BladeLogic Server Automation/Extended Objects

|  |
| --- |
| **Contents**   [[hide](javascript:toggleToc())]   * [1 AIX Compliance](http://norris.bmc.com/wiki/Extended_Objects#AIX_Compliance) * [2 Windows Extended Objects](http://norris.bmc.com/wiki/Extended_Objects#Windows_Extended_Objects)   + [2.1 Extended Objects written using Microsoft's Scriptomatic](http://norris.bmc.com/wiki/Extended_Objects#Extended_Objects_written_using_Microsoft.27s_Scriptomatic) * [3 Database Extended Objects](http://norris.bmc.com/wiki/Extended_Objects#Database_Extended_Objects) * [4 Ports](http://norris.bmc.com/wiki/Extended_Objects#Ports)   + [4.1 Example 1](http://norris.bmc.com/wiki/Extended_Objects#Example_1)   + [4.2 Example 2](http://norris.bmc.com/wiki/Extended_Objects#Example_2) * [5 Process Queries](http://norris.bmc.com/wiki/Extended_Objects#Process_Queries) * [6 User Queries](http://norris.bmc.com/wiki/Extended_Objects#User_Queries) * [7 Disk Queries](http://norris.bmc.com/wiki/Extended_Objects#Disk_Queries) * [8 Log File Queries](http://norris.bmc.com/wiki/Extended_Objects#Log_File_Queries) * [9 Easy piped or multiple commands](http://norris.bmc.com/wiki/Extended_Objects#Easy_piped_or_multiple_commands) |

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=1)]AIX Compliance

Here's an EO Frank Lamprea wrote for an AIX ODM

*#!perl*

*# Script Name: AIX ODM Data Dump*

*# Version: 1.0*

*# Date: March 5, 2008*

*# Author: Frank Lamprea*

*# BladeLogic, Inc.*

*# Description:*

*# Retrieve ODM information via a NEXEC call to the target server*

*# Once the data is returned the structure is converted to INI format*

*# The code is derived from AIX::ODM - A Perl module for retrieving*

*# IBM AIX ODM information*

*# Extended Object:*

*# Mode: Central Execution*

*# Grammar: INI*

*# Command: perl <script> ??TARGET.HOST?? <C|P>*

*# The path to the script needs to be in non-NSH format*

*# Check Arguments*

my $numArgs = $*#ARGV + 1;*

if ($numArgs ne 2) {

print "Usage: <script.pl> hostname C|P ";

print "Where C retrieves Custom ODM Objects**\n**";

print "and P retrieves Predefined ODM Objects**\n**";

exit 1;

}

my $hostName = "$ARGV[0]";

my $mode = "$ARGV[1]";

**sub** odm\_classes {

my ${corp} = ${\_[0]}?${\_[0]}:'C';

my @classes;

my @devlist;

my $class;

my $devname;

my %dev;

*# Retrieve the list of classes from the ODM*

@classes = `nexec $hostName lsdev -${corp} -r class`;

foreach ${class} (@classes) {

chomp(${class});

*# Retrieve the list of devices associated with each class from the ODM*

@devlist = `nexec $hostName lsdev -Cc ${class} -F name`;

foreach ${devname} (@devlist) {

chomp(${devname});

${dev{${devname}}} = ${class};

}

}

return %dev;

};

*################################################################*

**sub** odm\_class {

my ${corp} = ${\_[0]}?${\_[0]}:'C';

return -1 if ( ${corp} ne 'C' );

return -1 if (!${\_[1]});

*# Retrieve the class of a device from the ODM*

my ${devclass} = `nexec $hostName lsdev -${corp} -r class -l ${\_[1]}`;

chomp(${devclass});

return ${devclass};

};

*################################################################*

**sub** odm\_subclass {

my ${corp} = ${\_[0]}?${\_[0]}:'C';

return -1 if ( ${corp} ne 'C' );

return -1 if (!${\_[1]});

*# Retrieve the subclass of a device from the ODM*

my ${devsub} = `nexec $hostName lsdev -${corp} -r subclass -l ${\_[1]}`;

chomp(${devsub});

return ${devsub};

};

*################################################################*

**sub** odm\_attributes {

my @{line};

my ${ndx};

my ${aname};

my %attrib;

*# Retrieve the attributes associated with the device from the ODM*

*# Two lines are returned, the attribute names are returned on the*

*# first line, the attribute values returned on the second.*

my @lines = `nexec $hostName lsattr -EOl ${\_[0]}`;

chomp(${lines[0]});

${lines[0]} =~ *s/^#//g*;

my (@attr\_name) = split(*/:/*,${lines[0]});

chomp(${lines[1]});

${lines[1]} =~ *s/^#//g*;

my (@attr\_valu) = split(*/:/*,${lines[1]});

${ndx} = 0;

foreach ${aname} (@attr\_name) {

${attrib{${aname}}} = ${attr\_valu[${ndx}]};

${ndx} = ${ndx} + 1;

}

return %{attrib};

};

*################################################################*

**sub** odm\_dump {

*# Create a hash of devices by their associated class*

my ${corp} = ${\_[0]}?${\_[0]}:'C';

my %devlist = &odm\_classes(${corp});

my %attrout;

my %devices;

my $ndx;

my $subndx;

foreach $ndx (keys %devlist) {

*# create a hash of attributes associated with each device*

%{attrout} = &odm\_attributes(${ndx});

*# Add a hash value for 'class' and 'devname'*

${devices{${ndx}}{'class'}} = ${devlist{${ndx}}};

${devices{${ndx}}{'subclass'}} = odm\_subclass(${corp},${ndx});

chomp(${devices{${ndx}}{'subclass'}});

${devices{${ndx}}{'devname'}} = $ndx;

foreach ${subndx} (keys %attrout) {

${devices{${ndx}}{${subndx}}} = ${attrout{${subndx}}};

}

}

return %devices;

}

*#### MAIN ####*

my %odm = odm\_dump("$mode");

while ( ($ndx1, $lev2) = each %odm ) {

print "[${ndx1}]**\n**";

while ( ($ndx2, $val) = each %$lev2 ) {

print "${ndx2}=${odm{${ndx1}}{${ndx2}}}**\n**";

}

}

exit 0;

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=2)]Windows Extended Objects

* [get local shares.nsh](http://norris.bmc.com/wiki/Get_local_shares.nsh) - Lists Windows shares for each server, including share level permissions and file system level permissions for each share
* [get\_net\_adapter\_config\_info.nsh](http://norris.bmc.com/wiki/Get_net_adapter_config_info.nsh) - Retrieves the configuration information for network adapter on a target host

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=3)]**Extended Objects written using Microsoft's Scriptomatic**

The following scripts were written with the help of [Microsoft's Scriptomatic](http://www.microsoft.com/downloads/details.aspx?FamilyID=09dfc342-648b-4119-b7eb-783b0f7d1178&displaylang=en). This is a fantastically useful tool that queries the machine to show all available WMI classes, and creates a VBS script for you to return the values.

Take a look at the following scripts. The only thing you need to do is to copy the vbs script to the section of the NSH script beginning with "### END OF MAIN SCRIPT", and edit it to make it output in the right format (usually name=value or Windows INI)

* [Domain Configuration.nsh](http://norris.bmc.com/wiki/Domain_Configuration.nsh) - Shows Windows Domain Configuration (or just Workgroup info if not in a Domain
* [Active Directory Users.nsh](http://norris.bmc.com/wiki/Active_Directory_Users.nsh) - Shows Active Directory Users when executed on a Domain Controller
* [BIOS Configuration.nsh](http://norris.bmc.com/wiki/BIOS_Configuration.nsh) - Shows Windows Server's BIOS Configuration
* [Memory Configuration.nsh](http://norris.bmc.com/wiki/Memory_Configuration.nsh) - Shows Windows Server Memory Configuration
* [Windows Configuration.nsh](http://norris.bmc.com/wiki/Windows_Configuration.nsh) - Shows Windows Configuration

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=4)]Database Extended Objects

* [Oracle Extended Objects](http://norris.bmc.com/wiki/ExtendedObjects_Oracle)

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=5)]Ports

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=6)]**Example 1**

netstat -an | grep -v Active | sed -e `s/^ \*//`

The output will be a simple name space multivalues grammar. (The example above applies to windows system but you can adapt this to UNIX as well changing the pattern in the grep -v)

To include process id for Windows:

netstat -ano

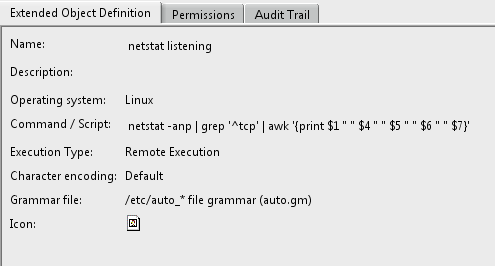
For Linux to include process id/name:

netstat -anp

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=7)]**Example 2**

netstat -anp | grep '^tcp' | awk '{print $1 " " $4 " " $5 " " $6 " " $7}'

User the auto.gm grammar to parse it. It returns both tcp and tcp6 along with To/From, status, and pid/service.

[](http://norris.bmc.com/wiki/File:EO-netstat_config.png)

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=8)]Process Queries

# Check the number of CFT processes running

"ps -ef | grep -i cft | sed -n '/grep/ !p' | awk '{print $1}'"

"ps -ef | grep -i cftmain | sed -n '/grep/ !p' | awk '{print $NF}'"

"ps -ef | awk '/CFTMAIN/ && !/awk/ {print $NF}'"

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=9)]User Queries

# Check to see if the cft user is in /etc/passwd

"RESULT=`grep -i cft /etc/passwd | awk '{print $1}'`; if [[ $result ]] ; then RESULT=PASS; else RESULT=FAIL; fi; echo $RESULT"

"RESULT=`grep -i cft /etc/passwd` ; if [[ $RESULT = "" ]] ; then RESULT=PASS; else RESULT=FAIL; fi; echo $RESULT"

"RESULT=`awk 'BEGIN { a=0 } /cft/ { a=++a } END { print a } ' /etc/passwd`; echo RESULT=$RESULT"

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=10)]Disk Queries

Total Disk Space

"df | awk '{print $2}' | grep '[0-9]' | awk '{sum = sum + $1} END {print sum}'"

Total Available Disk Space

"df | awk '{print $4}' | grep '[0-9]' | awk '{sum = sum + $1} END {print sum}'"

Get the total disk space for one mount on UNIX

"size=`df -k | grep VM\_INF | awk '{print $2}'` ; if [[ $size -gt 2000000 ]] ; then RESULT=PASS; else RESULT=FAIL; fi; echo $RESULT"

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=11)]Log File Queries

# Scan the last 200 lines of /var/log/messages

sh -c "tail -200 /var/log/messages | awk '{ printf \"%04d=%s\\n\",NR,$0 }'"

[[edit](http://norris.bmc.com/index.php?title=BladeLogic_Server_Automation/Extended_Objects&action=edit&section=12)]Easy piped or multiple commands

You can put anything you want inside of the doublequotes in a a 'nsh -c ""' block, as long as nsh knows how to do it.

1. Copy your script into /usr/nsh/share/extended\_objects/myscript.pl

2. Build a new extended object:

Command / Script: nsh -c "cp //blfs/usr/nsh/share/extended\_objects/myscript.pl //??TARGET.HOST??/tmp ; nexec ??TARGET.HOST?? perl /tmp/myscript.pl"

(.) Central Execution