><BLOCKQUOTE><I><STRONG>${TT}</STRONG></I></BLOCKQUOTE></P>"
  return 0
}
################################################################
function processdata_htmlend {
  typeset FD="${1:?# ERROR: ${0}: file descriptor number not specified}"
  processdata_print ${FD} "</BODY>"
  processdata_print ${FD} "</HTML>"
  return 0
}
################################################################
function processdata {
  typeset VERSION="1.0"
  typeset TRUE="1"
  typeset FALSE="0"
  typeset LCNT="0"
  typeset TCNT="0"
  typeset VERBOSE="${FALSE}"
  typeset VERYVERB="${FALSE}"
  typeset HMCNAME=""
  typeset HMCUSER="hscroot"
  typeset SYSNAME=""
  typeset MENUGEN="${FALSE}"
  typeset HTMLEXT="html"
```

Virtualization Standards for Business Continuity: Part 4

```
  typeset MGPART=""
  typeset HMCDATA="/tmp/hmcdata"
  typeset HMCWWW="/tmp/htmldata"
  typeset HMCNAME
  typeset PARMNAME
  typeset LPARNAME
  typeset LPARFILE
  typeset ATTRLINE
  typeset ATTRFILE
  typeset ATTRVAR
  typeset ATTRVAL
  typeset PREFIX
  typeset MGNTXT
  typeset IDX
  typeset ARG
  typeset ARGS

#### Process the command line options and arguments, saving
#### the values as appropriate.

  while getopts ":vVMd:w:" OPTION
  do
     case "${OPTION}" in
        'd') HMCDATA="${OPTARG}"
             HMCDATA="${HMCDATA%%+(/)}";;
        'w') HMCWWW="${OPTARG}"
             HMCWWW="${HMCWWW%%+(/)}";;
        'M') MENUGEN="${TRUE}"
             MGPART="content.";;
        'v') VERBOSE="${TRUE}";;
        'V') VERYVERB="${TRUE}";;
        '?') usagemsg_processdata "${0}" && return 1 ;;
        ':') usagemsg_processdata "${0}" && return 1 ;;
        '#') usagemsg_processdata "${0}" && return 1 ;;
     esac
  done

  MGNTXT="DISABLED"
  (( MENUGEN == TRUE ))  && MGNTXT="ENABLED"
  (( MENUGEN == TRUE ))  && HTMLEXT="shtml"

  shift $(( ${OPTIND} - 1 ))

#### Verify the data and HTML directories.

  trap "usagemsg_processdata ${0}" EXIT
  if [[ ! -d "${HMCDATA}" ]]
  then
     processdata_print 2 "ERROR: Unable to access HMC data directory: ${HMCDATA}"
     return 2
  fi
  if [[ -d "${HMCWWW}" && ( ! -w "${HMCWWW}" || ! -x "${HMCWWW}" ) ]]
  then
     processdata_print 2 "ERROR: Unable to access HTML directory: ${HMCWWW}"
     return 3
  fi
  if [[ ! -d "${HMCWWW}" ]]
  then
     processdata_print 2 "WARNING: Unable to access HMC WWW directory,"
     processdata_print 2 "         ${HMCWWW} will be created."
  fi

#### Create the local directory in which to store the HTML reports.

  mkdir -p "${HMCWWW}"
  if ! cd "${HMCWWW}"
  then
     processdata_print 2 "# ERROR: unable to cd to ${HMCWWW}"
     return 4
  fi
```

Virtualization Standards for Business Continuity: Part 4

```
 trap "-" EXIT

#### Display some program and parameter info, if "VERBOSE" mode was specified.

 (( VERYVERB == TRUE )) && set -x
 (( VERBOSE  == TRUE )) && processdata_print 2 "# Version.......: ${VERSION}"
 (( VERBOSE  == TRUE )) && processdata_print 2 "# Data Directory.: ${HMCDATA}"
 (( VERBOSE  == TRUE )) && processdata_print 2 "# HTML Directory.: ${HMCWWW}"
 (( VERBOSE  == TRUE )) && processdata_print 2 "# Menugen status.: ${MGNTXT}"
 (( VERBOSE  == TRUE )) && processdata_print 2 "# HTML Extension.: ${HTMLEXT}"

################################################################

 (( VERBOSE  == TRUE )) && processdata_print 2 "# Removing Files.: ${HMCWWW}"

#### Remove any existing HTML report files.

 rm -f "${HMCWWW}/"*".${MGPART}${HTMLEXT}"
 rm -f "${HMCWWW}/menugen.dat"

 typeset -A HMCS
 typeset -A SYSTEMS
 typeset -A LPARS
 typeset -A PARMS
 typeset -A HEADER

#### HMC data files are assumed to have names of the form:
####     <SystemName>.<rOption>.<rsubtype>.<level>.out

 cd "${HMCDATA}"
 for FNAME in *.*.*.*.out
 do
  (( VERBOSE == TRUE )) && processdata_print 2 "\n# Retrieving Data: ${FNAME}"
  SYSNAME="${FNAME%%.*}"
  PARMNAME="${FNAME%.out}"
  PARMNAME="${PARMNAME#*.}"
  SYSTEMS[${SYSNAME}]="${SYSNAME}"

#### Build the "lssyscfg" command line options from the HMC data file name.

  IFS="."
  ARGS=( ${PARMNAME} )
  IFS=$' \t\n'
  [[ "_${ARGS[0]}" == "_null" ]] && ARGS[0]=""
  [[ "_${ARGS[1]}" == "_null" ]] && ARGS[1]=""
  [[ "_${ARGS[2]}" == "_null" ]] && ARGS[2]=""
  ARG="${ARGS[0]:+-r ${ARGS[0]}} ${ARGS[1]:+--rsubtype ${ARGS[1]}} ${ARGS[2]:+--level ${ARGS[2]}}"

#### Set a HEADER value to indicate the HTML header tags should be
#### output to the report file.

  HEADER[${SYSNAME}.${PARMNAME}]="${TRUE}"
  PARMS[${SYSNAME}.${PARMNAME}]="${ARG}"

  (( VERBOSE  == TRUE )) && processdata_print 2 "# System Name....: ${SYSNAME}"
  (( VERBOSE  == TRUE )) && processdata_print 2 "# Parameters.....: ${PARMNAME}"

################################################################

  ATTRFILE="/tmp/processData.${$}.tmp"
  unset ATTRVARS
  typeset -A ATTRVARS

#### Read each data line of the HMC output file and restructure them
#### into executable lines of code which define shell variables for
#### each HMC value. Store these executable lines of code in a temporary
#### file for later use.

  sed -e "s/\"feature_codes=.*\",/feature_codes=SeeHMC,/g" "${FNAME}" |
```

Virtualization Standards for Business Continuity: Part 4

```
    sed -e "s/,\"feature_codes=.*\"/feature_codes=SeeHMC/g" |
    sed -e "s/,\"backing_devices=\(.*\)\"/backing_devices=\"\1\"/g" |
    sed -e "s/^\([0-9]\)/_\1/g" |
    sed -e "s/,\([0-9]\)/,_\1/g" |
    sed -e "s/S\/N=/SerialNbr=/g" |
    sed -e "s/\".*\"//g;s/=/=\"/g;s/,,*/\";/g;s/$/\"/g" > "${ATTRFILE}"

#### Loop through each record line of the data file, which
#### will contain multiple variable/value pairs.

    while read -r -- ATTRLINE
    do
      [[ "_${ATTRLINE}" != _+([A-Za-z_])=* ]] && continue

      eval ${ATTRLINE}

      if [[ "_${lpar_name}" != "_" ]]
      then
        LPARS[${SYSNAME}.${lpar_name}]="${SYSNAME}.${lpar_name}"
        HEADER[${SYSNAME}.${lpar_name}.${PARMNAME}]="${TRUE}"
        if (( MENUGEN == FALSE ))
        then
          exec 3>"${HMCWWW}/${SYSNAME}.${lpar_name}.${MGPART}${HTMLEXT}"
          processdata_htmlbegin 3 "${lpar_name} - LPAR Level Information"
          processdata_tablebegin 3
          exec 3>&-
        fi
      fi

      if [[ "_${hmcname}" != "_" ]]
      then
        HMCS[${SYSNAME}.${hmcname}]="${SYSNAME}.${hmcname}"
      fi

#### The attribute line of data from the HMC output is now parsed into
#### the semi-colon delimited variable/value pairs.  Each variable name
#### is used as index of the associative array, and each value is used
#### as the value for this element of the associative array.

      IFS=";"
      for ATTR in ${ATTRLINE}
      do
        [[ "_${ATTR}" != _*=* ]] && continue
        ATTRVARS[${ATTR%%=*}]="${ATTR#*=}"
      done
      IFS=$' \t\n'

    done < "${ATTRFILE}"

#################################################################

#### Open a file to contain the system level HTML report information.

    exec 3>"${HMCWWW}/${SYSNAME}.${PARMNAME}.${MGPART}${HTMLEXT}"
    (( MENUGEN == FALSE )) && processdata_htmlbegin 3 "${SYSNAME} - System Level Information"

#### Output some heading information to the system level HTML report
#### file using the same file descriptor number used to open the file.

    processdata_titlebar 3 "Managed System Name: ${SYSNAME}" "HMC Command: lssyscfg ${ARG} -m ${SYSNAME}"
    processdata_tablebegin 3

    LCNT="0"
    while read -r -- ATTRLINE
    do

#### Verify the attribute data line from the HMC contains valid data
#### by checking to see if the line contains one or more alphabetic
#### characters or underscores followed by an equal sign.  If not,
#### continue to the next line of HMC data.
```

```
    if [[ "_${ATTRLINE}" != _+([A-Za-z_])=* ]]
    then
      (( VERBOSE == TRUE )) && processdata_print 2 "#         INVALID data line"
      continue
    else
      (( VERBOSE  == TRUE )) && processdata_print 2 "# Valid Data Line: $(( ++LCNT ))  $(( ++TCNT ))"
    fi

#### The attribute data line has been formatted to be an executable
#### statement that will define shell variables.  Execute the
#### attribute dataline using the "eval" command.

    eval ${ATTRLINE}

#### Output header information and the beginning of the attribute table
#### to the system level HTML report file.

    if (( ${HEADER[${SYSNAME}.${PARMNAME}]} == TRUE ))
    then
      processdata_rowbegin 3
      for ATTRVAR in "${!ATTRVARS[@]}"
      do
        processdata_tablehead 3 "${ATTRVAR}"
      done
      processdata_rowend 3
      HEADER[${SYSNAME}.${PARMNAME}]="${FALSE}"
    fi

#### Output data attribute information to the system level HTML report file.

    processdata_rowbegin 3
    for ATTRVAR in "${!ATTRVARS[@]}"
    do
      eval ATTRVAL="\"\${${ATTRVAR}}\""
      processdata_tablecell 3 "${ATTRVAL:- }"
    done
    processdata_rowend 3

#### Open another HMTL report file to contain the LPAR specific
#### data attribute information.

    if [[ "_${lpar_name}" != "_" ]]
    then
      exec 4>>"${HMCWWW}/${SYSNAME}.${lpar_name}.${MGPART}${HTMLEXT}"

      if (( ${HEADER[${SYSNAME}.${lpar_name}.${PARMNAME}]} == TRUE ))
      then
        processdata_tableend 4
        processdata_titlebar 4 "Managed System Name: ${SYSNAME}" "HMC Command: lssyscfg ${ARG} -m ${SYSNAME}"
        processdata_print 4 "\n<P><BLOCKQUOTE><H4>LPAR Name: ${lpar_name}</H4></BLOCKQUOTE></P>"
        processdata_tablebegin 4
        processdata_rowbegin 4
        for ATTRVAR in "${!ATTRVARS[@]}"
        do
          processdata_tablehead 4 "${ATTRVAR}"
        done
        processdata_rowend 4
        HEADER[${SYSNAME}.${lpar_name}.${PARMNAME}]="${FALSE}"
      fi

      processdata_rowbegin 4
      for ATTRVAR in "${!ATTRVARS[@]}"
      do
        eval ATTRVAL="\"\${${ATTRVAR}}\""
        processdata_tablecell 4 "${ATTRVAL:- }"
      done
      processdata_rowend 4
      exec 4>&-
```

Virtualization Standards for Business Continuity: Part 4

```
    fi

#### Unset all attribute variables so these values do not remain
#### when processing the next data line record from the attribute file.

    for ATTRVAR in "${!ATTRVARS[@]}"
    do
      unset ${ATTRVAR}
    done

  done < "${ATTRFILE}"

  rm -f "${ATTRFILE}"

#### Finalize and close the system level HTML report files.

  processdata_tableend 3
  processdata_footer 3
  (( MENUGEN == FALSE )) && processdata_htmlend 3
  exec 3>&-

  (( MENUGEN == TRUE )) && processdata_mgput 9 "${HMCWWW}" "${SYSNAME}" "${SYSNAME}.${PARMNAME}"
"${PARMNAME}"

 done

#### Finalize and close the LPAR specific HTML report files

 for PREFIX in "${LPARS[@]}"
 do
  (( VERBOSE == TRUE )) && processdata_print 2 "# Finalizing.....: ${PREFIX}"
  exec 3>>"${HMCWWW}/${PREFIX}.${MGPART}${HTMLEXT}"
    (( MENUGEN == FALSE )) && processdata_htmlbegin 3 "${PREFIX} - LPAR Level Information"
    processdata_tableend 3
    processdata_footer 3
    (( MENUGEN == FALSE )) && processdata_htmlend 3
  exec 3>&-
 done

################################################################

####  Create an HTML index file for each managed system referencing
#### managing systems, HMCs, LPARs, and "lssyscfg" parameters.

 for SYSNAME in "${SYSTEMS[@]}"
 do
  (( VERBOSE == TRUE )) && processdata_print 2 "# System Name....: ${SYSNAME}"
  exec 3>"${HMCWWW}/${SYSNAME}.${MGPART}${HTMLEXT}"
  (( MENUGEN == FALSE )) && processdata_htmlbegin 3 "${SYSNAME} - System Index"
  processdata_titlebar 3 "Managed System Name: ${SYSNAME}" "Resources listed by LPAR"
  processdata_tablebegin 3
  processdata_rowbegin 3
  processdata_tablehead 3 "System Name"
  processdata_tablehead 3 "LPAR Name"
  processdata_rowend 3
  for PREFIX in "${LPARS[@]}"
  do
    [[ "_${SYSNAME}" != "_${PREFIX%%.*}" ]] && continue
    LPARNAME="${PREFIX##*.}"
    processdata_rowbegin 3
    processdata_tablecell 3 "<A Href=\"${SYSNAME}.${HTMLEXT}\">${SYSNAME}</A>"
    processdata_tablecell 3 "<A Href=\"${PREFIX}.${HTMLEXT}\">${LPARNAME}</A>"
    processdata_rowend 3
    (( MENUGEN == TRUE )) && processdata_mgput 9 "${HMCWWW}" "${SYSNAME}" "${SYSNAME}.${LPARNAME}"
"${LPARNAME}"
  done
  processdata_tableend 3

  processdata_titlebar 3 "Managed System Name: ${SYSNAME}" "Resources listed by Frame"
  processdata_tablebegin 3
```

Virtualization Standards for Business Continuity: Part 4

```
   processdata_rowbegin 3
   processdata_tablehead 3 "System Name"
   processdata_tablehead 3 "lssyscfg Parameters"
   processdata_rowend 3

   for IDX in "${!PARMS[@]}"
   do
    [[ "_${SYSNAME}" != "_${IDX%%.*}" ]] && continue
    PARMNAME="${PARMS[${IDX}]}"
    processdata_rowbegin 3
    processdata_tablecell 3 "<A Href=\"${SYSNAME}.${HTMLEXT}\">${SYSNAME}</A>"
    processdata_tablecell 3 "<A Href=\"${IDX}.${HTMLEXT}\">${PARMNAME}</A>"
    processdata_rowend 3
   done
   processdata_tableend 3

   processdata_titlebar 3 "Other HMC Managed Systems" "Resources listed by HMC and Frame"
   processdata_tablebegin 3
   processdata_rowbegin 3
   processdata_tablehead 3 "HMC Name"
   processdata_tablehead 3 "Other Managed System Name"
   processdata_rowend 3

   for VAL in "${HMCS[@]}"
   do
    HMCNAME="${VAL##*.}"
    SYSNAME="${VAL%%.*}"
    processdata_rowbegin 3
    processdata_tablecell 3 "<A Href=\"${HMCNAME}.${HTMLEXT}\">${HMCNAME}</A>"
    processdata_tablecell 3 "<A Href=\"${SYSNAME}.${HTMLEXT}\">${SYSNAME}</A>"
    processdata_rowend 3
   done
   processdata_tableend 3
   processdata_footer 3
   (( MENUGEN == FALSE )) && processdata_htmlend 3

   exec 3>&-
  done

#################################################################

####  Create an HTML index file for each HMC.

  for HMCNAME in "${HMCS[@]##*.}"
  do
   (( VERBOSE == TRUE )) && processdata_print 2 "# HMC Name.......: ${HMCNAME}"
   exec 3>"${HMCWWW}/${HMCNAME}.${MGPART}${HTMLEXT}"

   (( MENUGEN == FALSE )) && processdata_htmlbegin 3 "${HMCNAME} - HMC Index"
   processdata_titlebar 3 "HMC Name: ${HMCNAME}" "HMC Managed Systems"
   processdata_tablebegin 3
   processdata_rowbegin 3
   processdata_tablehead 3 "HMC Name"
   processdata_tablehead 3 "System Name"
   processdata_rowend 3

   for VAL in "${HMCS[@]}"
   do
    SYSNAME="${VAL%%.*}"
    [[ "_${SYSNAME}.${HMCNAME}" != "_${VAL}" ]] && continue
    processdata_rowbegin 3
    processdata_tablecell 3 "${HMCNAME}"
    processdata_tablecell 3 "<A Href=\"${SYSNAME}.${HTMLEXT}\">${SYSNAME}</A>"
    processdata_rowend 3
   done
   processdata_tableend 3
   processdata_footer 3
   (( MENUGEN == FALSE )) && processdata_htmlend 3

   exec 3>&-
```

```
 done

#################################################################

#### Create an HTML directory index file referencing all HMCs and
#### managed systems.

 (( VERBOSE == TRUE )) && processdata_print 2 "# HTML Index.....: index.html"

 exec 3>"${HMCWWW}/index.${MGPART}${HTMLEXT}"

 (( MENUGEN == FALSE )) && processdata_htmlbegin 3 " "
 processdata_titlebar 3 "Frame and LPAR Resources" "Hardware Management Consoles"
 processdata_tablebegin 3
 processdata_rowbegin 3
 processdata_tablehead 3 "HMC Name"
 processdata_tablehead 3 "Managed System Name"
 processdata_rowend 3

 for VAL in "${HMCS[@]}"
 do
  HMCNAME="${VAL##*.}"
  SYSNAME="${VAL%%.*}"
  processdata_rowbegin 3
  processdata_tablecell 3 "<A Href=\"${HMCNAME}.${HTMLEXT}\">${HMCNAME}</A>"
  processdata_tablecell 3 "<A Href=\"${SYSNAME}.${HTMLEXT}\">${SYSNAME}</A>"
  processdata_rowend 3
  (( MENUGEN == TRUE )) && processdata_mgput 9 "${HMCWWW}" "${HMCNAME}" "${SYSNAME}" "${SYSNAME}"
  (( MENUGEN == TRUE )) && processdata_mgput 9 "${HMCWWW}" "index" "${HMCNAME}" "${HMCNAME}"
 done
 processdata_tableend 3
 processdata_footer 3
 (( MENUGEN == FALSE )) && processdata_htmlend 3
 exec 3>&-

 (( MENUGEN == TRUE )) && processdata_mgput 9 "${HMCWWW}" "NULL" "index" "HMC Data"
 return 0
}
#################################################################

processdata "${@}"
```

To view the user configurable options of the script “processdata.ksh”, execute the script with the “-?” option:

```
./processdata.ksh -?
```

The HTML based report pages generated by this script are, by default, stored in the directory “/tmp/htmldata”. This location is configurable by the command line option “-w”. If this script is executed on a web server, the report files can be written to a document directory where they can be presented by a web server. Otherwise, these files will have to be copied to the intended web server. When executing the script “processdata.ksh” in verbose mode (-v), it is common to see the message “***Invalid data line***”. This is not an error, it just means the frame was not configured for the parameters the script was working on.

One of the available options is “-M” and provides “menugen compatibility mode”. The script “menugen” is an automated web site generator which provides the ability to standardize the look-and-feel of all managed web pages. A full description of “menugen” is beyond the scope of this article, but

will be discussed in a later article.

The Business Continuity policies, guidelines, standards, and procedures to be learned from this article are as follows:

Policies:

- Automated document generation shall be included as a goal in all system administration efforts.

Guidelines:

- All HMC's should have remote access enabled for administration purposes and to aid in automation programming.

Standards:

- The script “processdata.ksh” provides a standard for how the HMC data will be gathered, assembled, and presented.

Procedures:

- The “processdata.ksh” shell script provides an automated procedure for gathering, assembling, and presenting data that identifies all physical and virtual adapters assigned to all VIO servers and client LPAR's.

The next article in this series will discuss procedures for allocating physical and virtual storage on the VIO server for use by the client LPAR's. The client LPAR's will utilize this storage as operating system boot disks, database and application storage.

Dana French
President, Mt Xia, Inc.
http://www.mtxia.com
615.556.0456