WBEM/WMI Provider Data Sheet
Utilization Provider

Provider overview

This data sheet provides basic information about the Utilization WBEM/WMI provider. This provider is implemented using the Pegasus C++ provider API on Unix and OpenVMS systems, and Windows Management Instrumentation (WMI) APIs on Windows. The intended audience of this document is client/provider developers and end-users who want to use this provider.

Description

The Utilization Provider (UP) provides CPU, memory, disk and network utilization data. It is comprised of a daemon/service (utild on Unix and OpenVMS, UPService on Windows) that wakes up every 5 minutes and records data to either /var/adm/util (Unix) or \Documents and Settings\All Users\Application Data\Hewlett-Packard\UtilProvider (Windows) or /SYS$SPECIFIC/WBEMPROVIDERS/util (OpenVMS). Up to 30 days worth of data is stored on the system, and on a typical installation this will be less than 5 Megabytes worth of data.

The WBEM/WMI Provider simply provides access to this utilization data. The UP classes described in this document are registered in the root/cimv2 namespace.

Requirements

The UP is implemented and tested on the following Operating System versions:

- HP-UX 11i v1 (11.11)
- HP-UX 11i v2 (11.23)
- Microsoft Windows Server 2003
- Linux Red Hat Enterprise Linux ES Version 4 (RHEL4)
- SUSE LINUX Enterprise Server 9 (SLES9)
- HP OpenVMS IA64 V8.4 (scheduled April 2009)

Release history

- HP-UX (v1&v2): AR and OEUR media (December, 2005)
- Windows: HP Integrity Smart Setup 4.5 (July, 2006)
- Linux RHEL4: HP Integrity Essentials Foundation Pack for Linux 1.01 (July, 2006)
- Linux SLES9: HP Integrity Essentials Foundation Pack for Linux 1.01 (August, 2006)
- HP I64VMS WBEMPROVIDERS V1.7-16 (May 2009)
- HP I64VMS WBEMPROVIDERS V2.0-4 (June 2010)
- HP I64VMS WBEMPROVIDERS V2.1-4 (August 2010)
- HP I64VMS WBEMPROVIDERS V2.2-3 (February 2011)

Setting up this provider

Installing this provider

- **HP-UX**: The bundle name for UP is UtilProvider. The shared library for UP will be installed at /opt/util/lib/libUtilizationProvider.1, and a symbolic link to this library will be created at /opt/wbem/providers/lib (libUtilizationProvider.sl for PA platform and libUtilizationProvider.so for IPF platform). The data collection daemon is /usr/lbin/utild, and is launched via an entry in /etc/inittab.
- **Linux**: The rpm names for UP are hp-utilprovider-<version>.sles9.ia64.rpm and hp-utilprovider-<version>.rhel4.ia64.rpm. The shared library for UP will be installed at /opt/util/lib/libUtilizationProvider.1, and a symbolic link to this library will be created at /opt/tog-pegasus/providers/lib/libUtilizationProvider.so. The data collection daemon is /usr/sbin/utild, and is launched via an entry in /etc/inittab.

- **Windows**: The install package name for UP is WMIUtilProvider64.msi. The WMI provider is implemented as a decoupled provider, and runs as a Windows service named 'HP UtilProvider WMI Provider' and the executable is located at \Program Files\Hewlett-Packard\UtilProvider\bin\WMIUtilProvider.exe. The data collection service is named 'HP UtilProvider Data Collector' and the executable is located at \Program Files\Hewlett-Packard\UtilProvider\bin\UPService.exe. Both of these services are started automatically by the Windows Services infrastructure, and will respawn automatically if necessary.
- **OpenVMS**: The Utilization provider is bundled in the V1.7-16 and later versions of the HP I64VMS WBEMPROVIDERS product. . The shareable image library is installed at SYS$COMMON:[SYSLIB]LIBUTILIZATIONPROVIDER.EXE by the PRODUCT INSTALL. The data collection server image is installed at SYS$COMMON:[WBEMPROVIDERS.BIN] WBEMPROVIDERS$UTILD.EXE and is started by the procedure WBEMPROVIDERS$RUN_OPENVMS_UTILD.COM in the same directory. The utild server will be started automatically by the WBEMCIM services startup procedure if the logical name WBEM_IGNORE_WBEMPROVIDERS is not defined.

Using this provider

Schema supported by this provider

The UP supports 4 classes. The tables mentioned below describe the properties in each class, intrinsic/extrinsic methods, types and units. The property tables have 3 columns. The first is the property name including types and units. The second is the property inheritance, including which class or superclass defines the property. The third is a description of the possible values and, where relevant, the data source for each property. Each row describes a property. The tables of extrinsic methods (those that are explicitly declared in each class) also have 3 columns. The first lists the method name and signature. The second describes the action of the method and all return values and arguments. The third column lists an exception that might be thrown by the method.

Table 1: HP_Utilization

This class contains utilization information. Each instance of this class will represent 24 hours worth of utilization data, containing 'SnapshotWidth' granularity. The present implementation of the provider will have samplings about every 5 minutes, so snapshot width will be nominally 300 (seconds) and NumSnapshots will be 288. As such, there will be 288 elements in each attribute of this class representing each 5 minute utilization value. The first element in each array will be the oldest. The utilization provider will maintain up to seven days worth of data. Consumers of this class who enumerateInstances(), will receive instances of all derived classes for all 7 days worth of data (if they exist). This is an abstract class (no defined keys), and data will only be returned for the various derived classes (which vary by granularity – OS, User, RP, etc.)

For attributes of this class which are arrays indexed by snapshot, if snapshot information wasn't available for the timeslice represented by the index, -1 will be returned. For these timeslices, the 'SnapshotTimeStamp' value will be a multiple of 'SnapshotWidth' seconds from the previous midnight since no snapshot actually exists for the timeslice. These timeslices will have values defined for the LAN and disk related attributes but the values will be zero-length arrays and the values of the 'MaxLanCards' and 'MaxDiskInterfaceCards' attributes will be zero.

If a 'getInstance' request is made specifying a period in the past when no snapshot information is available, an instance will be returned with no snapshot information (that is, all arrays indexed by snapshot will contain -1) as long as an interval exists earlier than the specified time that contains at least one snapshot.

A time other than midnight (GMT) may be passed to a 'getInstance' request to retrieve an instance covering the 24-hour period following the time specified. These instances will not be available from an 'enumerateInstances' request and their names will not be returned by a 'enumerateInstanceNames' request.

If a 'getInstance' request is made specifying a period before the earliest sample gathered or beginning in the future, a 'CIMObjectNotFoundException' will be thrown and no instance will be returned.

Note that the Utilization Provider defines Kilobytes (K) as 1024 bytes wherever this denomination is used.

Property name	***Property inheritance***	***Property value (and data source)***
datetime DailyTimeStamp	Local	The timestamp of the beginning of the 24 hour snapshot of utilization. This field will be overridden as a key in derived classes.
sint64 GUID	Local	This field represents a globally unique identifier for this class of utilization data. This attribute will be -1 until the setGUID() method is called. As with the DailyTimeStamp, this field will be overridden as a key in derived classes.
uint32 SnapshotWidth	Local	This field defines the number of seconds between snapshots in seconds.

uint32 CPUClockSpeed	Local	This field defines the CPU clock speed in MHZ.
uint16 NumSnapshots	Local	Number of utilization shapshots contained in the instance of the class.
sint8 DiskDataType	Local	Indicator of whether we are reporting disk utilization data by disk interface card or by disk. Values{"Unknown", "Disk", "DiskInterfaceCard" }, ValueMap {"-1", "0", "1" }
datetime SnapshotTimeStamp[]	Local	Timestamp of the end of the utilization snapshot.
sint8 MemoryUtilization[]	Local	Memory utilization in percentage form. Valid values returned in this field are integers between 0 and 100. This value will include User, DBC and System memory.
sint32 UserMemory[]	Local	Number of megabytes of user memory in use at the time of the utilization snapshot.
sint32 DBCMemory[]	Local	Number of megabytes of dynamic buffer cache at the time of the utilization snapshot.
sint32 SystemMemory[]	Local	Number of megabytes of system (kernel) memory in use at the time of the utilization snapshot.
sint32 TotalMemory[]	Local	Total number of megabytes of physical memory available during the utilization snapshot.
boolean MemoryBottleneckReading[]	Local	Indicator if the memory utilization snapshot was taken at a time when the system was swapping memory to disk. This is generally an indication that the memory utilization number may be artificially low.
sint8 CPUUtilization[]	Local	CPU utilization in percentage form. Valid values in this field are integers between 0 and 100.
sint16 NumActiveCPUs[]	Local	Number of CPU cores active during the utilization snapshot.
sint16 NumTotalCPUs[]	Local	Total number of CPU cores during the utilization snapshot. This number is the total number of CPUs visible to the OS, which would not include CPUs in inactive cells or deconfigured CPUs. This would include inactive iCOD CPUs, unassigned CPUs when running vPars, etc...
sint32 NumUsedCPUClockTicks[]	Local	Number of clock ticks consumed during the utilization snapshot.
sint32 TotalCPUClockTicks[]	Local	Total number of clock ticks available during the utilization snapshot.
boolean CPUBottleneckReading[]	Local	Indicator if the CPU utilization snapshot was taken at a time when the system had processes in the run queue. This is generally an indication that the CPU utilization number may be artificially low.
sint8 NumThreadsPerCore[]	Local	Number of threads (logical CPUs) per core at the time when the snapshot was taken. Since this attribute will be delivered in various sub-classes that may have different values, a special value of -2 will be returned at the OS level when a hybrid of values exist at the RP level.
sint16 MaxDiskInterfaceCards	Local	Number of disk interface cards. This field is equivalent to the size of the DiskInterfaceCards array. This field may or may not represent the number of cards present on the system at a given point in time, as adding and

		removing is allowed. For a card that is added or removed during the time represented by an instance of this class, incomplete data will be returned in the utilization arrays, but the cards will still be listed in the diskInterfaceCards array.
string DiskInterfaceCards[]	Local	This field returns the list of disk interface cards present during the utilization snapshot. For each entry in this array, there is a corresponding array of snapshots in the disk utilization arrays, which are effectively flat two dimensional arrays. For example, the first 'NumSnapshots' entries in the DiskUtilization array will be specific to the first disk interface card in this array, and so on. On Windows and Linux, the string for a disk device is a parseable, generated string as follows: 'drv:1,bus:0,dev:1,fun:0,tgt:1,lun:0'. NOTE: The special string of 'OS_LEVEL' will be used for disk metrics for the entire operating system and will be returned for all instances. By-disk metrics will be available only if the request is for the present or previous day. Earlier than this, only OS_LEVEL data will be available.
sint8 DiskBWUtilization[]	Local	Disk bandwidth utilization (by interface card or disk) in percentage form. Valid values returned in this field are integers between 0 and 100.
sint32 NumDiskRW[]	Local	Disk reads and writes (in kilobytes per second, by interface card) during the utilization snapshot.
sint32 MaxDiskRW[]	Local	Maximum disk reads and writes (in kilobytes per second, by interface card) during the utilization snapshot.
boolean DiskBottleneckReading[]	Local	Indicator if the disk utilization snapshot was taken at a time when the system had disk read/writes in the queue. This is generally an indication that the disk utilization number may be artificially low. As with other disk metric arrays, this is a 2 dimensional array.
sint16 MaxLanCards	Local	Number of LAN cards represented in the lanCards array. If a card is present or active only for a portion of a time period, snapshot data will only be available for the times when the card was active. In addition, some types of utilization data instances may not contain LAN utilization information. In that case, MaxLanCards will be 0. Therefore, this value may not represent the current number of active LAN cards present on the system.
string LanCards[]	Local	The list of LAN cards (by OS-specific identifier) present during the utilization snapshot. On HP-UX, the identifier is the special device file name for the card. For each entry in this array, there will be a corresponding array segment of snapshots in the LAN utilization arrays. See LanUtilization, LanPayloadTransferred, LanMaxPayloadPossible, and LanPacketsTransferred for more information. If MaxLanCards is 0, this property will be present, and contain an array with no elements. On Windows and Linux, the string for a lan device is a parseable, generated string as follows: 'bus:128,dev:1,fun:0,desc:Broadcom NetXtreme Gigabit Ethernet'. Note: The special string of 'OS_LEVEL' will be used for

		lan metrics for the entire operating system and will be returned for all instances. By-lan metrics will be available only if the request is for the present or previous day. Earlier than this, only OS_LEVEL data will be available.
sint8 LanLinkAggregator[]	Local	Indicator if a given LAN card is a link aggregator. If this attribute has a value of True(1), then data for this card should not be used in summing card data to determine the total LAN utilization for the OS instance. Otherwise, there would redundancy between data in the link aggregator and the real physical cards' data. Values{"Unknown", "False", "True" }, ValueMap {"-1", "0", "1" }
sint8 LanUtilization[]	Local	Average LAN utilization (by LAN card) in percent of the maximum transfer rate during the utilization snapshot. Valid values are integers between 0 and 100, inclusive. This array is a row-major order 1-d representation of a 2-d array where row 'n' represents the snapshots for the card with index 'n' in the LanCards array. Snapshot 's' for card 'n' is located at index 'n*NumSnapshots+s'. If MaxLanCards is 0, this property will be present, and contain an array with no elements.
sint32 LanPayloadTransferred[]	Local	Amount of LAN data transferred (incoming and outgoing) in kilobytes per second by LAN card during the utilization snapshot. This array is a row-major order 1-d representation of a 2-d array where row 'n' represents the snapshots for the card with index 'n' in the LanCards array. Snapshot 's' for card 'n' is located at index 'n*NumSnapshots+s'. If MaxLanCards is 0, this property will be present, and contain an array with no elements.
sint32 LanMaxPayloadPossible[]	Local	Maximum possible amount of LAN data in kilobytes (per second) transferable by a LAN card during the utilization snapshot, based on the maximum transfer rate of the card, e.g a 100 base T ethernet card has a maximum transfer rate of 100,000,000 bits per second or 12,207 kilobytes per second. If the snapshot interval were 10 seconds, then the maximum possible payload tranferrable would be 122,070 KB. This array is a row-major order 1-d representation of a 2-d array where row 'n' represents the snapshots for the card with index 'n' in the LanCards array. Snapshot 's' for card 'n' is located at index 'n*NumSnapshots+s'. If MaxLanCards is 0, this property will be present, and contain an array with no elements.
sint32 LanPacketsTransferred[]	Local	Number of packets of lan data transferred (incoming and outgoing) by LAN card during the utilization snapshot. This array is a row-major order 1-d representation of a 2-d array where row 'n' represents the snapshots for the card with index 'n' in the LanCards array. Snapshot 's' for card 'n' is located at index 'n*NumSnapshots+s'. If MaxLanCards is 0, this property will be present, and contain an array with no elements.
boolean LanBottleneckReading[]	Local	Indicator if the LAN utilization snapshot was taken at a time when the system had packets in the queue. This is generally an indication that the LAN utilization number may be artificially low. As with the other LAN metrics, this is a 2 dimensional array.

sint16 NumProcesses[]	Local	This field represents the number of processes that were found in the process table that were used as part of the utilization data.
sint32 UtilizationStatus[]	Local	This field represents the status of each utilization snapshot. The possible status values are: 0: No destabilizing events were detected when the snapshot was taken. -1: The snapshot does not contain any data. -2: The associated SG package was found to be running Page 6 on another node when the snapshot was taken. -3: The associated SG package was found to not exist when the snapshot was taken. -4: The associated SG package was found to exist but not be running when the snapshot was taken. -5: An online event such as an OL* change was detected when the snapshot was taken. -6: The associated procmap script does not exist, or we cannot access it without error. -7: The associated procmap script exists and we can access it, but it failed during execution. -8: A disk counter overflow was detected. Information was lost. -9: A lan counter overflow was detected. Information was lost. -10: Overlapping workload definitions detected. -11: User does not exist on host. -12: Procmap does not yeild any PIDs. -13: Executable does not physically exist on host. -14: Some data not collected due to WMI access problems. -15: Some data not collected due to PDH access problems.
sint32 CPUClockRate[]	Local	The average CPU clock speed seen during the snapshot. Under some forms of power-management, the value does not always reflect the current state of the computer system during the snapshot.
sint32 HPVM_NumUsedCPUClockTicks[]	Local	Number of clock ticks consumed during the utilization snapshot as reported by HPVM under HP-UX. Valid values returned in this field are integers greater than or equal to 0 on an HPVM guest; -1 in all other situations.
sint32 HPVM_TotalCPUClockTicks[]	Local	Total number of clock ticks available during the utilization snapshot as reported by HPVM under HP-UX. Valid values returned in this field are integers greater than or equal to 0 on an HPVM guest; -1 in all other situations. Note that this field may not exactly match TotalCPUClockTicks due to HPVM interval granularity.
sint8 HPVM_CPUUtilization[]	Local	CPU utilization in percentage form as reported by HPVM under HP-UX. The value is a measure of percent busy, not percent of capacity. Percent of capacity would take the current and maximum clock speed ratio into account. Valid values returned in this field are integers between 0 and 100; -1 in all other situations.
datetime CurrentTime	Local	The current date and time.
sint32 CPUMaxClockRate[]	Local	This field represents the maximum CPU clock speed of the computer system.
sint32 TrueSnapshotWidth[]	Local	This field defines the true length of each snapshot. The goal for the width of each snapshot equal 300 seconds,

		however system overhead can alter the length.
sint16NumActiveCPUSockets[]	Local	Average number of CPU sockets active during the snapshot. This number is useful to get the number of physical processors on multi-core/multi-threaded systems. Dividing the number of cores by the number of CPU sockets can provide the number of cores per processor

Table 2: HP_Utilization extrinsic methods

This table describes the extrinsic methods of HP_Utilization class.

Method name	***Description***	***Exception Thrown***
sint8 setGUID([in] sint64 newGUID)	Method that sets the GUID for the class of utilization data. The old GUID (defaults to -1) should be provided as the key, and the new GUID as the parameter to this method.	CIMOperationFailedException can be thrown on failure or CIMAccessDeniedException if the user has insufficient authority.
sint8 getCurrentUtilization([in] uint16 NumSecondsOfHistory, [out] sint64 GUID, [out] uint32 CPUClockSpeed, [out] sint8 MemoryUtilization, [out] sint32 UserMemory, [out] sint32 DBCMemory, [out] sint32 SystemMemory, [out] sint32 TotalMemory, [out] boolean MemoryBottleneckReading, [out] sint8 CPUUtilization, [out] sint16 NumActiveCPUs, [out] sint16 NumTotalCPUs, [out] sint32 NumUsedCPUClockTicks, [out] sint32 TotalCPUClockTicks, [out] boolean CPUBottleneckReading, [out] sint8 NumThreadsPerCore, [out] sint8 DiskDataType, [out] sint16 MaxDiskInterfaceCards, [out] string DiskInterfaceCards[], [out] sint8 DiskBWUtilization[], [out] sint32 NumDiskRW[], [out] sint32 MaxDiskRW[], [out] boolean DiskBottleneckReading[], [out] sint16 MaxLanCards, [out] string LanCards[], [out] sint8 LanLinkAggregator[], [out] sint8 LanUtilization[], [out] sint32 LanMaxPayloadPossible[], [out] sint32 LanPayloadTransferred[], [out] sint32 LanPacketsTransferred[], [out] boolean LanBottleneckReading[], [out] sint16 NumProcesses, [out] sint32 UtilizationStatus, [out] sint32 HPVM_NumUsedCPUClockTicks, [out] sint32 HPVM_TotalCPUClockTicks, [out] sint8 HPVM_CPUUtilization, [out] sint32 CPUClockRate, [out] datetime CurrentTime, [out] sint32 CPUMaxClockRate);	Method that returns utilization data immediately and return an average value per snapshot for the snapshots included in the previous number of seconds specified. NumSecondsOfHistory must be in increments of SnapshotWidth. This method returns zero in the success case. In the failure case, an exception (of the type suitable for the failure) will be thrown. For the disk and LAN utilization data, data is returned by either disk interface or LAN card. Thus, the data for the card listed in index I of the LanCards array will be in index I of the other Lan data arrays. Also, for disk and lan data, the values returned are the average value per snapshot for the group of snapshots included in the requested NumSecondsOfHistory. For example, the value of LanPayloadTransferred[I] is the average number of kilobytes of data transferred by the LAN card per snapshot over the number of snapshots included in NumSecondsOfHistory for card index I. It is not the sum of the payload transferred over the snapshots included in NumSecondsOfHistory. Each metric (CPU, Memory, Disk and LAN) has an associated bottleneck reading indicator. If any of the snapshots for a given metric used to calculate the average contained a bottleneck reading, this output parameter will be set to true. Note that CPUClockSpeed defines the current CPU clock speed and not the average (CPUClockRate defines the average). Under some forms of power-management, the value does not always reflect the true state of the computer system.	CIMOperationFailedException can be thrown on failure.

Table 3: HP_OSUtilization

This class contains utilization information at the OS level. Notes for getCurrentUtilization() method (inherited from HP_Utilization): The DailyTimeStamp key will be ignored when calling this method via invokeMethod(). A value of zero can be passed for this key (00000000000000.000000:000).

Property name	***Property inheritance***	***Property value (and data source)***
datetime DailyTimeStamp; (key)	Inherited from HP_Utilization	See description in HP_Utilization.
sint64 GUID; (key)	Inherited from HP_Utilization	See description in HP_Utilization.

Table 4: Non-Standard intrinsic methods supported by HP_OSUtilization

This table describes the non-standard intrinsic methods that supported by HP_OSUtilization.

Method name	***Inherited from***
getCurrentUtilization()	HP_Utilization
setGUID()	HP_Utilization

Table 5: HP_RPUtilization (not supported on Windows or OpenVMS)

This class contains utilization information at the resource partition level.

Notes for getCurrentUtilization() method (inherited from HP_Utilization): The DailyTimeStamp key will be ignored when calling this method via invokeMethod(). A value of zero can be passed for this key (00000000000000.000000:000), and only the rpID key will be used.

Property name	***Property inheritance***	***Property value (and data source)***
datetime DailyTimeStamp; (key)	Inherited from HP_Utilization	See description in HP_Utilization.
sint64 GUID; (key)	Inherited from HP_Utilization	See description in HP_Utilization.
string RpID; (key)	Local	This field represents the identifier for the resource partition. This is in string form, and will identify the type of resource partition as well. Fair share scheduler (FSS) resource partitions will have an RpID of FSS_#, and processor sets (PSET) resource partitions will have an RpID of PSET_#.

Table 6: Non-Standard intrinsic methods supported by HP_RPUtilization (not supported on OpenVMS)

This table describes the non-standard intrinsic methods that supported by HP_RPUtilization.

Method name	***Inherited from***
getCurrentUtilization()	HP_Utilization

Table 7: HP_WorkloadUtilization (not supported on Windows or OpenVMS)

This class contains utilization information at the workload level. In order to begin tracking utilization for a given workload, the watchWorkload() method should be called. Once a workload is being tracked, an additional instance of this class will be returned with a special WorkloadName of 'OTHER'. This instance will represent the remainder of utilization data not being explicitly tracked.

Notes for getCurrentUtilization() method (inherited from HP_Utilization):

1) The DailyTimeStamp key will be ignored when calling this method via invokeMethod(). A value of zero can be passed for this key (00000000000000.000000:000), and only the WorkloadName key will be used.

2) The watchWorkLoad() method must have been called and data collected for some period of time (as specified by the NumSecondsOfHistory parameter) in order for getCurrentUtilization() to return utilization data.

Property name	***Property inheritance***	***Property value (and data source)***
datetime DailyTimeStamp; (key)	HP_Utilization	See description in HP_Utilization.
sint64 GUID; (key)	HP_Utilization	See description in HP_Utilization.
string WorkloadName; (key)	Local	This field represents the name of the group of processes (workload) whose system utilization will be tracked.
string WorkloadDescription;	Local	This field represents the user specified description of the workload.
string SGPackageName;	Local	This field represents the Service Guard package name that this workload is associated with. This attribute can be set with the setSGPackageName() method. When this value is set, the workload will be dynamically activated and deactivated depending on the state of the Service Guard package.
boolean ActiveWorkload;	Local	This field is true if the workload is presently being watched. If a user calls the stopWatchingWorkload() method the data will not be deleted for the workload. Instead, the data will still be accessible until the normal housekeeping mechanism prunes the data directory.
uint8 CriteriaType[]; Values{Unknown, User, Executable, Group, Process }, ValueMap {0, 1, 2, 3, 4 }	Local	This field represents the type of criteria being watched for this workload (user, executable...). The indexes of the values in this array relate to the indexes of the values in the CriteriaName and AlternateCriteriaName arrays. This array will be empty for the OTHER instance.
string CriteriaName[];	Local	This field represents the value of criteria being watched for this workload (user's name, executable's path). See note for AlternateCriteriaName for info on processes with script-style process table signatures. This array will be empty for the OTHER instance.
string AlternateCriteriaName[];	Local	This field represents an alternate name for the criteria. This is used when an executable appears in the process table as '/usr/bin/perl script_name'. The CriteriaName in this case would be '/usr/bin/perl' and the

	AlternateCriteriaName would be 'script_name'. Since this is an optional field, some indexes will be null if an AlternateCriteriaName was not specified. This array will be empty for the OTHER instance.

Table 8: Non-Standard intrinsic methods supported by HP_WorkloadUtilization (not supported on OpenVMS)

This table describes the non-standard intrinsic methods that supported by HP_WorkloadUtilization.

Method name	***Inherited from***
getCurrentUtilization()	HP_Utilization
setGUID()	HP_Utilization

Table 9: HP_WorkloadUtilization extrinsic methods (not supported on OpenVMS)

This table describes the extrinsic methods of HP_WorkloadUtilization class.

Method name	***Description***	***Exception thrown***
sint8 watchWorkload([in] string WorkloadDescription, [in] uint8 CriteriaType[], [in] string CriteriaName[], [in] string AlternateCriteriaName[], [in] string SGPackageName);	Method that will cause the utilization provider to begin watching a workload's utilization. The utilization will be watched until the stopWatchingWorkload() method is called. The type, value, and alternname name arrays should have correlated indexes. Meaning, the CriteriaType in the first index should have its value in the first index of the CriteriaValue array (and so on). For more details about the values that can be passed for these input parameters, refer to their attribute descriptions earlier in this file.	CIMOperationFailedException can be thrown on failure, or CIMAccessDeniedException if the user has insufficient authority.
sint8 modifyWorkload([in] string newWorkloadName, [in] sint64 newGUID, [in] string WorkloadDescription, [in] uint8 CriteriaType[], [in] string CriteriaName[], [in] string AlternateCriteriaName[], [in] string SGPackageName);	Method that will modify a workload definition. This method will use the workload name and guid passed in as part of the key as 'old' values, and the input parameters below will be 'new' values. For more details about the values that can be passed for these input parameters, refer to their attribute descriptions earlier in this file.	CIMOperationFailedException can be thrown on failure, or CIMAccessDeniedException if the user has insufficient authority.
sint8 stopWatchingWorkload();	Method that will cause the utilization provider to stop watching a workload's utilization. The historic data that was recorded for this process group (up to seven days) will not be deleted when this method is called, but will over time be cleaned up by the utilization provider.	CIMOperationFailedException can be thrown on failure, or CIMAccessDeniedException if the user has insufficient authority.

sint8 getDescription([out] string WorkloadDescription);	Method that will return the description for a given workload. Note that the workload name is extracted from the WorkloadName key value and the DailyTimeStamp is ignored (as with the getCurrentUtilization() method).	CIMOperationFailedException can be thrown on failure.
sint8 getSGPackageName([out] string SGPackageName);	Method that will return the Service Guard package name for the workload. This method allows consumers to access the SG package name without getting an instance of historic data.	CIMOperationFailedException can be thrown on failure.

Table 10: Standard intrinsic methods.

This table describes the supported intrinsic methods of all the classes in this schema.

Method name	***Supported by***
createInstance	None
deleteInstance	None
modifyInstance	None
GetInstance	All classes
enumerateInstances	All classes
enumerateInstanceNames	All classes

Indications and Queries generated by this provider

This provider does not support indications or queries.

Associations provided by this provider

The Utilization Provider does not support any associations.

More Information

- **Unix:**
  - Schema (mof) files are delivered in /opt/util/mof.
  - utild manpage.

- **Windows:**
  - Schema (mof) files are delivered in \Program Files\Hewlett-Packard\UtilProvider\mof.
  - UPService.exe manpage is delivered in \Program Files\Hewlett-Packard\UtilProvider\doc.

- **OpenVMS:**
  - Schema (mof) files are delivered in /SYS$COMMON/WBEMPROVIDERS/MOF

- **WBEM Information:**
  - For a CIM tutorial, go to http://www.dmtf.org/education/cimtutorial.php.
  - For information about the Pegasus WBEM infrastructure, see http://www.opengroup.org/pegasus.
  - For information about WBEM and a list of providers/clients available, see the Pegasus Home Page.
  - For information about the WBEM infrastructure and administration information, see the WBEM manual or the man pages.

Known Defects and Performance Considerations

None.

For additional information on HP products and services, visit us at http://www.hp.com.

For the location of the nearest sales office, call:

United States: +1 800 637 7740

Canada: +1 905 206 4725

Japan: +81 3 3331 6111

Latin America: +1 305 267 4220

Australia/New Zealand: +61 3 9272 2895

Asia Pacific: +8522 599 7777

Europe/Africa/Middle East: +41 22 780 81 11

For more information, contact any of our worldwide sales offices or HP Channel Partners (in the U.S., call 1 800 637 7740).

02/2011