Athena Cluster User's Guide

PRINTED BY Doug Lindsey

The Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Notice

This is a preliminary document and it may change substantially from time to time. The document is offered AS-IS.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information, including without limitation any scripts contained in this document presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation, except for the script contained in the Appendix to the document "ATHENAJOB.JS – JavaScript Example" which Microsoft grants you a limited license to reproduce, use and modify.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Microsoft and Microsoft Compute Cluster Server 2003 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Revision History

Date	Version	Description	Authors
19 Sept. 2006	0.1	Content sourced from Version 0.1.640, Revision 19 of "HPC Production User's Guide" by Ed Lassettre	Dougli, Edlas
21 November 2006	0.2	Updated text from LCA, Script edits	Dougli; RichCi
22 November 2006		Added RichCi updates	Dougli;Rich Ci
7 January 2007	0.3	Removed WebUI section pending completion	Dougli
27 January 2007	0.4	Added ftp support section	Dougli
23 February	0.5	Modified ftp transfer section for Microsoft ftp server	Dougli
18 May 2005	0.6	Updated support aliases, added MPI links, added Beta web UI section	Dougli

Table of Contents

1 Welcome to the HPC Production cluster!

The Microsoft High Performance Computing (HPC) Team[1] and the Microsoft Partners Solution Center (MPSC)[2] host the "Athena" cluster in the MPSC facility housed in building 25 on the Microsoft campus in Redmond, WA.

This cluster is sponsored by the HPC team and select MPSC partners, including AMD, Hitachi, HP, and Myricom. The system objective is successful execution of production jobs. The production goals of this cluster are:

- To provide a reference production cluster for analysis by the HPC team, running on the commercially-available version of Microsoft Compute Cluster Server 2003
- To gain production experience in the day-to-day administration & support of this cluster and leverage that experience as direct product feedback
- To provide compute cluster resources for the HPC Community

This manual includes descriptions of the hardware and software that comprise the cluster, the interfaces that can be used to access the cluster, how cluster scheduling works, and tips and techniques to ensure efficient utilization of the cluster. There are also separate sections about 'parameter sweeps,' .NET execution, running 'R' scripts and special application deployments. The final section is the do's and don'ts list—tips that will ensure cooperative and productive use of the cluster with all users.

2 Cluster Configuration

The Athena cluster is a shared, multi-user system based on Microsoft Windows Server 2003.

The current hardware configuration will grow over time, with the total number of available processors expected to double within the coming year.

2.1 Cluster Hardware Configuration

2.1.1 Servers

[1] See http://windowshpc.net/default.aspx for more details
[2] See http://www.microsoft.com/mpsc for more details

The cluster consists of (64) HP DL145 G2 (AMD Opteron-based) compute nodes, (1) HP DL145 G2 (AMD Opteron-based) head-node, and a supporting server infrastructure. Support servers are:

- web front-end: HP DL385 G1
- file server: HP DL145 G2
- monitoring management server: HP DL385 G1
- monitoring database server: HP Dl385 G1

Servers attached to the MPI network rely on Myrinet 10G PCI/E adapters for high-speed connectivity. All servers use built-in Ethernet ports for Gig-E connectivity.

The head node and each compute node have 8GB of RAM. Each supporting server has 4GB of RAM.

Remote keyboard/video management over TCP/IP is provided by 3 Raritan CommandCenter switches.

2.1.2 Network

The cluster network configuration consists of three networks: a public, private, and high-speed MPI network.

MPI network: Consists of (1) M3-CLOS-ENCL switch with Myrinet line cards. The compute nodes and file server are connected to this switch. This network is the high-speed application network used for fast communications and MPI support.

Public network: Consists of (3) 24-port Cisco 3560 Gig-E switches. All nodes and all support servers are connected to these switches. This network is used for system imaging and management from the MPSC operations team.

Private network: Consists of a Cisco 6504 Switch with (2) Cisco 6748 48-port line cards. The compute nodes and file server are connected to this switch. This network is used for backup and monitoring.

Clients do not access the cluster directly. Instead, clients interact with a bridgehead web server. The web server supports standard web protocols, and communicates with the cluster head node and compute nodes via the public network. Private and MPI networks have no bridging connectivity to the public network.

2.1.3 Storage

All compute nodes and the head node have 72GB of local SCSI storage on 10,000rpm disks. The file server is SAN-attached, and has 1.5 TB of live storage, provided by a Hitachi SAN.

Each user is provided with a file server storage sub-directory. Storage quotas are not currently enforced, but will be over time.

2.2 Cluster Software Configuration

The Athena cluster runs Microsoft Windows Compute Cluster Server 2003. The standard server images are based on Microsoft Windows Server 2003 Standard, x64 Edition.

There are currently no software packages installed directly on the compute nodes. The preferred method of software application usage is to run applications off of a share on a fileserver.

Applications that require administrator privilege to install will be supported by the system administrators (send mail to clustadm@microsoft.com), provided that the licensing model supports a multi-user cluster scenario.

The HPC team intends to support the following applications on the cluster in the near future:

- Microsoft Office 2007 – Excel Calculation Services
- Matlab
- R (see http://www.r-project.org/)
- WRF (http://www.wrf-model.org/index.php)

User-supplied applications that run side-by-side (or from within a user data storage location on a file server) are also permitted to run on the cluster.

3 Administrative Contacts

3.1 Helpdesk

Helpdesk support is via E-mail, and is available from 9am-6pm M-F. For Helpdesk issues pertaining specifically to the Athena environment, send mail to clustadm@microsoft.com

For issues, problems, or questions concerning application development or troubleshooting, please submit questions on the Windows HPC Community site. The forums are listed here: http://windowshpc.net/forums/default.aspx

3.2 Initial System Access or Software provisioning

Access to the Athena cluster requires the establishment of a User Agreement between Microsoft and the institutional sponsor of the user or users who will be accessing the cluster. For more information, contact clustadm@microsoft.com

Athena cluster users do not have administrative access to install applications. While we will soon have facilities for provisioning jobs, right now if you need software installed, please send a note to edlas@microsoft.com or dougli@microsoft.com. Licensed or server software requires special attention, please send a note to dougli@microsoft.com for these needs. While we can accommodate some server software on the head node, we would prefer that you host any servers (including license servers) on your own equipment.

4 System Maintenance and Upgrades

The head node, file server, and compute nodes are generally around the clock with the following exceptions:

1.) A 1-hour maintenance window is allocated each Wednesday afternoon at 12pm. During this time, the head node or file server may be rebooted.
2.) Emergency security patches for zero-day exploits, or hotfixes necessary to maintain system availability may be installed on an infrequent basis.
3.) Compute nodes may be upgraded individually at any time as they become available or idle.

In all instances, when maintenance and patching activities dictate bringing the systems offline, e-mail will be sent out to all cluster users prior to downtime.

For planned/routine Wednesday maintenance window outages, e-mail notification will be sent 1 week in advance. For emergency or system-availability affecting patches, mail will be sent at least 1 hour prior to downtime.

In order to prevent running jobs from being adversely affected by maintenance activities, compute nodes may be paused prior to downtime to permit running jobs to complete while not permitting new jobs in the queue to begin running. This process is known as "drain-stopping."
Job queue information has a default lifetime of 5 days following the completion of a job. Some maintenance activities – particularly those related to upgrades of the Compute Cluster Server software – may induce a reset of the job queue.

5 System Backups and Data Integrity

Data stored on the file server \\DL145G2066 is backed up nightly. Nightly backups are retained for a period of 30 days.

Upon termination of a user account, data will be retained on the file server for a period of 60 days. After that period of time, the data will be deleted.

The head node and compute nodes are not backed up regularly. If either of these fails in production, information in the job queue and data on running compute nodes may be lost.

Data and backups are retained on Hitachi SAN storage housed in the Microsoft Partner Solutions Center. At the present time, there is no provision for offsite backups.

A recommended best practice for users is to routinely offload and keep copies of important data.

6 Using the Cluster

Figure 1 below shows a logical representation of the connectivity to the cluster and file server from the IIS server on the Internet.

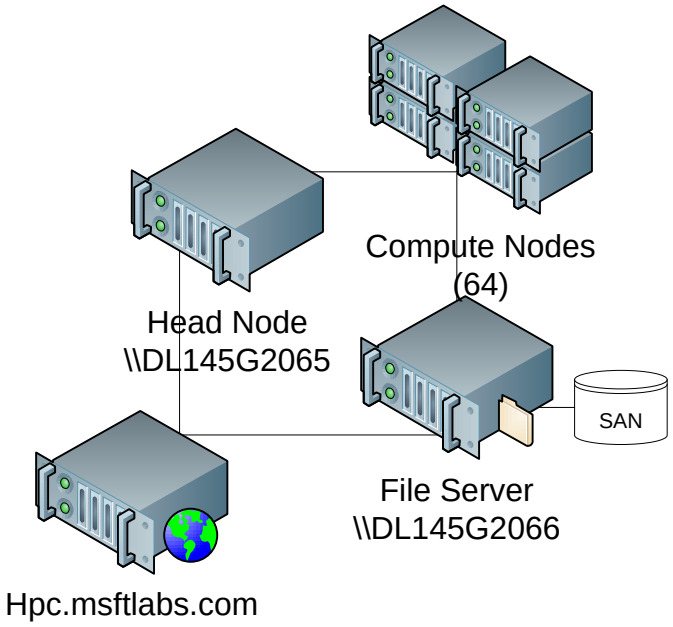


Figure 1- Logical Topology

6.1 Data Management

6.1.1 User Data Directory

A user data directory is provisioned by the system administrators when a user account request is fulfilled. By default, a user has access only to the directory that he/she has been provided.

The file server \\DL145G2066 supports the share \\DL145G2066\HPC . Underneath the \HPC share are partitions for user directories that begin with \USER . For example:

\\DL145G2066\HPC\USER
\HPC_Fred
\HPC_Jane
\HPC_Wendy

Each user directory is identified by the account name of the user for which the directory was created. In the above example, user HPC_Fred has full permission to add, delete, and modify files to the \HPC_Fred directory. HPC_Fred can also change the permissions on files in the \HPC_Fred directory to permit other users to access them.

Maintenance of permissions and data on user-owned subdirectories under the \USER share is the responsibility of each user. The system administrators do not actively manage content or access permissions within these directories once they have been assigned.

Any data file stored in a location other than the \\DL145G2066\HPC\USER\ directories is subject to automatic (automated) deletion.

Each user has a total storage quota of 30GB. When this storage limit has been exceeded, system administrators reserve the right to notify users that excess data must be migrated off of the file server.

6.1.2 Shared Data Directory

Users may occasionally want to share programs or data widely with other users or applications on the system. The directory \HPC\sharedapplications has been set aside for this purpose.

Use \sharedapplications with caution. Other users of the system have the ability to read and write, modify, or delete files that are stored in this directory. A best practice is to use this location only as temporary storage.

6.1.3 Compute Node Storage

Each compute node has 72GB of locally-attached disk storage. A percentage of that storage is dedicated to the operating system and default applications. The remainder of total storage is available for local data processing.

Upon job completion, no data should be left on the compute nodes. Automated processes will ensure that data on compute nodes is periodically deleted. Additionally, the compute nodes are not backed up. Any data left on a compute node will be lost during routine system maintenance.

6.1.4 Data Access

To upload and download files and applications, the Athena environment supports FTP.

The server has been tested with the Microsoft Windows Server and IPSwitch WS_FTP Professional ftp clients.[3] Support for other clients will be tested on an as-requested basis.

The hostname of the ftp server is: hpc.msftlabs.com

A sample text-based ftp session from a Windows ftp client is documented below as an example; actual interface details will be client-dependent:

```
C:\>ftp hpc.msftlabs.com
Connected to dl385g1104.msftlabs.com.
220-Microsoft FTP Service
220 ==AUTHORIZED ACCESS ONLY==
User (dl385g1104.msftlabs.com:(none)): msftlabs\hpc_joeuser
331 Password required for msftlabs\hpc_joeuser.
Password:
230 User msftlabs\hpc_joeuser logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
01-09-07  12:25PM       <DIR>          sharedapps
02-19-07  03:10PM       <DIR>          user
02-19-07  02:28PM       <DIR>          user2
226 Transfer complete.
ftp: 142 bytes received in 0.00Seconds 142000.00Kbytes/sec.
ftp> cd user
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
10-26-06  10:05AM       <DIR>          test
02-09-07  11:17AM       <DIR>          hpc_user1
09-19-06  09:51AM       <DIR>          hpc_user2
02-19-07  03:41PM       <DIR>          hpc_user3
```

[3] The FTP server software does not currently support SFTP or FTP-S (SSL-based) authentication setup. Support for authentication via SSL is planned for the future.

```
02-07-07  12:01PM       <DIR>          hpc_joeuser
11-01-06  06:29PM       <DIR>          hpc_user4
10-02-06  05:54PM       <DIR>          hpc_buffy
226 Transfer complete.
ftp: 1025 bytes received in 0.00Seconds 1025000.00Kbytes/sec.
ftp> cd hpc_joeuser
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
01-29-07  02:51PM                   12 joe.txt
226 Transfer complete.
ftp: 48 bytes received in 0.00Seconds 48000.00Kbytes/sec.
ftp> get joe.txt
200 PORT command successful.
150 Opening ASCII mode data connection for joe.txt(12 bytes).
226 Transfer complete.
ftp: 12 bytes received in 0.00Seconds 12000.00Kbytes/sec.
ftp> put joe_new.txt
200 PORT command successful.
150 Opening ASCII mode data connection for joe_new.txt.
226 Transfer complete.
ftp: 12 bytes sent in 0.00Seconds 12000.00Kbytes/sec.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
01-29-07  02:51PM                   12 joe.txt
02-23-07  10:29AM                   12 joe_new.txt
226 Transfer complete.
ftp: 100 bytes received in 0.00Seconds 100000.00Kbytes/sec.
ftp> quit
221
```

6.2 Debugging and Interactive/Simulation Steering

There are no user-level debugging facilities provided on the cluster. It is a multi-user production system and such facilities could adversely affect system availability.

The cluster topology has no support for visualization and simulation steering, although both are in principle possible.

6.3 User Interface

The Athena cluster is accessed via a web-service-based job submission and monitoring interface.

A Web-based user interface prototype is also available.

6.3.1 Web Services Job Submission and Monitoring Interface

The Athena cluster can be accessed through a web service. The web server name is https://hpc.msftlabs.com/ . This service currently provides the following functions:

- Job Submission
- Get Job's Definition
- Get Job's Status
- Cancel Job

A sample script that illustrates the commands supported by the web service is provided in section 9 of this document. The script supports a subset of the command line arguments supported by the JOB.VBS script that is included with the Microsoft Compute Cluster Server 2003 client tools.

6.3.1.1 General

The commands listed in the table below assume the use of the athenajob.js script detailed in section 9.

Command Line	**Description**
AthenaJob /? or /help	Outputs general help for all commands

6.3.1.2 Job Submission

The commands listed in the table below assume the use of the athenajob.js script detailed in section 9.

Command Line	**Description**
AthenaJob submit	Outputs general description for command
AthenaJob submit /? or /help	Outputs help for submitting job.
AthenaJob submit /jobfile:JSDLFile [/scheduler:uri] [/user:[domain\]username] [/password:{password|*}]	Submits job to Athena web service • /jobfile – Specifies the JSDL file that describes the job to submit • /scheduler – Optional. Specifies the URI of the Athena web service. If not specified, uses well known Athena URI. • /user – Optional. Specifies the username for job authentication and authorization. If not specified, prompts user for username. In prompt, user can just press ENTER to accept username specified in USERDOMAIN and USERNAME

	environment variables if present (i.e. Enter username (press ENTER if northamerica\richci):) • /password – Optional. Requires /username. Allows client to specify password for username. If not specified or '*' is specified, user is prompted for the password. Characters entered by the user are not echoed to the console window. If successful, returns ID of job.

The following is an example of a JSDL file that would specified by the /jobfile argument:

```
1  <jsdl:JobDefinition
2        xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
3        xmlns:jsdl- hpcp=http://schemas.ggf.org/jsdl/2006/07/jsdl-
4        hpcp
5        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6     <jsdl:JobDescription>
7        <jsdl:JobIdentification>
8           <jsdl:JobName>MyApp</jsdl:JobName>
9           <jsdl:JobProject>MyProject</jsdl:JobProject>
10       </jsdl:JobIdentification>
11       <jsdl:Application>
12          <jsdl-hpcp:HPCProfileApplication>
13             <jsdl-hpcp:Executable>
14                myapp.exe
15             </jsdl-hpcp:Executable>
16             <jsdl-hpcp:Argument>100</jsdl-hpcp:Argument>
17             <jsdl-hpcp:Input>
18                myapp_input.txt
19             </jsdl-hpcp:Input>
20             <jsdl-hpcp:Output>
21                myapp_output.txt
22             </jsdl-hpcp:Output>
23             <jsdl-hpcp:Error>
24                myapp_err.txt
25             </jsdl-hpcp:Error>
26             <jsdl-hpcp:WorkingDirectory>
27                \\DL145G2066\user\hpc_user
28             </jsdl-hpcp:WorkingDirectory>
29          </jsdl-hpcp:HPCProfileApplication>
30       </jsdl:Application>
31       <jsdl:Resources>
32          <jsdl:ExclusiveExecution>
33             False
34          </jsdl:ExclusiveExecution>
35          <jsdl:TotalCPUCount>
36             <jsdl:UpperBoundedRange>
37                1.0
```

```
38              </jsdl:UpperBoundedRange>
39              <jsdl:LowerBoundedRange>
40                 1.0
41              </jsdl:LowerBoundedRange>
42           </jsdl:TotalCPUCount>
43        </jsdl:Resources>
44     </jsdl:JobDescription>
45  </jsdl:JobDefinition>
```

Line(s)	Description
8	Name of the job
9	Project the job is associated with
14	Job's executable name. The executable must be located in the specified working directory.
16	Command line arguments that are passed to the executable. Multiple space-delimited arguments can be specified in one Argument element or individually in multiple Argument elements.
18	Name of the standard input file for the specified executable. The input file must be located in the specified working directory.
21	Name of the standard output file to be created by the specified executable. The output file is created in the specified working directory.
24	Name of the standard error file to be created by the specified executable. The error file is created in the specified working directory.
27	The job's working directory. Must be \\DL145G2066\user\{UserName} where {UserName} is the username specified in the WS-Security UsernameToken header (without the domain name).
33	Specifies whether the job requires exclusive access to the compute node(s) on which it runs.
37	The maximum number of processors that the job requires
40	The minimum number of processors that the job requires

6.3.1.3 Job Monitoring

The commands listed in the table below assume the use of the athenajob.js script detailed in section 9.

Command Line	Description
AthenaJob view	Outputs general description for command
AthenaJob view /? or /help	Outputs help for viewing job's description and status
AthenaJob view jobID [/scheduler:uri] [/jsdl]	Views job's status or definition • JobID – ID of job as returned from AthenaJob /submit • /scheduler – Optional. Specifies the URI of the Athena web service. If

	not specified, uses well known Athena URI. • /jsdl – Optional. Outputs job's entire JSDL document. If not specified, outputs status of job. If successful and /jsdl is not specified, example of job summary would be: Job ID : 3275 Status : Finished If /jsdl is specified, entire JSDL is returned.

6.3.1.4 Job Cancellation

The commands listed in the table below assume the use of the athenajob.js script detailed in section 9.

AthenaJob cancel	Outputs general description for command
AthenaJob cancel /? or /help	Outputs help for cancelling a job
AthenaJob cancel jobID [/scheduler:uri]	Cancel's specified job • JobID – ID of job as returned from AthenaJob /submit • /scheduler – Optional. Specifies the URI of the Athena web service. If not specified, uses well known Athena URI. Script outputs whether job was cancelled.

6.4 Programmatic Interface

Users can develop custom solutions that interface directly with the Athena cluster via Athena's web service. This section covers the message formats for each of the web service's operations. The web service's interface is based on a draft of OGSA HPC Basic Profile and is subject to change.

6.4.1 General

Web Service URL: https://hpc.msftlabs.com/Athena/JobSubmission.svc
Web Service WSDL: https://hpc.msftlabs.com/Athena/JobSubmission.svc?WSDL
SOAP Version: 1.1

6.4.2 Web Service Operations

6.4.2.1 CreateActivity

Submits a new job (a.k.a. activity) to the cluster

HTTP Header	Value
Content-Type	text/xml;charset=utf-8
SOAPAction	http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/CreateActivity

SOAP Header	Value
<Security>	A WS-Security <Security> header containing a WS-Security UsernameToken element

Request Message (SOAP Body)

```
<CreateActivity xmlns='http://schemas.ggf.org/bes/2006/08/bes-factory'>
    <ActivityDocument>
        <jsdl:JobDefinition
            xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
            xmlns:jsdl-hpcp="http://schemas.ggf.org/jsdl/2006/07/jsdl-hpcp"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <jsdl:JobDescription>
                <jsdl:JobIdentification>
                    <jsdl:JobName>MyApp</jsdl:JobName>
                    <jsdl:JobProject>MyProject</jsdl:JobProject>
                </jsdl:JobIdentification>
                <jsdl:Application>
                    <jsdl-hpcp:HPCProfileApplication>
                        <jsdl-hpcp:Executable>
                            myapp.exe
                        </jsdl-hpcp:Executable>
                        <jsdl-hpcp:Argument>100</jsdl-hpcp:Argument>
                        <jsdl-hpcp:Input>
                            myapp_input.txt
                        </jsdl-hpcp:Input>
                        <jsdl-hpcp:Output>
                            myapp_output.txt
                        </jsdl-hpcp:Output>
```

```
26                          <jsdl-hpcp:Error>
27                              myapp_err.txt
28                          </jsdl-hpcp:Error>
29                          <jsdl-hpcp:WorkingDirectory>
30                              \\DL145G2066\user\hpc_user
31                          </jsdl-hpcp:WorkingDirectory>
32                      </jsdl-hpcp:HPCProfileApplication>
33                  </jsdl:Application>
34                  <jsdl:Resources>
35                      <jsdl:ExclusiveExecution>
36                          False
37                      </jsdl:ExclusiveExecution>
38                      <jsdl:TotalCPUCount>
39                          <jsdl:UpperBoundedRange>
40                              1.0
41                          </jsdl:UpperBoundedRange>
42                          <jsdl:LowerBoundedRange>
43                              1.0
44                          </jsdl:LowerBoundedRange>
45                      </jsdl:TotalCPUCount>
46                  </jsdl:Resources>
47              </jsdl:JobDescription>
48          </jsdl:JobDefinition>
49      </ActivityDocument>
50  </CreateActivity>
```

Line(s)	Description
4-50	JSDL document describing job to execute
11	Name of the job
12	Project the job is associated with
17	Job's executable name. The executable must be located in the specified working directory.
19	Command line arguments that are passed to the executable. Multiple space-delimited arguments can be specified in one Argument element or individually in multiple Argument elements.
21	Name of the standard input file for the specified executable. The input file must be located in the specified working directory.
24	Name of the standard output file to be created by the specified executable. The output file is created in the specified working directory.
27	Name of the standard error file to be created by the specified executable. The error file is created in the specified working directory.
30	The job's working directory. Must be \\DL145G2066\user\{UserName} where {UserName} is the username specified in the WS-Security UsernameToken header (without the domain name).
36	Specifies whether the job requires exclusive access to the compute node(s) on which it runs.
40	The maximum number of processors that the job requires
43	The minimum number of processors that the job requires

Response Message (SOAP Body)

```
1  <CreateActivityResponse
2  xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
```

```
3       <ActivityIdentifier>
4           <Address xmlns="http://www.w3.org/2005/08/addressing">
5               https://hpc.msftlabs.com/Athena/JobSubmission.svc
6           </Address>
7           <ReferenceParameters
8   xmlns="http://www.w3.org/2005/08/addressing">
9               <JobID xmlns="http://tempuri.org/">1437</JobID>
10              <TaskID xmlns="http://tempuri.org/">1</TaskID>
11          </ReferenceParameters>
12      </ActivityIdentifier>
13  </CreateActivityResponse>
```

Range	Description
3-12	The activity identifier for the job as a WS-Addressing EndpointReference. The activity identifier can be passed to other operations to get information about the job or terminate the job.

6.4.2.2 GetActivityStatuses

HTTP Header	Value
Content-Type	text/xml;charset=utf-8
SOAPAction	http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/GetActivityStatuses

SOAP Header	Value
<Security>	A WS-Security <Security> header containing a WS-Security UsernameToken element

Request Message (SOAP Body)

```
1   <GetActivityStatuses
2   xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
3       <ActivityIdentifier>
4           <Address xmlns="http://www.w3.org/2005/08/addressing">
5               https://hpc.msftlabs.com/Athena/JobSubmission.svc
6           </Address>
7           <ReferenceParameters
8   xmlns="http://www.w3.org/2005/08/addressing">
9               <JobID xmlns="http://tempuri.org/">1437</JobID>
10              <TaskID xmlns="http://tempuri.org/">1</TaskID>
11          </ReferenceParameters>
12      </ActivityIdentifier>
13  </GetActivityStatuses>
```

Range	Description
3-12	One or more activity identifiers, as returned from CreateActivity, that specify the job or jobs whose status is requested

Response Message (SOAP Body)

```
1  <GetActivityStatusesResponse
2  xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
3      <Response>
4          <ActivityIdentifier>
5              <Address xmlns="http://www.w3.org/2005/08/addressing">
6                  https://hpc.msftlabs.com/Athena/JobSubmission.svc
7              </Address>
8              <ReferenceParameters
9  xmlns="http://www.w3.org/2005/08/addressing">
10                 <JobID>1435</JobID>
11                 <TaskID>1</TaskID>
12             </ReferenceParameters>
13         </ActivityIdentifier>
14         <ActivityStatus>
15             <Finished>
16                 <Finished xmlns=""/>
17             </Finished>
18         </ActivityStatus>
19     </Response>
20 </GetActivityStatusesResponse>
```

Range	Description
3-19	One or more responses based on the number of activity IDs passed in the request message
4-13	Activity identifier of the job whose status is returned
14-18	The status of the job. Status could be: Pending, Running, Finished, Cancelled, Failed.

6.4.2.3 GetActivityDocuments

HTTP Header	Value
Content-Type	text/xml;charset=utf-8
SOAPAction	http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/GetActivityDocuments

SOAP Header	Value
<Security>	A WS-Security <Security> header containing a WS-Security UsernameToken element

Request Message (SOAP Body)

```
1  <GetActivityDocuments
2  xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
3      <ActivityIdentifier>
4          <Address xmlns="http://www.w3.org/2005/08/addressing">
5              https://hpc.msftlabs.com/Athena/JobSubmission.svc
6          </Address>
7          <ReferenceParameters
8  xmlns="http://www.w3.org/2005/08/addressing">
9              <JobID xmlns="http://tempuri.org/">1437</JobID>
```

```
10              <TaskID xmlns="http://tempuri.org/">1</TaskID>
11          </ReferenceParameters>
12      </ActivityIdentifier>
13  </GetActivityDocuments>
```

Range	Description
3-12	One or more activity identifiers, as returned from CreateActivity, that specify the job or jobs whose JSDL document is requested

Response Message (SOAP Body)

```
1   <GetActivityDocumentsResponse
2   xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
3       <Response>
4           <ActivityIdentifier>
5               <Address xmlns="http://www.w3.org/2005/08/addressing">
6                   https://hpc.msftlabs.com/Athena/JobSubmission.svc
7               </Address>
8               <ReferenceParameters
9   xmlns="http://www.w3.org/2005/08/addressing">
10                  <JobID>1435</JobID>
11                  <TaskID>1</TaskID>
12              </ReferenceParameters>
13          </ActivityIdentifier>
14          {JSDL document}
15      </Response>
16  </GetActivityDocumentsResponse>
```

Range	Description
3-15	One or more responses based on the number of activity IDs passed as input
4-13	Activity identifier of the job whose status is returned
14	The job's JSDL document. See CreateActivity for details on the JSDL document.

6.4.2.4 TerminateActivityStatuses

HTTP Header	Value
Content-Type	text/xml;charset=utf-8
SOAPAction	http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/TerminateActivities

SOAP Header	Value
<Security>	A WS-Security <Security> header containing a WS-Security UsernameToken element

Request Message (SOAP Body)

```
1   <TerminateActivities
2   xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
```

```
3       <ActivityIdentifier>
4           <Address xmlns="http://www.w3.org/2005/08/addressing">
5               https://hpc.msftlabs.com/Athena/JobSubmission.svc
6           </Address>
7           <ReferenceParameters
8   xmlns="http://www.w3.org/2005/08/addressing">
9               <JobID xmlns="http://tempuri.org/">1437</JobID>
10              <TaskID xmlns="http://tempuri.org/">1</TaskID>
11          </ReferenceParameters>
12      </ActivityIdentifier>
13  </TerminateActivities>
```

Range	Description
3-12	One or more activity identifiers, as returned from CreateActivity, that specify the job or jobs to be terminated

Response Message (SOAP Body)

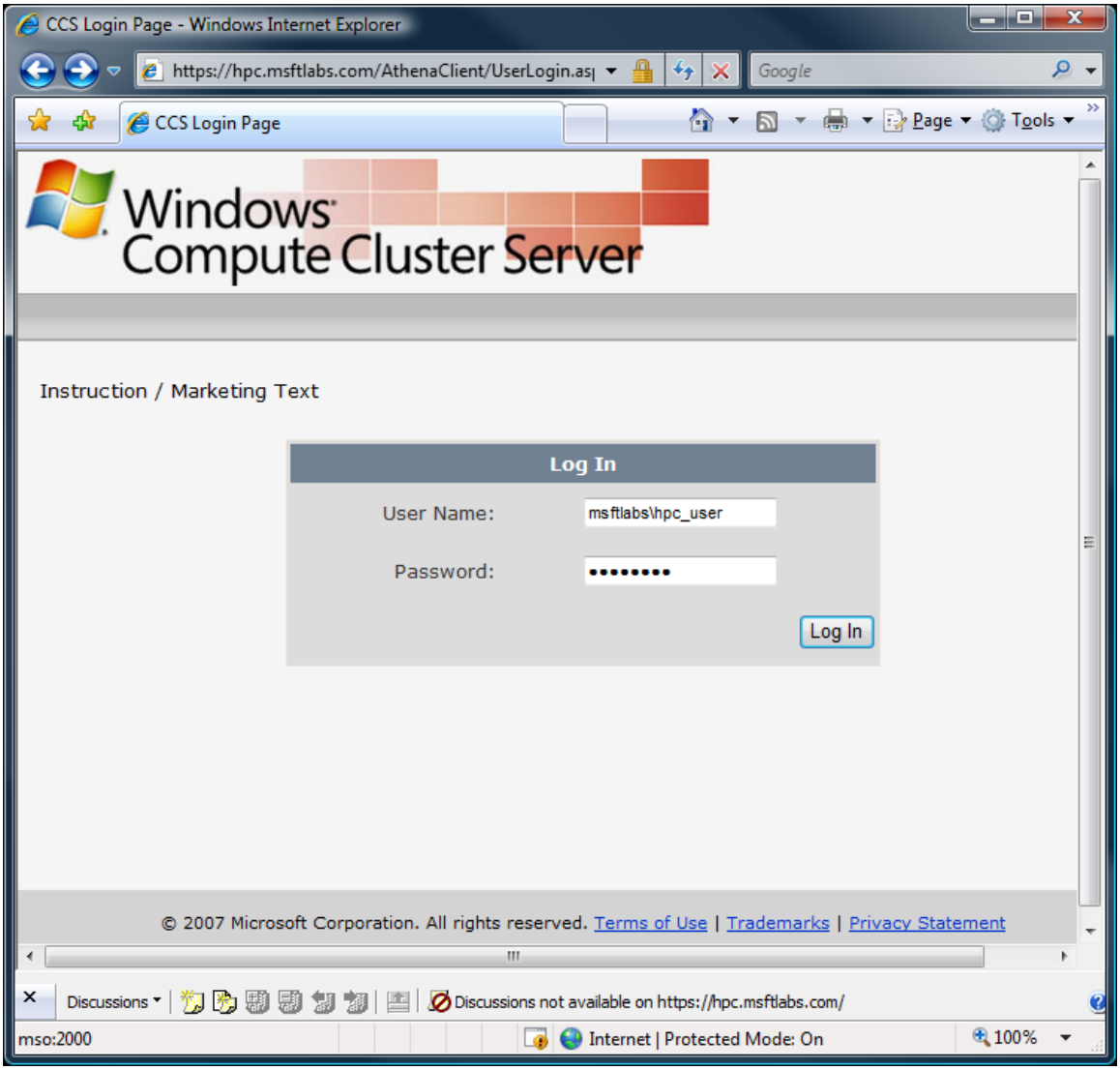
```
1   <TerminateActivitiesResponse
2   xmlns="http://schemas.ggf.org/bes/2006/08/bes-factory">
3       <Response>
4           <ActivityIdentifier>
5               <Address xmlns="http://www.w3.org/2005/08/addressing">
6                   https://hpc.msftlabs.com/Athena/JobSubmission.svc
7               </Address>
8               <ReferenceParameters
9   xmlns="http://www.w3.org/2005/08/addressing">
10                  <JobID>1435</JobID>
11                  <TaskID>1</TaskID>
12              </ReferenceParameters>
13          </ActivityIdentifier>
14          <Cancelled>false</Cancelled>
15      </Response>
16  </TerminateActivitiesResponse>
```

Range	Description
3-15	One or more responses based on the number of activity IDs passed as input
4-13	Activity identifier of the job whose status is returned
14	Whether the job was cancelled

6.5 Web UI

As of May 2007, a web-based UI prototype is available at https://hpc.msftlabs.com/AthenaClient . Please send all feedback, bug reports, etc. to clustadm@microsoft.com

When first accessing the Web UI, you will be prompted for logon credentials:



Enter credentials as shown.

Once logged in, the user interface should be relatively intuitive to use.

Note: As a prototype service, the UI is subject to change and to periods of unavailability. If you are unable to access the Web UI, please try submitting jobs via the method described in sections 6.2-6.4 before escalating.

6.6 MPI Jobs

A very good resource for understanding how MPI works and how to use it can be found on Microsoft Technet: “Using Microsoft Message Passing Interface”

http://technet2.microsoft.com/windowsserver/en/library/4cb68e33-024b-4677-af36-28a1ebe9368f1033.mspx?mfr=true

MSMPI is based on the MPICH2 standard. A tutorial on getting started with MSMPI is here: http://windowshpc.net/files/4/getting_started/entry441.aspx

A complete list of MPIEXEC arguments can be found here:
http://technet2.microsoft.com/windowsserver/en/library/7876c216-b704-473c-b97f-e8a15c67551b1033.mspx

The Microsoft Compute Cluster Server 2003 SDK can be found here:
http://www.microsoft.com/downloads/details.aspx?FamilyID=d8462378-2f68-409d-9cb3-02312bc23bfd&DisplayLang=en

7 Guidelines

The following sections define common cluster terminology and best practices.

7.1 Tasks

A task is a fundamental unit of execution on the Microsoft Compute Cluster Server 2003 platform. Tasks represent the combination of an executable program or script and the environmental variables necessary to support the execution of that program or script.

7.1.1 Task Types

Tasks are either serial or parallel. A task is serial if it is not parallel.

Parallel tasks are concurrent executions of the same program, with high speed interconnections between the program instances. Parallel tasks require additional support for startup, linking instances together, localized naming, and communication. Microsoft Compute Cluster Server 2003 cluster relies on the MSMPI stack for this support. The Microsoft Compute Cluster Server 2003 User Operations Manual is the best source of information about MSMPI. For general guidelines when running parallel tasks, see the subheading called “Parallel tasks” for each of the headings that follow.

By default, each task is assigned to one processor. Requesting more than one processor per task provides no guarantee that all the processors will be on the same node. This can lead to unexpected execution behaviors, although the results will be correct.

7.1.2 Dependent Tasks

In some instances, users may wish to control the order in which tasks are executed. For example, task #2 should not begin until after task #1 has been launched. These are tasks where the execution sequence is significant. A focal task depends on the execution of other tasks. The Compute Cluster Server 2003 User Operations Manual is the best source of information about inter-task dependencies.

7.1.3 Parameter Sweeps and Cluster Performance

Parameter sweeps are repeated executions of a program with different input parameters. These are a form of serial application. These are often called embarrassingly parallel applications because the order of execution doesn't matter. As a general rule, cluster efficiency is greatest when each input parameter set is submitted as a job with a single task. Unless the task is a parallel task, only one processor should be requested. Likewise, cluster efficiency is significantly reduced when the parameter sets are batched, either by submitting a job with one task for each parameter set, or by submitting a single job with a single task that internally plows through the parameter set.

For this reason, **it is strongly recommended that parameter sweep jobs be architected as one job per parameter set, one task per job**. In addition, unless an application is SMP-capable (can fully utilize multiple processors if available), the "isExclusive" parameters at both the job and task level should be set as Isexclusive =False. Following these practices helps to ensure full utilization of the cluster resources.

7.2 The Processors Resource

Processors are a logical representation of a resource allocated by the job scheduler that loosely corresponds to physical processors. This is a convenient way of scheduling more than one job or task to a node without overloading the node. However, this allocation is not enforced, so the execution character of the job determines processor usage.

The processor resource request is expressed by the MinProcessors and MaxProcessors attributes of the job and task context descriptions. The default values are MinProcessors=MaxProcessors=1. There are very few situations where this is incorrect – that is, either MinProcessors or MaxProcessors is too small.

7.2.1 Parameter Sweeps

For parameter sweeps, a separate job should be launched for each case, using the default values. This practice gives the best blend of rapid turnaround time and cluster utilization.

7.2.2 Parallel tasks

For jobs with MPI tasks, the situation is different. Here the goal is to specify the desired rank of the MPI task – the number of copies running in parallel. A best practice is to set MaxProcessors=MinProcessors = *rank* at the job level and to repeat that specification for the MPI task itself. If there is more than one MPI task in the job, then the job-level MinProcessors and MaxProcessors values should be set the maximum rank, with the task-level settings matching the desired rank for the task.

7.2.3 Jobs in Which Sequence of Task Execution is Important

For jobs in which one task must not start until another task has been started or has completed, the sequencing should be specified through the Depend attribute of the focal tasks. Please refer to the Microsoft Compute Cluster Server 2003 User Operation Manual for a detailed explanation of dependent task scheduling. The shortest job turnaround will occur when there are enough processors for the maximum in degree of any focal task. If this is mysterious, don't adjust the number of processors beyond what is needed for task execution.

7.3 IsExclusive Parameter

The IsExclusive parameter determines whether or not several jobs or tasks can share a node. At the job level, the default value is **true**, primarily to simplify scheduling engineering tasks. At the task level, the default value is **false**.

7.3.1 Parameter Sweeps

For parameter sweeps, the job and task levels, the value of the "isExclusive" parameter should be **false**.

7.3.2 Parallel Tasks

For parallel tasks, the job and task levels, the value of the "isExclusive" parameter should be **true**.

7.4 Time Estimates

All time estimates are wall clock times. It is possible to submit jobs with no time constraints. The only penalty incurred for not specifying time constraints is that the job is not a backfill candidate. Jobs that have run more than a week are subject to review, and a history of runaway jobs will trigger administrative action.

The time required for each task is accumulated. When this sum exceeds the job-level time estimate, the job is failed. Time estimation for parameter sweeps can be difficult, especially when the calculations are Monte Carlo simulations. In these cases, the task time estimate is a measure of user impatience.

7.5 Checkpoint/Restart

Tasks that will require more than one hour to complete should be capable of restarting from a checkpoint and should take checkpoints at least once each half hour.

7.6 IsRerunnable Parameter

When this parameter is **true**, when a node fails, the job scheduler will rerun it, assuming that the application will either restart from the beginning or has established a checkpoint and can restart from it. This parameter should be set to **false** unless there is a likelihood that the job will complete successfully if re-run.

7.7 Backfill Parameter

When this parameter is true, the job can be scheduled as a backfill job. Backfill opportunities occur when a high priority job is waiting for job resources. At that moment, some but not all of the resources are available. Under these circumstances, the job scheduler will compute an expected time when the needed resources are available. This defines a backfill window, and jobs that would complete during this window will be scheduled for execution if they are backfill capable.

7.8 Run until canceled Parameter

It is possible to retain job resources even after a job has completed execution. To take advantage of this feature, select the **run until canceled** option before all existing tasks have completed execution.

7.9 How to Improve and Optimize Job Turnaround Time

The job scheduler is at once very powerful and very basic. It avoids deadlock by never over-committing resources, by allocating all needed resources prior to launching a job, and by holding those resources until job execution is complete. In addition, job aging avoids starvation – every job will eventually be executed, even under continuously heavy loading. For efficiency, it will allocate up to the maximum amount of resources requested. These simple rules can result in unintended consequences.

Using the default values for job and task parameters will usually result in correct execution. Changing some of the parameter values can improve turnaround time. For example, increasing the rank of an MPI task, with a matching Min/MaxProcessors value will often speed up the calculation being performed. Specifying IsExclusive=**true** at the same time will eliminate interference from other jobs on the same node.

There are some parameter modifications that do not improve turnaround time at all. For example, the 'R' statistical interpretation system is a single-threaded application. Running this task on a multiprocessing system uses only one of the available processors. Requesting more than one processor has no effect on turnaround time, but it can increase the time the task is on the job queue, either because the required resources aren't available for the job, or because the resources are being used by other job tasks.

In the same vein, when more than one processor is requested for a task, there is no guarantee that the 'allocated' processors are on the same compute node. This is because the value of MinProcessors (or MaxProcessors) at the task level is, in reality, a specification of the desired rank of a parallel (MPI-style) task. These tasks have a startup sequence that is distinctly different from the familiar 'main' startup, so that the parallel copies of the application are properly started and terminated. A standard 'main' startup sequence results in starting only one copy of the application, so any other nodes that are allocated to the task go unused.

In the early day of internal production cluster hosting on Microsoft Compute Cluster Server 2003, one user of a cluster submitted a job with an 'R' task, requesting 3 processors for the task. Since the cluster had two processors per compute node, a minimum of two nodes were allocated for the job. In this case, a number of jobs, each with one 'R' task, were submitted, so that in some cases three nodes were assigned. The user was quite surprised to discover that there was no improvement in the job turnaround time.

Another user, who routinely created multi-task parameter sweeps, was in the habit of submitting jobs with as many tasks as the cluster had processors. When the system was lightly loaded, the job scheduler would allocate all available resources to these jobs. However, since the task execution times were unequal, it was often the case that all of the tasks except one would be complete. The job scheduler wouldn't schedule another job until this lone task completed. The resulting job turnaround time was essentially equal to the execution time of this lone task. Additional jobs (some submitted by this same user) were delayed. A different approach to parameter sweeps would have improved the turnaround time per case, increased the number of cases that could be examined per unit of time, improved cluster efficiency and (because the user had submitted more than one job like this), improved the turnaround time of the task ensemble.

Another user took a different approach, scheduling jobs with a smaller number of tasks and limiting the resources so that some resources were available for other work. Since the number of tasks was quite large, when this user submitted several jobs at once, the first one was fully was assigned almost all the available resources, so the second one was given a much smaller share of the resources and put into execution. The result was once again that the turnaround time for the total task ensemble was much higher than expected.

Yet another user submitted parameter sweeps with a large number of processors, but this time increased the MinProcessors value to be about half the cluster capacity. Once again, if multiple jobs were submitted, the first job blocked the execution of the others. Since

the requisite resources weren't available until the first job finished, this blocking behavior continued throughout the execution of the job set, even though the intention was to have the jobs share the cluster, running two jobs at once.

Based on the above observed behaviors and consequences, it can be concluded that the MinProcessors and MaxProcessors values work best when used to specify the rank of a parallel computation. When used with purely serial jobs like parameter sweeps, low (single digit) values for MinProcessors and MaxProcessors at the job level will improve turnaround. Enthusiastic use of large values for either parameter will rarely produce the desired results.

8 More Information

For more information on Compute Cluster Server 2003, see the HPC team's web portal http://windowshpc.net/

Product documentation is available via the Microsoft Computer Cluster Server 2003 client software and utilities. From a computer on which the client utilities have been installed, go to

Start->All Programs->Microsoft Compute Cluster Pack->Compute Cluster User's Guide.

The documentation is also available from within the "Compute Cluster Pack" folder on the Microsoft Computer Cluster Server 2003 installation CD.

9 Appendix: ATHENAJOB.JS – Javascript Example

The following script can be used to access the cluster.

```
//
//  Change Uri to reference remote cluster as necessary
//
var clusterUri = "https://hpc.msftlabs.com/Athena/JobSubmission.svc";

//
//  SOAP Actions
//
var SOAPAction_SubmitJob = "http://schemas.ggf.org/bes/2006/08/bes-
factory/BESFactoryPortType/CreateActivity";
var SOAPAction_GetJobs = "http://schemas.ggf.org/bes/2006/08/bes-
factory/BESFactoryPortType/GetActivityDocuments";
var SOAPAction_GetJobStatuses =
"http://schemas.ggf.org/bes/2006/08/bes-
factory/BESFactoryPortType/GetActivityStatuses";
var SOAPAction_CancelJob = "http://schemas.ggf.org/bes/2006/08/bes-
factory/BESFactoryPortType/TerminateActivities";

//
//  Script Commands
//
var ScriptCommand_Submit = "SUBMIT";
var ScriptCommand_View = "VIEW";
var ScriptCommand_Cancel = "CANCEL";

//
//  Optional Argument Names
//
var ScriptArg_Help1 = "?";
var ScriptArg_Help2 = "HELP";
var ScriptArg_JSDL = "JSDL";

//
//  Optional Param Argument Names
//
var ScriptArg_JobFile = "JOBFILE";
var ScriptArg_Scheduler = "SCHEDULER";
var ScriptArg_User = "USER";
var ScriptArg_Password = "PASSWORD";

//
//  Error Codes
//
var ErrorCode_FileNotFound = -2146828235;

//
//  Web Service Error Responses
//
```

```
var ErrorResponse_AccessDenied = "At least one security token in the
message could not be validated.";

//
//  Event Log Event Types
//
var EventLog_Error = 1;

//
//  Shell object for error logging and env vars
//
var wshShell = new ActiveXObject("WScript.Shell");

//
//  file system object for file I/O
//
var fso = new ActiveXObject("Scripting.FileSystemObject");

main();

function main()
{
    var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    var filename;
    var jobID;
    var username;
    var defaultUsername;
    var password;
    var command;

    //
    //  If we didnt get enough arguments, print directions and exit
    //
    if ((WScript.Arguments.length == 0) ||
            CompareArguments(WScript.Arguments(0), ScriptArg_Help1) ||
            CompareArguments(WScript.Arguments(0), ScriptArg_Help2))
    {
        PrintDirections();
        return;
    }

    //
    //  Get command from commandline
    //
    command = WScript.Arguments(0).toUpperCase();

    //
    //  Update cluster URI
    //
    UpdateClusterURI();

    //
    //  Prepare web service proxy
    //
    SetupProxy(xmlHttp);

    //
```

```
//  Execute the specified command
//
if(command == ScriptCommand_Submit)
{
    //
    //  Check arguments
    //

    if (WScript.Arguments.length == 1)
    {
        PrintSubmitDirections();
        return;
    }

    else if(ArgumentExists(ScriptArg_Help1) ||
            ArgumentExists(ScriptArg_Help2))
    {
        PrintSubmitDirectionsVerbose();
        return;
    }

    //
    //  Get filename and validate
    //

    filename = GetArgumentValue(ScriptArg_JobFile);

    if(filename == null)
    {
        PrintSubmitDirectionsVerbose();
        return;
    }

    if(!fso.FileExists(filename))
    {
        WScript.Echo("File, " + filename + ", does not exist.");
        return;
    }

    //
    //  Get user credentials
    //

    username = GetUserName();
    password = GetPassword();

    //
    //  Submit job
    //
    SubmitJob(xmlHttp, filename, username, password);
}
else if(command == ScriptCommand_View)
{
    //
    //  Check arguments
    //
```

```
        if (WScript.Arguments.length == 1)
        {
            PrintViewDirections();
            return;
        }

        else if(ArgumentExists(ScriptArg_Help1) ||
                ArgumentExists(ScriptArg_Help2))
        {
            PrintViewDirectionsVerbose();
            return;
        }

        //
        //  Get the job ID
        //

        jobID = WScript.Arguments(1);

        if(isNaN(parseInt(jobID)))
        {
            WScript.Echo("Invalid job identifier \"" + jobID + "\"");
            return;
        }

        //
        //  Get user credentials
        //

        username = GetUserName();
        password = GetPassword();

        //
        //  If /jsdl is specified
        //
        if(ArgumentExists(ScriptArg_JSDL))
        {
            //
            //  Get the job's JSDL
            //

            var jsdl = GetJobJSDL(xmlHttp, jobID, username, password);

            if(jsdl == null)
                return;

            WScript.Echo("Job ID               : " + jobID);
            WScript.Echo("JSDL                 : " + jsdl);
        }
        else
        {
            //
            //  Otherwise just return status
            //

            var jobStatus = GetJobStatus(xmlHttp, jobID, username,
password);
```

```
            if(jobStatus == null)
                return;

            var responseBody = xmlHttp.responseXML;

            WScript.Echo("Job ID                : " + jobID);
            WScript.Echo("Status                : " + jobStatus);
        }
    }
    else if(command == ScriptCommand_Cancel)
    {
        //
        //  Check arguments
        //

        if (WScript.Arguments.length == 1)
        {
            PrintCancelDirections();
            return;
        }

        else if(ArgumentExists(ScriptArg_Help1) ||
                ArgumentExists(ScriptArg_Help2))
        {
            PrintCancelDirectionsVerbose();
            return;
        }

        jobID = WScript.Arguments(1);

        if(isNaN(parseInt(jobID)))
        {
            WScript.Echo("Invalid job identifier \"" + jobID + "\"");
            return;
        }

        //
        //  Get user credentials
        //

        username = GetUserName();
        password = GetPassword();

        //
        //  Cancel job
        //
        CancelJob(xmlHttp, jobID, username, password);
    }
    else
    {
        //
        //  Display error if operation is unknown
        //
        WScript.Echo("Unknown job operation \"" + WScript.Arguments(0)
+ "\"");
        return;
```

```
    }
}

//
//  Setup proxy for web service calls
//
function SetupProxy(xmlHttp)
{
    xmlHttp.open("POST", clusterUri, false);
    xmlHttp.setRequestHeader("Connection", "Keep-Alive");
    xmlHttp.setRequestHeader("Content-Type", "text/xml;charset=utf-8");
}

//
//  Submits job to cluster
//
function SubmitJob(xmlHttp, filename, username, password)
{
    try
    {
        var request = GetSubmitJobRequest(filename, username,
password);

        xmlHttp.setRequestHeader("SOAPAction", SOAPAction_SubmitJob);
        xmlHttp.send(request);

        if(xmlHttp.statusText == "OK")
        {
            var jobID = GetSubmitJobResponse(xmlHttp.responseXML,
username, password);

            WScript.Echo("Job Submitted. Job ID = " + jobID + ".");
        }
        else
        {
            if(-1 !=
xmlHttp.responseText.indexOf(ErrorResponse_AccessDenied))
            {
                WScript.Echo("Access Denied. Please check Event Viewer
for more details.");

                LogWSErrorEvent(SOAPAction_SubmitJob,
xmlHttp.responseText);
            }
            else
            {
                WScript.Echo("Unexpected error occurred. Please check
Event Viewer for more details.");

                LogWSErrorEvent(SOAPAction_SubmitJob,
xmlHttp.responseText);
            }
        }
    }

    catch(e)
    {
```

```
        if(e.number == ErrorCode_FileNotFound)
        {
            WScript.Echo("File, " + filename + ", not found.");
        }
        else
        {
            WScript.Echo("Unexpected error occurred. Please check Event
Viewer for more details.");

            LogErrorEvent(SOAPAction_SubmitJob, e);
        }
    }
}

//
//  Create a request message for submitting a job
//
function GetSubmitJobRequest(filename, username, password)
{
    var msg = "<s:Envelope
xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'>" +
                "<s:Header>" +
                    "<o:Security s:mustUnderstand='1'
xmlns:o='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd'>" +
                        "<o:UsernameToken>" +
                            "<o:Username>%USERNAME%</o:Username>" +
                            "<o:Password>%PASSWORD%</o:Password>" +
                        "</o:UsernameToken>" +
                    "</o:Security>" +
                "</s:Header>" +
                "<s:Body xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>" +
                    "<CreateActivity
xmlns='http://schemas.ggf.org/bes/2006/08/bes-factory'>" +
                        "<ActivityDocument>%JSDL%</ActivityDocument>" +
                    "</CreateActivity>" +
                "</s:Body>" +
            "</s:Envelope>";

    var re = /%USERNAME%/g;
    msg = msg.replace(re, username);

    re = /%PASSWORD%/g;
    msg = msg.replace(re, password);

    var jsdl = ReadJSDLFile(filename);
    re = /%JSDL%/g;
    msg = msg.replace(re, jsdl);

    return msg;
}

//
//  Gets the jobID from response message
//
function GetSubmitJobResponse(responseXML)
```

```
{
    var nodeList = responseXML.getElementsByTagName("JobID");
    var jobID = nodeList[0].text;

    if(nodeList.length == 0)
        return null;

    return jobID;
}

//
//  Read JSDL file from disk
//
function ReadJSDLFile(filename)
{
    var ForReading = 1, ForWriting = 2, ForAppending = 8;
    var TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0;

    var ts = fso.OpenTextFile(filename, ForReading, false,
TristateUseDefault);
    var jsdl = ts.ReadAll( );

    ts.Close( );

    return jsdl;
}

//
//  Gets job's JSDL document
//
function GetJobJSDL(xmlHttp, jobID, username, password)
{
    var jsdl = null;

    try
    {
        var request = GetGetJobRequest(jobID, username, password);

        xmlHttp.setRequestHeader("SOAPAction", SOAPAction_GetJobs);
        xmlHttp.send(request);

        if(xmlHttp.statusText == "OK")
        {
            jsdl = GetGetJobResponse(xmlHttp.responseXML);
        }
        else
        {
            if(-1 !=
xmlHttp.responseText.indexOf(ErrorResponse_AccessDenied))
            {
                WScript.Echo("Access Denied. Please check Event Viewer
for more details.");

                LogWSErrorEvent(SOAPAction_GetJobs,
xmlHttp.responseText);
            }
            else
```

```
            {
                WScript.Echo("Unexpected error occurred. Please check
Event Viewer for more details.");

                LogWSErrorEvent(SOAPAction_GetJobs,
xmlHttp.responseText);
            }
        }
    }

    catch(e)
    {
        WScript.Echo("Unexpected error occurred. Please check Event
Viewer for more details.");

        LogErrorEvent(SOAPAction_GetJobs, e);
    }

    return jsdl;
}

//
//  Creates request message for getting job's JSDL
//
function GetGetJobRequest(jobID, username, password)
{
    var msg = "<s:Envelope
xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'>" +
                "<s:Header>" +
                    "<o:Security s:mustUnderstand='1'
xmlns:o='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd'>" +
                        "<o:UsernameToken>" +
                            "<o:Username>%USERNAME%</o:Username>" +
                            "<o:Password>%PASSWORD%</o:Password>" +
                        "</o:UsernameToken>" +
                    "</o:Security>" +
                "</s:Header>" +
                "<s:Body xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>" +
                    "<GetActivityDocuments
xmlns='http://schemas.ggf.org/bes/2006/08/bes-factory'>" +
                        "<ActivityIdentifier>" +
                            "<Address
xmlns='http://www.w3.org/2005/08/addressing'>%JOB_URI%</Address>" +
                            "<ReferenceParameters
xmlns='http://www.w3.org/2005/08/addressing'>" +
                                "<JobID>%JOB_ID%</JobID>" +
                                "<TaskID>1</TaskID>" +
                            "</ReferenceParameters>" +
                        "</ActivityIdentifier>" +
                    "</GetActivityDocuments>" +
                "</s:Body>" +
            "</s:Envelope>";

    var re = /%USERNAME%/g;
    msg = msg.replace(re, username);
```

```
    re = /%PASSWORD%/g;
    msg = msg.replace(re, password);

    re = /%JOB_URI%/g;
    msg = msg.replace(re, clusterUri);

    re = /%JOB_ID%/g;
    msg = msg.replace(re, jobID);

    return msg;
}

//
//  Gets JSDL from response message
//
function GetGetJobResponse(responseXML)
{
    var nodeList = responseXML.getElementsByTagName("JobDefinition");

    if(nodeList.length == 0)
        return null;

    return nodeList[0].xml;
}

//
//  Gets the specified job's status
//
function GetJobStatus(xmlHttp, jobID, username, password)
{
    var jobStatus = null;

    try
    {
        var request = GetGetJobStatusRequest(jobID, username,
password);

        xmlHttp.setRequestHeader("SOAPAction",
SOAPAction_GetJobStatuses);
        xmlHttp.send(request);

        if(xmlHttp.statusText == "OK")
        {
            jobStatus = GetGetJobStatusResponse(xmlHttp.responseXML);

            if(jobStatus == null)
            {
                WScript.Echo("No job returned.");
            }
        }
        else
        {
            if(-1 !=
xmlHttp.responseText.indexOf(ErrorResponse_AccessDenied))
            {
```

```
                WScript.Echo("Access Denied. Please check Event Viewer
for more details.");

                LogWSErrorEvent(SOAPAction_GetJobStatuses,
xmlHttp.responseText);
            }
            else
            {
                WScript.Echo("Unexpected error occurred. Please check
Event Viewer for more details.");

                LogWSErrorEvent(SOAPAction_GetJobStatuses,
xmlHttp.responseText);
            }
        }
    }

    catch(e)
    {
        WScript.Echo("Unexpected error occurred. Please check Event
Viewer for more details.");

        LogErrorEvent(SOAPAction_GetJobStatuses, e);
    }

    return jobStatus;
}

//
//  Create a request message to get the job's status
//
function GetGetJobStatusRequest(jobID, username, password)
{
    var msg = "<s:Envelope
xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'>" +
                "<s:Header>" +
                    "<o:Security s:mustUnderstand='1'
xmlns:o='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd'>" +
                        "<o:UsernameToken>" +
                            "<o:Username>%USERNAME%</o:Username>" +
                            "<o:Password>%PASSWORD%</o:Password>" +
                        "</o:UsernameToken>" +
                    "</o:Security>" +
                "</s:Header>" +
                "<s:Body xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>" +
                    "<GetActivityStatuses
xmlns='http://schemas.ggf.org/bes/2006/08/bes-factory'>" +
                        "<ActivityIdentifier>" +
                            "<Address
xmlns='http://www.w3.org/2005/08/addressing'>%JOB_URI%</Address>" +
                            "<ReferenceParameters
xmlns='http://www.w3.org/2005/08/addressing'>" +
                                "<JobID>%JOB_ID%</JobID>" +
                                "<TaskID>1</TaskID>" +
                            "</ReferenceParameters>" +
```

```
                        "</ActivityIdentifier>" +
                    "</GetActivityStatuses>" +
                "</s:Body>" +
            "</s:Envelope>";

    var re = /%USERNAME%/g;
    msg = msg.replace(re, username);

    re = /%PASSWORD%/g;
    msg = msg.replace(re, password);

    re = /%JOB_URI%/g;
    msg = msg.replace(re, clusterUri);

    re = /%JOB_ID%/g;
    msg = msg.replace(re, jobID);

    return msg;
}

//
//  Gets the job's status from response message
//
function GetGetJobStatusResponse(responseXML)
{
    var nodeList = responseXML.getElementsByTagName("ActivityStatus");

    if(nodeList.length == 0)
        return null;

    var statusElement = nodeList[0].firstChild;

    return statusElement.nodeName;
}

//
//  Cancels specified job
//
function CancelJob(xmlHttp, jobID, username, password)
{
    try
    {
        var request = GetCancelJobRequest(jobID, username, password);

        xmlHttp.setRequestHeader("SOAPAction", SOAPAction_CancelJob);
        xmlHttp.send(request);

        if(xmlHttp.statusText == "OK")
        {
            var cancelled = GetCancelJobResponse(xmlHttp.responseXML);

            if(cancelled != null)
            {
                if(cancelled.toLowerCase() == "true")
                {
                    WScript.Echo("Job " + jobID + " is cancelled.");
                }
```

```
                else
                {
                    WScript.Echo("Job " + jobID + " is not
cancelled.");
                }
            }
            else
            {
                WScript.Echo("No job returned.");
            }
        }
        else
        {
            if(-1 !=
xmlHttp.responseText.indexOf(ErrorResponse_AccessDenied))
            {
                WScript.Echo("Access Denied. Please check Event Viewer
for more details.");

                LogWSErrorEvent(SOAPAction_CancelJob,
xmlHttp.responseText);
            }
            else
            {
                WScript.Echo("Unexpected error occurred. Please check
Event Viewer for more details.");

                LogWSErrorEvent(SOAPAction_CancelJob,
xmlHttp.responseText);
            }
        }
    }

    catch(e)
    {
        WScript.Echo("Unexpected error occurred. Please check Event
Viewer for more details.");

        LogErrorEvent(SOAPAction_CancelJob, e);
    }
}

//
//  Creates a request message for cancelling the job
//
function GetCancelJobRequest(jobID, username, password)
{
    var msg = "<s:Envelope
xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'>" +
                "<s:Header>" +
                    "<o:Security s:mustUnderstand='1'
xmlns:o='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd'>" +
                        "<o:UsernameToken>" +
                            "<o:Username>%USERNAME%</o:Username>" +
                            "<o:Password>%PASSWORD%</o:Password>" +
                        "</o:UsernameToken>" +
```

```
                    "</o:Security>" +
                "</s:Header>" +
                "<s:Body xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>" +
                    "<TerminateActivities
xmlns='http://schemas.ggf.org/bes/2006/08/bes-factory'>" +
                        "<ActivityIdentifier>" +
                            "<Address
xmlns='http://www.w3.org/2005/08/addressing'>%JOB_URI%</Address>" +
                            "<ReferenceParameters
xmlns='http://www.w3.org/2005/08/addressing'>" +
                                "<JobID>%JOB_ID%</JobID>" +
                                "<TaskID>1</TaskID>" +
                            "</ReferenceParameters>" +
                        "</ActivityIdentifier>" +
                    "</TerminateActivities>" +
                "</s:Body>" +
            "</s:Envelope>";

    var re = /%USERNAME%/g;
    msg = msg.replace(re, username);

    re = /%PASSWORD%/g;
    msg = msg.replace(re, password);

    re = /%JOB_URI%/g;
    msg = msg.replace(re, clusterUri);

    re = /%JOB_ID%/g;
    msg = msg.replace(re, jobID);

    return msg;
}

//
//  Gets results of CancelJob operation from response message
//
function GetCancelJobResponse(responseXML)
{
    var nodeList = responseXML.getElementsByTagName("Cancelled");

    if(nodeList.length == 0)
        return null;

    return nodeList[0].text;
}

//
//  Prints general directions
//
function PrintDirections()
{
    WScript.Echo("USAGE:");
    WScript.Echo("    AthenaJob {operator} [options] [arguments]");
    WScript.Echo("\n");
    WScript.Echo("where");
    WScript.Echo("    operator:");
```

```
    WScript.Echo("        submit    - Submit job for execution.");
    WScript.Echo("        view      - View job details.");
    WScript.Echo("        cancel    - Terminate a job.");
    WScript.Echo();
    WScript.Echo("options:");
    WScript.Echo("       /? or /help - Display this help message");
    WScript.Echo();
}

//
//  Prints submit job directions
//
function PrintSubmitDirections()
{
    WScript.Echo("Job submit should contain switch jobfile.");
}

//
//  Prints verbose submit job directions
//
function PrintSubmitDirectionsVerbose()
{
    WScript.Echo("USAGE:");
    WScript.Echo("    athenajob submit /jobfile:jobfile
[/scheduler:uri]");
    WScript.Echo("    \t[/user:[domainname\\]username]
[/password:{password|*}]");
    WScript.Echo();
    WScript.Echo("where");
    WScript.Echo("    options:");
    WScript.Echo("        /? or /help - Display this help message.");
    WScript.Echo("        /jobfile    - Load job terms from the
specified job file.");
    WScript.Echo("        /password   - Password of the specified
user.");
    WScript.Echo("        /scheduler  - Scheduler URI. Default is the
Athena scheduler URI.");
    WScript.Echo("        /user       - Job user name.");
    WScript.Echo();
    WScript.Echo("Submit job by specified job file.");
}

//
//  Prints view job directions
//
function PrintViewDirections()
{
    WScript.Echo("Job view should only contain job identifier and
switches.");
}

//
//  Prints verbose view job directions
//
function PrintViewDirectionsVerbose()
{
    WScript.Echo("USAGE:");
```

```
    WScript.Echo("    athenajob view jobID [/scheduler:uri]")
    WScript.Echo("    \t[/user:[domainname\\]username]
[/password:{password|*}]");
    WScript.Echo();
    WScript.Echo("where");
    WScript.Echo("    jobID           - Identifier of the job to
view.");
    WScript.Echo();
    WScript.Echo("    options:");
    WScript.Echo("        /? or /help - Display this help message.");
    WScript.Echo("        /jsdl       - Return entire JSDL document.");
    WScript.Echo("        /password   - Password of the specified
user.");
    WScript.Echo("        /scheduler  - Scheduler URI. Default is the
Athena scheduler URI.");
    WScript.Echo("        /user       - Job user name.");
    WScript.Echo();
    WScript.Echo("View details of the specified job.");
}

//
//  Prints cancel directions
//
function PrintCancelDirections()
{
    WScript.Echo("Job cancel should only contain job identifier and
switches.");
}

//
//  Prints verbose cancel job directions
//
function PrintCancelDirectionsVerbose()
{
    WScript.Echo("USAGE:");
    WScript.Echo("    athenajob cancel jobID [/scheduler:uri]")
    WScript.Echo("    \t[/user:[domainname\\]username]
[/password:{password|*}]");
    WScript.Echo();
    WScript.Echo("where");
    WScript.Echo("    jobID           - Identifier of the job to
view.");
    WScript.Echo();
    WScript.Echo("    options:");
    WScript.Echo("        /? or /help - Display this help message.");
    WScript.Echo("        /password   - Password of the specified
user.");
    WScript.Echo("        /scheduler  - Scheduler URI. Default is the
Athena scheduler URI.");
    WScript.Echo("        /user       - Job user name.");
    WScript.Echo();
    WScript.Echo("Cancel the specified job.");
}

//
//  Error log helper (action and exception)
//
```

```
function LogErrorEvent(action, e)
{
    var errorMsg = "AthenaJob.js logged the following error:\r\n" +
                "\tSOAP Action = " + action + "\r\n" +
                "\tError Number = " + e.number + "\r\n" +
                "\tError Description = " + e.description;

    wshShell.LogEvent(EventLog_Error, errorMsg);
}

//
//  Error log helper (action and response message)
//
function LogWSErrorEvent(action, response)
{
    var errorMsg = "AthenaJob.js logged the following web service
error:\r\n" +
                "\tSOAP Action = " + action + "\r\n" +
                "\tResponse = " + response;

    wshShell.LogEvent(EventLog_Error, errorMsg);
}

//
//  Retrieves current username and domain
//
function GetUserName()
{
    //
    //  Get default user name from env variables
    //
    envVars = wshShell.Environment("PROCESS");
    defaultUsername = envVars("USERDOMAIN") + "\\" +
envVars("USERNAME");

    //
    //  Prompt for username if it wasnt passed in cmdline
    //
    if(WScript.Arguments.Named.Exists(ScriptArg_User))
    {
        username = WScript.Arguments.Named(ScriptArg_User);
    }
    else
    {
        username = PromptForUserName(defaultUsername);
    }

    return username;
}

//
//  Get user's for password from arguments or prompt
//
function GetPassword()
{
    //
    //  Prompt for password if it wasnt passed in cmdline
```

```
    //
    if(WScript.Arguments.Named.Exists(ScriptArg_Password))
    {
        password = WScript.Arguments.Named(ScriptArg_Password);

        if(password == "*")
        {
            password = PromptForPassword();
        }
    }
    else
    {
        password = PromptForPassword();
    }

    return password;
}

//
//  Prompt user for username
//
function PromptForUserName(defaultUsername)
{
    WScript.Echo("Enter UserName (Hit ENTER if " + defaultUsername +
"):");
    var username = WScript.StdIn.ReadLine();
    if(username.length == 0)
    {
        username = defaultUsername;
    }
    return username;
}

//
//  Update cluster's URI if /scheuduler cmd line param is specified
//
function UpdateClusterURI()
{
    if(WScript.Arguments.Named.Exists(ScriptArg_Scheduler))
    {
        clusterUri = WScript.Arguments.Named(ScriptArg_Scheduler);
    }
}

//
//  Primpt for password
//
function PromptForPassword()
{
    var pwdPrompt = new ActiveXObject("ScriptPW.Password");
    WScript.Echo("Enter Password:");
    return pwdPrompt.GetPassword();
}

//
//  Compares 2 arguments
//
```

```
function CompareArguments(cmdlineArg, arg)
{
    cmdlineArg = cmdlineArg.toUpperCase();
    arg = arg.toUpperCase();

    if(cmdlineArg == arg)
    {
        return true;
    }
    else if(cmdlineArg == "-" + arg)
    {
        return true;
    }
    else if(cmdlineArg == "/" + arg)
    {
        return true;
    }
    else
    {
        return false;
    }
}

//
//  Returns whether argument exists
//
function ArgumentExists(arg)
{
    var exists = false;
    var i = 0;

    for(i=0;i<WScript.Arguments.length;i++)
    {
        if(WScript.Arguments.Named.Exists(arg))
        {
            exists = true;
            break;
        }
    }

    return exists;
}

//
//  Get specified arguments value
//
function GetArgumentValue(arg)
{
    var argVal = null;

    if(WScript.Arguments.Named.Exists(arg))
    {
        argVal = WScript.Arguments.Named(arg);
    }

    return argVal;
}
```