

Partner Common Services

Functional Specification

Version 7.09

October 13, 2011

Partner Common Services Team

Hewlett Packard Company

© Copyright 2010 Hewlett-Packard Development Company, L.P.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor’s standard commercial license.

# Revision History

The following table provides a history of changes to this document.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Revision | Modifications |
| 03/04/2011 | 0 | 1 | Steve Collins: Initial content consolidated from previous Service Definition Specification and CS.Next\_2011ArchDesign specs dated 05/24/2010 – 02/23/2011 |
| 03/09/2011 | 0 | 2 | Jing Zhao: Updated Configuration and Password with rack manager and server specific info. Updated SNMP data source in Data Collection. |
| 03/09/2011 | 0 | 3 | Tina Rogers: Updated Data Collection general topics and WBEM. |
| 03/10/2011 | 0 | 4 | Chris Frantz: Update discovery params, event service and event descriptions. |
| 03/10/2011 | 0 | 5 | Tina Rogers: Updated configuration properties table. Updated the data collection status table. |
| 03/10/2011 | 0 | 6 | Tina Rogers: Updated TOC |
| 03/11/2011 | 0 | 7 | Jing Zhao: Updated ProLiant Server. |
| 03/11/2011 | 0 | 8 | Jing Zhao: Updated Discovery to add supported plugin parameters and how to retrieve a portion of data model. Moved ProLiant Server from Interfaces to Functional Description. |
| 03/15/2011 | 0 | 9 | Tina Rogers: Added Breckenridge detailed inventory tables in section 4. |
| 03/16/2011 | 0 | 10 | Chris Frantz: Added section for deleting managed entities and a section describing the event service. |
| 03/16/2011 | 0 | 11 | Tina Rogers: Added WBEM event discovery. |
| 03/16/2011 | 0 | 12 | Tina Rogers: Added WBEM configuration descriptions. |
| 03/17/2011 | 1 | 0 | Steve Collins: Accepted all changes and did minor format cleanup – this version posted external for client teams |
| 03/17/2011 | 1 | 1 | Tina Rogers: Added more detail to Event Service to describe usage of MessageEntities.xml file. Added MP properties to detailed inventory |
| 03/18/2011 | 1 | 2 | Steve Collins: Updated the architecture diagram in section 4. Moved Eventing section after Discovery. Removed comments (documented in Action Item spreadsheet). Other miscellaneous cleanup. |
| 03/29/2011 | 1 | 3 | Chris Frantz: Update the configuration sections to reflect the reorganized config.xml file. Add REST API and description of the Discovery Engine. |
| 04/04/2011 | 1 | 4 | Jing Zhao: Updated data source for ProLiant Server entity. |
| 04/05/2011 | 1 | 5 | Tina Rogers: Updated inventory tables to reflect only the properties available for SNMP and WBEM |
| 04/19/2011 | 1 | 6 | Tina Rogers: Updated configuration parameters and detailed inventory. |
| 4/25/2011 | 1 | 7 | Chris Frantz:  Updated Configuration section with info on DNS, traptest, discovery dns and dbretry.  Updated 3.3.2 and 3.3.3 with error information.  Added 3.3.4 about DNS cache.  Inserted 4.1 with the startup procedure.  Updated 4.2.3 with DNS/discovery info.  Updated 4.2.5.2 with Win32 SNMP api info.  Inserted 5.2.11 with Privilege Error. |
| 5/3/2011 | 1 | 8 | Tina Rogers:  Added TinyVCenter Functional Description.  Updated ProLiant Server WBEM discovery.  Removed references to Rack Manager. |
| 5/23/2011 | 1 | 9 | Jing Zhao:  Updated ProLiant Server iLO4 OOB data source. |
| 7/29/2011 | 1 | 9 | Tina Rogers:  Added flow chart for vCenter discovery process.  Added description for setting entity status for server and vCenter.  Added vCenter config parameters |
| 7/29/2011 | 1 | 11 | Jing Zhao:  Updated server summary data, discovery algorithm and discovery interface for Gen8 server support. |
| 8/01/2011 | 1 | 12 | Chris Frantz  Updated config and events. |
| 8/11/2011 | 7 | 01 | Chris Frantz  Re arrange “discovery” into “entity” and “discovery”  Add info about passing credentials with discovery  Add info about entity updates. |
| 8/29/2011 | 7 | 02 | Tina Rogers  Added sections for retry count and REST interface \_\_updatenow |
| 9/07/2011 | 7 | 03 | Gary Hester Add new section for control verbs Added description for the iLO power on/off and OA set LCD user notes verbs. |
| 9/12/2011 | 7 | 04 | Gary Hester Added descriptions for iLO uid\_control and clear\_eventlog verbs. |
| 9/27/2011 | 7 | 05 | Jing Zhao Updated server properties. |
| 9/28/11 | 7 | 06 | Tina Rogers  Added documentation for retryforever configuration parameter |
| 10/11/2011 | 7 | 07 | Chris Frantz  Document web/transport, collectore/useWinSnmpApi and the “replace” discovery parameter. |
| 10/12/2011 | 7 | 08 | Jing Zhao  Added documentation for cached server data. |
| 10/13/2011 | 7 | 09 | Chris Frantz  Better documentation about the replace flag.  Document the NT event conversion.  Gary Hester  CSERIES event indication |

# Table of Contents

[Revision History 2](#_Toc306270573)

[Table of Contents 4](#_Toc306270574)

[1 Introduction 6](#_Toc306270575)

[1.1 Summary 6](#_Toc306270576)

[1.2 Requirements 6](#_Toc306270577)

[1.3 Numbers 7](#_Toc306270578)

[1.4 Terminology 7](#_Toc306270579)

[1.5 References 7](#_Toc306270580)

[2 Use Cases 8](#_Toc306270581)

[2.1 (Use Case Name) 8](#_Toc306270582)

[3 Interfaces 9](#_Toc306270583)

[3.1 Configuration 9](#_Toc306270584)

[3.1.1 Get Common Services Version 12](#_Toc306270585)

[3.1.2 List all loaded plugins 12](#_Toc306270586)

[3.1.3 List all services 13](#_Toc306270587)

[3.1.4 Retrieve the configuration schema 13](#_Toc306270588)

[3.1.5 Retrieve the configuration 14](#_Toc306270589)

[3.1.6 Modify the Configuration 14](#_Toc306270590)

[3.2 Password 15](#_Toc306270591)

[3.2.1 Retrieve the password list 15](#_Toc306270592)

[3.2.2 Update an existing password 16](#_Toc306270593)

[3.2.3 Delete an existing password 16](#_Toc306270594)

[3.2.4 Create a new password 17](#_Toc306270595)

[3.3 Entities 17](#_Toc306270596)

[3.3.1 List all managed element names 17](#_Toc306270597)

[3.3.2 List all entities of a given Manged Entity Type 18](#_Toc306270598)

[3.3.3 Retrieve a schema for a managed entity type 18](#_Toc306270599)

[3.3.4 Retrieve the schema version of a managed type 19](#_Toc306270600)

[3.3.5 Retrieve a managed entity data model 20](#_Toc306270601)

[3.3.6 Update a managed entity data model 22](#_Toc306270602)

[3.3.7 Delete a managed entity data model 23](#_Toc306270603)

[3.4 Discovery 23](#_Toc306270604)

[3.4.1 Discover an IP or IP range 23](#_Toc306270605)

[3.4.2 Query the Discovery Engine for status 26](#_Toc306270606)

[3.4.3 Query the Discovery DNS cache 27](#_Toc306270607)

[3.5 Eventing 27](#_Toc306270608)

[3.5.1 Retrieving the Event Schema 27](#_Toc306270609)

[3.5.2 Retrieve All Events 27](#_Toc306270610)

[3.5.3 Retrieve the Last Event 28](#_Toc306270611)

[3.5.4 Retrieve a sequence of events 28](#_Toc306270612)

[3.6 Virtual Connect 29](#_Toc306270613)

[3.6.1 Uniquely Identifying Virtual Connect Domains 29](#_Toc306270614)

[3.6.2 Items of Interest in the Virtual Connect Data Model 29](#_Toc306270615)

[3.6.3 Retrieving the Virtual Connect Data Model 29](#_Toc306270616)

[3.6.4 Retrieving a subset of the Virtual Connect data model for a specific host 30](#_Toc306270617)

[3.7 Control Verbs 30](#_Toc306270618)

[3.7.1 iLO Control 31](#_Toc306270619)

[3.7.2 OA Control 32](#_Toc306270620)

[4 Functional Description 33](#_Toc306270621)

[4.1 Startup Process 33](#_Toc306270622)

[4.2 Services and Plugins 34](#_Toc306270623)

[4.2.1 Configuration Service 34](#_Toc306270624)

[4.2.2 Password Service 34](#_Toc306270625)

[4.2.3 Discovery Service 34](#_Toc306270626)

[4.2.4 Event Service 35](#_Toc306270627)

[4.2.5 ProLiant Server Plugin 36](#_Toc306270628)

[4.2.6 Tiny VCenter Plugin 46](#_Toc306270629)

[4.2.7 Entity Reconnect Algorithm 48](#_Toc306270630)

[4.3 Error Handling 50](#_Toc306270631)

[4.3.1 Error Messages 50](#_Toc306270632)

[4.4 Security 51](#_Toc306270633)

[4.5 Logging 51](#_Toc306270634)

[4.6 Boundary Conditions 51](#_Toc306270635)

[4.7 Constraints 51](#_Toc306270636)

[4.8 Platforms 51](#_Toc306270637)

[4.9 Internationalization 51](#_Toc306270638)

[4.10 Performance 51](#_Toc306270639)

[4.11 Portability 52](#_Toc306270640)

[4.12 Expandability 52](#_Toc306270641)

[4.13 Support and Maintenance 52](#_Toc306270642)

[4.14 Migration and Upgrade 52](#_Toc306270643)

[5 Event Descriptions 53](#_Toc306270644)

[5.1 Event Structure 53](#_Toc306270645)

[5.2 Formalized Events 53](#_Toc306270646)

[5.2.1 Start of Day Event 53](#_Toc306270647)

[5.2.2 Test Event 53](#_Toc306270648)

[5.2.3 Entity Created 54](#_Toc306270649)

[5.2.4 Entity Modified 54](#_Toc306270650)

[5.2.5 Entity Destroyed 54](#_Toc306270651)

[5.2.6 No Credentials 54](#_Toc306270652)

[5.2.7 Auth Error 54](#_Toc306270653)

[5.2.8 Invalid Firmware 54](#_Toc306270654)

[5.2.9 Failover 54](#_Toc306270655)

[5.2.10 Connect Error 55](#_Toc306270656)

[5.2.11 PrivilegeError 55](#_Toc306270657)

[5.2.12 C-Series Hardware Event 55](#_Toc306270658)

[5.2.13 C-Series Status Event 55](#_Toc306270659)

[5.2.14 SNMP Event 55](#_Toc306270660)

[5.2.15 WBEM Indication 55](#_Toc306270661)

[5.2.16 CSERIES Indication 56](#_Toc306270662)

# Introduction

Partner Common Services provides core services to the IC4SC (Red Mountain) and IC4VC (Breckenridge) products (clients).

## Summary

Partner Common Services 6.3 will be integrated with the Red Mountain 6.3 and Breckenridge 6.3 products targeted for release mid-2011.

Partner Common Services is integrated “tightly” with the Red Mountain and Breckenridge client products such that it ships with and installs as part of those products themselves. Partner Common Services is not treated as a separate customer deliverable.

The 6.3 release is the first release in which Partner Common Services is integrated with the Red Mountain product, and the 2nd release in which it is integrated with the Breckenridge product.

A late 2H2011 release will target support for Gen8 servers, and Positano/iRaptor (embedded management) in particular.

## Requirements

1H2011 Requirements are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req #** | **Description** | **Source** | **Client Usage** |
| 1 | Design specification that encompasses all items being developed | RM |  |
| 2 | Blade support – OA authentication | RM |  |
| 3 | Blade support – OA data collection | RM |  |
| 4 | Blade support – OA eventing | RM |  |
| 5 | Event support – Register for traps | RM |  |
| 6 | Event support – Process traps according to meta-data in master XML | RM |  |
| 7 | ESX/Linux support (SNMP) using HP SNMP agents on ESX 4.x, RHEL 5, RHEL 6, SLES 10 and SLES 11 – Authentication for ESX/Linux over SNMP | RM |  |
| 8 | ESX/Linux support (SNMP) using HP SNMP agents on ESX 4.x, RHEL 5, RHEL 6, SLES 10 and SLES 11 – Data collection for ESX/Linux over SNMP | RM |  |
| 9 | ESX/Linux support (SNMP) using HP SNMP agents on ESX 4.x, RHEL 5, RHEL 6, SLES 10 and SLES 11 – Eventing for ESX/Linux over SNMP | RM |  |
| 10 | Support ProLiant G7 servers (including 100-series servers) | RM |  |
| 11 | Capable of running on a Windows Failover Cluster | RM |  |
| 12 | VC-FC support – Update WSDL to account for new data structure associated with Sheppard | Br |  |
| 13 | Network diagram enhancements – Support for VC FW v3.30 – integrate new telemetry data with the rest of the data structures | Br |  |
| 14 | Communication with SNMP agents and CIM providers | Br |  |
| 15 | Programmatic interface to restart CS | Br |  |

2H2011 Requirements are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req #** | **Description** | **Source** | **Client Usage** |
| 16 | ESXi/MN support (WBEM) using HP WBEM providers on ESXi 4.2 and ESX M/N – Authentication for ESX/Linux over WBEM | RM |  |
| 17 | ESXi/MN support (WBEM) using HP WBEM providers on ESXi 4.2 and ESX M/N – Data collection for ESX/Linux over WBEM | RM |  |
| 18 | ESXi/MN support (WBEM) using HP WBEM providers on ESXi 4.2 and ESX M/N – Eventing for ESX/Linux over WBEM | RM |  |
| 19 | Converged Infrastructure support – Authentication for Virtual Connect | RM |  |
| 20 | Converged Infrastructure support – Data collection for Virtual Connect | RM |  |
| 21 | Converged Infrastructure support – Eventing for Virtual Connect | RM |  |
| 22 | Converged Infrastructure support – Authentication for Fenway | RM |  |
| 23 | Converged Infrastructure support – Data collection for Fenway | RM |  |
| 24 | Converged Infrastructure support – Eventing for Fenway | RM |  |
| 25 | Positano support | RM |  |

## Numbers

Breckenridge is expected to manage up to 200 server nodes.

Red Mountain is expected to manage up to 10,000 server nodes.

## Terminology

Red Mountain = Insight Control for Microsoft System Center

IC4SC = Insight Control for Microsoft System Center

Breckenridge = Insight Control for VMware vCenter

IC4VC = Insight Control for VMware vCenter

## References

[Ref1]

Fielding, Roy T.; Taylor, Richard N. (2002-05), "Principled Design of the Modern Web Architecture" (PDF), ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) 2 (2): 115–150, doi:10.1145/514183.514185, ISSN 1533-5399

# Use Cases

*TBD*

Following are expected use cases for Partner Common Services.

## (Use Case Name)

|  |  |
| --- | --- |
| **Use Case ID** |  |
| **Description** |  |
| **Preconditions** |  |
| **Assumptions** |  |
| **Post conditions** |  |
| **Normal Course** |  |
| **Alternative Course** |  |
| **Notes and issues** |  |

# Interfaces

Partner Common Services uses a REST style interface for clients.

Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web [Ref1]. A REST style architecture has many benefits including its usage for manageability services. The main attractive benefits for manageability services include: simplicity, efficiency and scalability. Other RPC mechanisms such as the Simple Object Access Protocol (SOAP) have previously been used for manageability, but are considered heavier and more complex.

The following sections detail the REST interface exposed to client products by Partner Common Services.

## Configuration

The Configuration service allows a Common Services client to query and modify the configuration of the Common Services. The Configuration service does not correspond exactly to the generic REST operations listed in section 2 of the manageREST specification. The operations that the Configuration service implements are a best fit.

The Common Services configuration is stored in share/config.xml. Common Services will read this file at startup to learn its configuration. If the file is missing, Common Services will assign reasonable default values as indicated below and create config.xml.

A partner application (such as RM or Br) that installs Common Services may pre-populate the config.xml file.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Configuration Item** | | **Meaning** | **Default** | |
| hostname | | The fully qualified domain name of the host. | None | |
| datadir | | The directory where Common Services will store persistent information. | Db | |
| dbretry | | The number of minutes between attempts to restore entities from the database during startup. Entities that experience a non-fatal error during restore (such as a communication error) will be retried. | 15 | |
| web/address | | The address of the interface the Common Services webserver will bind to. | localhost | |
| web/port | | The TCP port the webserver will listen on. | 50026 | |
| web/debug | | Whether or not to enable debugging in the web.py framework.  This is used internally for dev/test/debug only; not used for clients/customers | false | |
| web/useSSL | | Whether or not to use SSL for serving content. | false | |
| web/defaultlang | | The default language for REST responses.  Allowed values: xml or json | xml | |
| web/authorization | | The authorization scheme to use.  Allowed values:  None = no authentication required  win32 = use Win32 credentials (any Win32 Administrator credential)  Pwfile = use credentials from the Common Services password.xml  any = use either win32 or pwfile | any | |
| web/certinfo | | The list of fields that should be used when generating a certificate signing request or a self-signed certificate. When the Common Services web server is put into SSL mode, this data is used to generate a self-signed certificate. | None | |
| web/transport | | Specify which http client library to use for client transactions (such as SOAP and WBEM) Current implementation supports urllib2 and curl. | Urllib2 | |
| dns/fcr | | Whether or not to use forward confirmed reverse DNS for reverse DNS lookups. | true | |
| dns/timeout | | Number of seconds before DNS requests timeout. | 30 | |
| dns/suffix | | List of DNS suffixes to use for forward lookups of short names. | <empty list> | |
| event/timeout | | Number of minutes events stay in the queue before expiring. | 15 | |
| event/redirect | | Name of the NT Event log to redirect extrinsic events. | <empty string> | |
| event/specfile | | The path to the Message Entities XML file | None | |
| event/unknownExtrinsic | | What to do with extrinsic events not specified in the specfile.  Allowed values: drop, keep | drop | |
| scheduler/interval | | The time interval for the scheduler to wake up and run schedulable tasks. | 3 | |
| scheduler/workpool | | The number of threads created in the work pool for schedulable task execution. | 16 | |
| scheduler/queuesize | | The size of the work queue. Tasks are placed on the work queue until a work thread is available. | 50 | |
| collector/wbemport | | The TCP port the WBEM webserver will listen on. | 50028 | |
| collector/snmpTrapPort | | SNMP trap listener port number. | 162 | |
| collector/useWinSnmpApi | | Use the Windows SNMP API for SNMP GETs and WALKs | true | |
| collector/traptest | | Test mode for SNMP traps. When using the Win32 SNMP api, the trap address will be taken from the SNMP packet rather than the IP header. | None (false) | |
| collector/subsystem/NIC | | Flag to trigger collection of the pre-defined inventory data for NIC | true | |
| collector/subsystem/SmartArray | | Flag to trigger collection of the pre-defined inventory data for Smart Array | | true |
| collector/subsystem/Software | | Flag to trigger collection of the pre-defined inventory data for Software | | true |
| collector/subsystem/Firmware | | Flag to trigger collection of the pre-defined inventory data for Firmware | | true |
| collector/subsystem/  ManagementProtocol | | Flag to trigger collection of the pre-defined inventory data for MP | | true |
| collector/subsystem/  FibreChannelHBA | | Flag to trigger collection of the pre-defined inventory data for FCHBA | | true |
| excepthook | | Whether or not to install a python exception hook for detecting and logging errors.  This is used internally for dev/test/debug only; not used for clients/customers | True | |
| plugins | | The list of plugins Common Services should load on startup.  The plugin name is specified in the name attribute:  <plugin name=”plugin-name”/>  Allowed values:  hp.plugins.discovery  hp.plugins.ilo  hp.plugins.oa  hp.plugins.vcm  hp.plugins.rm  hp.hplugins.server  hp.plugins.tinyvcenter | None | |
| The individual plugin configuration items belong in the appropriately named plugin node:  <plugin name=”hp.plugins.ilo”>  <ilo>  <interval>15</interval>  </ilo>  </plugin> | | | | |
| plugin/discovery/interval | How often the discovery engine retries discovery on a node (minutes) | | 60 | |
| plugin/discovery/retry | The number the discovery engine retries a node before giving up | | 5 | |
| plugin/discovery/threads | The number of threads the discovery engine will use | | 4 | |
| plugin/discovery/dnsinterval | Number of minutes to between DNS cache refreshes | | 30 | |
| plugin/discover/dnsexpiration | Age of a DNS cache entry (in minutes) before it is expired | | 120 | |
| plugin/discover/dnsreverse | Whether or not DiscoveryEngine should perform reverse DNS lookups. | | true | |
| plugin/ilo/interval | How often to poll iLO for data (minutes) | | 15 | |
| plugin/server/primaryprotocol | The primary protocol used by the server plugin | | Snmp | |
| plugin/server/interval/inventory | How often to poll the server for inventory data (minutes) | | 180 | |
| plugin/server/interval/summary | How often to poll the server for summary data (minutes) | | 15 | |
| plugin/server/interval/status | How often to poll the server for status data (minutes) | | 5 | |
| plugin/tinyvcenter/interval | How often to poll the vCenter for managed device data (minutes) | | 900 | |
| plugins/oa/reconnects | How many times to attempt reconnecting to OA after a failure or failover event. | | 15 | |
| plugins/vcm/reconnects | How many times to attempt reconnecting to VCM after a failure or failover event | | 15 | |
| plugins/server/reconnects | How many times to attempt reconnecting to a server after a failure | | 5 | |
| plugins/ilo/reconnects | How many times to attempt reconnecting to an ilo after a failure | | 5 | |
| plugins/ tinyvcenter /reconnects | How many times to attempt reconnecting to vCenter after a failure | | 5 | |
| plugins/server/backoffcap | The number of hours to continue scheduling after the reconnect count is reached (minutes) | | 5760 | |
| plugins/ilo/backoffcap | The number of hours to continue scheduling after the reconnect count is reached (minutes) | | 5760 | |
| plugins/tinyvcenter/backoffcap | The number of hours to continue scheduling after the reconnect count is reached (minutes) | | 5760 | |
| Plugins/oa/backoffcap | The number of hours to continue scheduling after the reconnect count is reached (minutes) | | 5760 | |
| Plugins/vcm/backoffcap | The number of hours to continue scheduling after the reconnect count is reached (minutes) | | 5760 | |
| plugins/server/retryforever | A flag to determine if the entity should continue to be scheduled once the backoffcap has been reached. The scheduling interval will be set at the last calculated interval when the backoff cap was reached. | | False | |
| plugins/ilo/ retryforever | A flag to determine if the entity should continue to be scheduled once the backoffcap has been reached. The scheduling interval will be set at the last calculated interval when the backoff cap was reached. | | False | |
| plugins/tinyvcenter/ retryforever | A flag to determine if the entity should continue to be scheduled once the backoffcap has been reached. The scheduling interval will be set at the last calculated interval when the backoff cap was reached. | | False | |
| Plugins/oa/ retryforever | A flag to determine if the entity should continue to be scheduled once the backoffcap has been reached. The scheduling interval will be set at the last calculated interval when the backoff cap was reached. | | False | |
| Plugins/vcm/ retryforever | A flag to determine if the entity should continue to be scheduled once the backoffcap has been reached. The scheduling interval will be set at the last calculated interval when the backoff cap was reached. | | False | |
|  |  | |  | |

### Get Common Services Version

Operation (implements 3.2.1):

**GET** http://<host>/version

Description:

Retrieve the Common Services Version number. The Common Services version number is a standard Windows 4-part version number. The components of the 4-part version number are constructed as follows:

|  |  |
| --- | --- |
| Major | The major version number represents major releases and/or architectural changes to the common services framework. |
| Minor | The minor version number represents a release with new features added, but without significant changes to the framework itself. |
| Subminor | The subminor version number represents a bugfix release. |
| Pass | The pass number represents how many times a (major, minor, subminor) release has been built. Normally, the pass number is not displayed to the end user. |

Result (xml or json):

The common services version number.

Example:

**Request:**

GET http://localhost:50026/plugins

**Response:**

200 OK

<version>1.0.3.1</version>

### List all loaded plugins

Operation (implements 3.2.1):

**GET** http://<host>/plugins

Description:

List all loaded Common Services plugins.

Result (xml or json):

The list of currently loaded plugins. Common Service plugins can provide data collectors, services or other functionality. Do not confuse plugins with services (section 2.3).

Example:

**Request:**

GET http://localhost:50026/plugins

**Response:**

200 OK

<plugins>

<plugin>hp.plugins.discovery</plugin>

<plugin>hp.plugins.oa</plugin>

<plugin>hp.plugins.vcm</plugin>

<plugin>hp.plugins.ilo</plugin>

<plugin>unknown</plugin>

</plugins>

### List all services

Operation (implements 3.2.1):

**GET** http://<host>/services

Description:

List all loaded services known to Common Services. Each service provides a set of REST URLs that enhance the Common Service functionality.

Result (xml or json):

Thie list of available services.

Example:

**Request:**

GET http://localhost:50026/services

**Response:**

200 OK

<services>

<service>discovery</service>

</services>

### Retrieve the configuration schema

Operation (implements 3.2.2):

GET http://<host>/config/schema

Description:

Retrieve the schema for the Common Services configuration manager.

Result (xml):

An XML schema document.

Example:

**Request:**

GET http://localhost:50026/config/schema

**Response:**

200 OK

<xs:schema xmlns="http://config" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://config" targetNamespace="http://config">

<xs:complexType name="config">

<xs:all>

<xs:element type="xs:int" name="webport"/>

<xs:element type="xs:string" name="webaddress"/>

<xs:element type="xs:string" name="hostname"/>

<xs:element type="xs:string" name="authorization"/>

<xs:element type="xs:boolean" name="webdebug"/>

<xs:element type="xs:boolean" name="useSSL"/>

<xs:element type="xs:boolean" name="excepthook"/>

<xs:element type="certinfo" name="certinfo"/>

<xs:element type="plugins" name="plugins"/>

</xs:all>

</xs:complexType>

</xs:schema>

### Retrieve the configuration

Operation (implements 3.2.3):

GET http://<host>/config

Description:

Retrieve the Common Services configuration.

Result (xml or json):

The configuration data structure.

Example:

**Request:**

GET [http://localhost:50026/config](http://hp.com/config)

**Response:**

200 OK

<config>

<hostname>cjf-wintest.americas.hpqcorp.net</hostname>

<webport>50026</webport>

<webaddress>0.0.0.0</webaddress>

<webdebug>true</webdebug>

<useSSL>false</useSSL>

<excepthook>true</excepthook>

<plugins>

<plugin>hp.plugins.discovery</plugin>

<plugin>hp.plugins.ilo</plugin>

<plugin>hp.plugins.oa</plugin>

<plugin>hp.plugins.vcm</plugin>

</plugins>

</config>

### Modify the Configuration

Operation (implements 3.2.4):

PUT http://<host>/config

Description:

Change the configuration data

Result (xml or json):

An HTTP status code.

Example:

**Request:**

PUT [http://localhost:50026/config](http://hp.com/config)

<config>

<useSSL>true</useSSL>

</config>

**Response:**

200 OK

## Password

The password service allows a Common Services client to manage the password database used by Common Services to log in to other managed devices, such as iLO, OA or Virtual Connect modules. The password service does not exactly match the REST operations listed in section 3. The operations that the Password service implements are a best fit.

Each password entry has several fields:

|  |  |
| --- | --- |
| **Field Name** | **Meaning** |
| username | The username used to log into a remote service. |
| password | The password used to log into a remote service. |
| host | The host to which this username/password belongs. You may use an IP address or an UUID, or you may use \* to mean “any host”. |
| type | The type of host (string).  Currently allowed types are:  HP Common Services  iLO  Onboard Administrator  HP SIM  Virtual Connect  vCenter  System Center  ProLiant Server  SNMP Community String |
| id | A uniquie ID assigned to this password entry. This ID has no relationship to other UUID used elsewhere in Common Services. |

### Retrieve the password list

Operation (implements 3.3.2):

GET http://<host>/password

Description:

Retrieves the current password database (all records). When retrieving password records, the password field is not returned.

Result (xml or json):

The current password database (except for the password field).

Example:

**Request:**

GET [http://localhost:50026/password](http://hp.com/password)?lang=json

**Response:**

200 OK

{

"password": [

{

"username": "Admin",

"host": "\*",

"type": "HP SIM",

"id": "8f205254-7bf2-11df-b946-00301bbe6019"

}

]

}

### Update an existing password

Operation (implements 3.2.5):

PUT http://<host>/password

Description:

Updates the password entry corresponding to the specified id. All fields must be provides, and all fields except the id must match the existing entry.

Result (xml or json):

An HTTP status code. Will return the entire password database if a query string parameter of list=true is specified.

Example:

**Request:**

PUT [http://localhost:50026/password](http://hp.com/password)?lang=json

{

"username": "Admin",

"host": "\*",

"type": "HP SIM",

"password": "abc123",

"id": "8f205254-7bf2-11df-b946-00301bbe6019"

}

**Response:**

200 OK

### Delete an existing password

Operation (implements 3.2.7):

DELETE http://<host>/password

Description:

Deletes the password entry corresponding to the specified id. Only the ID needs to be provided; other fields may be omitted.

Result (xml or json):

An HTTP status code. May also return the entire password database if a query string parameter of list=true is specified.

Example:

**Request:**

PUT [http://localhost:50026/password](http://hp.com/password)?lang=json

{

"id": "8f205254-7bf2-11df-b946-00301bbe6019"

}

**Response:**

200 OK

### Create a new password

Operation (implements 3.2.6):

POST http://<host>/password

Description:

Creates a new password record. NOTE: when creating a new password, the ID must be the value “new”.

Result (xml or json):

An HTTP status code and the new password entry. Returns HTTP 409 Conflict if the client attempts to create a password with host and type values that match an existing entry. Also returns an error condition if the ID value is not “new”.

May also return the entire password database if a query string parameter of list=true is specified.

Example:

**Request:**

POST [http://localhost:50026/password](http://hp.com/password)?lang=json

{

"username": "Admin",

"host": "\*",

"type": "Virtual Connect",

"password": "abc123",

"id": "new"

}

**Response:**

200 OK

{

"username": "Admin",

"host": "\*",

"type": "Virtual Connect",

"id": "b3450ce1-904e-11df-ae18-505400cfcf02"

}

## Entities

Common Services provides an entitiy service for retrieving, modifying and deleting entities stored in its database.

### List all managed element names

Operation (implements 3.2.1):

**GET** http://<host>/discovery

Description:

List all managed element types that can be discovered by the Discovery service

Result (xml or json):

The list of managed element types.

Example:

**Request:**

GET http://localhost:50026/discovery

**Response:**

200 OK

<metypes>

<type>oa</type>

<type>vcm</type>

<type>ilo</type>

<type>unknown</type>

<type>any</type>

</metypes>

### List all entities of a given Manged Entity Type

Operation (implements 3.3.2):

GET http://<host>/entity/<mename>

Description:

List all discovered entities of the specified managed entity type.

Result (xml or json):

A list of UUIDs.

Example:

**Request:**

GET http://localhost:50026/entity/ilo

**Response:**

200 OK

<uuids>

<uuid>36363134-3635-5532-5837-313930335233</uuid>

<uuid>2f284a14-70b9-0000-0000-000000000000</uuid>

<uuid>36363134-3635-5532-5837-31393032385a</uuid>

</uuids>

### Retrieve a schema for a managed entity type

Operation (implements 3.3.1):

GET http://<host>/entity/<mename>/schema

Description:

Retrieve the schema for the managed entity type.

Result (xml):

An XML schema document.

Example:

**Request:**

GET http://localhost:50026/entity/ilo/schema

**Response:**

200 OK

<xs:schema xmlns="http://hp.com/isb/commonservices/entity" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:hpvcd="http://hp.com/iss/net/vcm/resourceDomain" xmlns:tns="http://hp.com/isb/commonservices/entity" targetNamespace="http://hp.com/isb/commonservices/entity">

<xs:complexType name="Address">

<xs:all>

<xs:element type="xs:string" name="mac"/>

<xs:element type="xs:string" name="ipv4"/>

</xs:all>

</xs:complexType>

<xs:complexType name="RelationList">

<xs:all>

<xs:element maxOccurs="unbounded" type="xs:string" name="uuid" minOccurs="0"/>

</xs:all>

</xs:complexType>

<xs:complexType name="Timestamp">

<xs:all>

<xs:element type="xs:datetime" name="created"/>

<xs:element type="xs:datetime" name="accessed"/>

<xs:element type="xs:datetime" name="modified"/>

</xs:all>

</xs:complexType>

<xs:complexType name="SchemalessData">

<xs:sequence>

<xs:any minOccurs="0"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="ManagedEntity">

<xs:all>

<xs:element type="xs:string" name="uuid"/>

<xs:element type="xs:string" name="product"/>

<xs:element type="Address" name="address"/>

<xs:element type="Timestamp" name="time"/>

<xs:element type="RelationList" name="contains"/>

<xs:element type="RelationList" name="containedby"/>

</xs:all>

</xs:complexType>

<xs:complexType name="UnknownEntity">

<xs:complexContent>

<xs:extension base="ManagedEntity">

<xs:all>

<xs:element type="SchemalessData" name="RIMP"/>

</xs:all>

</xs:extension>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="iLO\_Entity">

<xs:complexContent>

<xs:extension base="ManagedEntity">

<xs:all>

<xs:element type="SchemalessData" name="RIMP"/>

</xs:all>

</xs:extension>

</xs:complexContent>

</xs:complexType>

</xs:schema>

### Retrieve the schema version of a managed type

Operation (implements 3.3.1):

GET http://<host>/entity/<mename>/version

Description:

Retrieve the schema version for the managed entity type. The version number for a managed entity type is a (major.minor) scheme. A change in the major version number represents an incompatible change to the data structure or meaning of the data. A change in the minor version number represents a minor change in the data structure that should still be compatible with the existing schema.

Result (xml or json):

The version number for the managed type definition.

Example:

**Request:**

GET http://localhost:50026/entity/ilo/schema

**Response:**

200 OK

<version>1.0</version>

### Retrieve a managed entity data model

Operation (implements 3.3.4):

GET http://<host>/entity/<mename>/<uuid>

Description:

Retrieve the data model for the managed entity.

Result (xml or json):

|  |  |  |  |
| --- | --- | --- | --- |
| **HTTP Result** | | **Data Returned** | |
| 200 OK | The data model for the managed entity is returned. | |
| 203 Non-Authoritative Information | | The data model for the managed entity is returned. The Database layer is still restoring this entity from the database, so the model may be out of date. | |
| 408 Timeout | | The data model is retuned. There was a communication error while restoring the entity from the database. The restore will be retried. | |
| 410 Gone | | No data is returned. There was an error restoring the entity from the database. The client should DELETE the entity and re-discover it if needed. | |

Note: The data model for the managed entity may reference external schemas. The client is responsible for knowing the referenced schema.

Example:

**Request:**

GET http://localhost:50026/entity/ilo/36363134-3635-5532-5837-313930335233

**Response:**

200 OK

<iLO\_Entity xmlns:ns1="http://hp.com/isb/commonservices/entity" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:iLO\_Entity">

<product>Integrated Lights-Out 2 (iLO 2)</product>

<time>

<created>2010-06-25 18:20:58.890000</created>

<accessed>2010-06-25 18:20:58.890000</accessed>

<modified>2010-06-25 18:20:58.890000</modified>

</time>

<RIMP>

<HSI>

<SBSN>2UX71903R3</SBSN>

<SPN>ProLiant BL460c G1</SPN>

<UUID>4166562UX71903R3</UUID>

<SP>1</SP>

<cUUID>36363134-3635-5532-5837-313930335233</cUUID>

<VIRTUAL>

<STATE>Inactive</STATE>

</VIRTUAL>

</HSI>

<MP>

<ST>1</ST>

<PN>Integrated Lights-Out 2 (iLO 2)</PN>

<FWRI>2.00</FWRI>

<BBLK>3; Jul 11 2006</BBLK>

<HWRI>ASIC: 7</HWRI>

<SN>ILO2UX71903R3</SN>

<UUID>ILO4166562UX71903R3</UUID>

<IPM>1</IPM>

<SSO>1</SSO>

<PWRM>0.5</PWRM>

</MP>

</RIMP>

<uuid>36363134-3635-5532-5837-313930335233</uuid>

<address>

<mac/>

<ipv4>16.83.121.23</ipv4>

</address>

</iLO\_Entity>

Common Services also allows client software to specify an optional object path to retrieve a portion of entity data model.

Example:

**Request:**

GET **[Error! Hyperlink reference not valid.](http://<host>/entity/server/<uuid)**>/data/summary

**Response:**

<Summary\_Data>

<networkName>localhost</networkName>

<manufacturer>HP</manufacturer>

<serialNumber>MXQ0190B4F</serialNumber>

<systemFirmware>03/30/2010, Family 386P64, Type 03</systemFirmware>

<systemType>64 bit</systemType>

<physicalMemorySize>4096</physicalMemorySize>

<totalDiskSize>-1</totalDiskSize>

<osType>VMware ESX 4.1</osType>

<osName>VMware ESX 4.1.0 build-307727</osName>

<managementSource>Insight Management Agents(SNMP) for HP ProLiant Systems</managementSource>

<managementVersion>8.6.0.18</managementVersion>

<iLOAddress>172.17.105.11</iLOAddress>

<ipv4Addresses>172.17.105.52</ipv4Addresses>

</Summary\_Data>

### Update a managed entity data model

Operation (implements 3.3.4):

PUT http://<host>/entity/<mename>/<uuid>

Description:

Update data in a managed entity’s data model. This API depends on the various entity controllers implementing relevant update methods for their data models. The data types accepted and returned for each call depend on the specific implementation.

Result:

Depends on the particular update call.

#### Update a managed entity’s credentials

PUT http://<host>/entity/<mename>/<uuid>/\_\_password

Description:

Update data in a managed entity’s credentials. The supplied credentials are tested and stored in the password database under the entity’s UUID if valid. Credentials may be passed either as clear text or as encrypted values.

<credentials>cleartext:username:password</credentials>

<credentials>ciphertext:base-64 encrypted username:base-64 encrypted password</credentials>

Result:

200 OK – Credentials updated and stored.

400 Bad Request – Credentials are invalid and not stored

Example:

**Request:**

PUT <http://localhost:50026/entity/ilo/36363134-3635-5532-5837-313930335233/__password>

<credentials>cleartext:Administrator:hpinvent</credentials>

**Response:**

200 OK

#### Update Server, vCenter or iLO entity data model

Note: extensible for OA and VC which are TBD

PUT http://<host>/entity/<mename>/<uuid>/\_\_updatenow

Description:

Update data in a managed entity’s data model. The update method is called on the declared entity. If the entity does not support the update method a .

Result:

200 OK – Data model is updated and stored

400 Bad Request – Credentials are invalid and not stored

Example:

**Request:**

PUT http://localhost:50026/entity/ilo/36363134-3635-5532-5837-313930335233/\_\_updatenow

**Response:**

200 OK

### Delete a managed entity data model

Operation (implements 3.3.4):

DELETE http://<host>/entity/<mename>/<uuid>

Description:

Delete the data model for the managed entity. If a password is stored in the password database under the entity’s UUID, the password record is automatically deleted.

Result (None):

The data model and control structures for the managed entity are destroyed.

Example:

**Request:**

DELETE http://localhost:50026/entity/ilo/36363134-3635-5532-5837-313930335233

**Response:**

200 OK

## Discovery

Common Services implements a discovery and identification service. The discovery service allows a client to discover and identify HP management devices at individual IP addresses or all HP management devices within an IP range.

Discovered devices are identified by the various data collection plugins. Devices for which no plugin is loaded will not be identified. For example, if the iLO plugin is not loaded, then iLO devices will not be identified.

### Discover an IP or IP range

Operation (implements 3.2.4):

POST http://<host>/discovery/<ipspec>

Description:

Discover and identify an IP address or IP address range. The address range may be specified in a number of formats:

* CIDR style: 192.168.10.0/24
* Address with Netmask: 192.168.10.0/255.255.255.0
* Range style: 192.168.10.1-192.168.10.254

The discovery API also allows the client to specify a list of plugins and parameters to restrict the type of discovery done. For example, if the client only wishes to discover iLOs in a given range, a discovery parameter may be used to limit the discovery to only iLO devices. Discovery parameters are given to common services as a list of plugins to use and a set of key/value parameters to pass to each plugin. See the DiscoveryParam and PluginParam nodes in entity.xsd for more details.

You may pass credentials with the discovery parameters instead of pre-populating the password database with credentials. Discovery will test the credentials and store them in the password database (under the device type and uuid) if a valid set of credentials is passed. If the credentials are invalid, discovery will fail with a 400 Bad Request and a NoCredentials error.

Result (xml or json):

For an individual address:

The discovered entity (200 OK, 203 Non-Authoritative Information

No entity at that address (404 Not Found)

NoCredentials, AuthError, PrivilegeError (400 Bad Request) Note: These errors also generate an event to the event queue.

For an address range:

The scan is queued (200 Ok)

Example 1: Discovery without Discovery Parameters

**Request:**

POST http://localhost:50026/discovery/192.168.1.1

**Response:**

200 OK

<iLO\_Entity xmlns:ns1="http://hp.com/isb/commonservices/entity" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:iLO\_Entity">

<product>Integrated Lights-Out 2 (iLO 2)</product>

<time>

<created>2010-06-25 18:21:58.031000</created>

<accessed>2010-06-25 18:21:58.031000</accessed>

<modified>2010-06-25 18:21:58.031000</modified>

</time>

<RIMP>

<HSI>

<SBSN></SBSN>

<SPN>ProLiant BL465c G1</SPN>

<UUID></UUID>

<SP>1</SP>

<cUUID>00000000-0000-0000-0000-000000000000</cUUID>

<VIRTUAL>

<STATE>Inactive</STATE>

</VIRTUAL>

</HSI>

<MP>

<ST>1</ST>

<PN>Integrated Lights-Out 2 (iLO 2)</PN>

<FWRI>1.80</FWRI>

<BBLK>3; Jul 11 2006</BBLK>

<HWRI>ASIC: 7</HWRI>

<SN>ILO</SN>

<UUID>ILO</UUID>

<IPM>1</IPM>

<SSO>0</SSO>

<PWRM>0.5</PWRM>

</MP>

</RIMP>

<uuid>00000000-0000-0000-0000-000000000000</uuid>

<address>

<mac/>

<ipv4>192.168.1.1</ipv4>

</address>

</iLO\_Entity>

Example 2: Discovery with Discovery Parameters

This example shows how to use discovery parameters to limit discovery to servers with SNMP agents or iLO devices.

**Request:**

POST <http://localhost:50026/discovery/192.168.1.1>

<parameters>

<plugin name=”hp.plugins.server”>

<arg>

<key>protocol</key>

<value>snmp</value>

</arg>

</plugin>

<plugin name=”hp.plugins.ilo”/>

</parameters>

**Response:**

200 OK

<iLO\_Entity> . . . </iLO\_Entity>

Example 3: Discovery with Discovery Parameters (credentials)

This example shows how to use discovery parameters to pass credentials during device discovery.

**Request:**

POST <http://localhost:50026/discovery/192.168.1.1>

<parameters>

<plugin name=”\*”>

<arg>

<key>credentials</key>

<value>cleartext:Administrator:password123</value>

</arg>

</plugin>

</parameters>

**Response:**

200 OK

<iLO\_Entity> . . . </iLO\_Entity>

Below is a list of key/value parameters currently supported by Common Services plugins:

* All plugins
  + Key: credentials
  + Value:
    - cleartext:username:password
    - ciphertext:<base-64 encoded encrypted username>:<base-64 encoded encrypted password>
  + Key: replace (replaces an existing entity with a newly discovered entity)
  + Value: true or false
* ProLiant Server plugin
  + Key: protocol,
  + Possible values: snmp, wbem
  + Key: interface
  + Possible values: in-band, out-of-band
* VCM plugin
  + Key: vcmContainer,
  + Possible values: c\_series, rack

Note about the “replace” parameter:

* When replace is true, and a new entity is found and replaces an existing entity, the HTTP status will be “201 Created”. The old entity will be destroyed and an EntityDestroyed event will be logged.
* When replace is true, but a new entity cannot be found to replace an existing entity, the HTTP status code will be “404 Not Found”. The existing entity will not be changed or destroyed.
* When replace is true, but no previous entity exists, no special status will be reported. The HTTP status code will either be “201 Created” or “404 Not Fount”.
* When replace is false (or not present), and an entity is created (because no previous entity exists), the HTTP status code will be “201 Created”
* When replace is false (or not present) and an entity already exists, the HTTP status code will be “200 Ok”.

### Query the Discovery Engine for status

GET **[Error! Hyperlink reference not valid.](http://<host>/discovery/engine/{optional)** IP address/hostname}

Description:

Retrieve information about addresses known to the discovery engine. The returned node structure contains discovery status of the node.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Address | The IP address of the node |
| Type | The type of the node (ilo, oa, etc) or nil for an undiscovered node. |
| Uuid | The uuid of the node or nil for an undiscovered node. |
| Retry | How many times the node has been retried. |
| Time | The UTC time of the last retry (seconds) |
| parameters | Any discovery parameters passed in to the original discovery call. |

Example:

**Request:**

GET http://localhost:50026/discovery/engine/172.17.3.30

**Response:**

200 OK

<DiscoveryNodes maxretries="5" interval="60">

<node>

<address>172.17.3.30</address>

<type>oa</type>

<uuid>53553930-3145-3330-3634-4a3000000000</uuid>

<retry>0</retry>

<time>1301352987</time>

<parameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>

</node>

</DiscoveryNodes>

### Query the Discovery DNS cache

GET **[Error! Hyperlink reference not valid.](http://<host>/discovery/dnscache/{optional)** IP address}

Description:

Retrieve information about addresses known to the discovery DNS cache. This call is provided primarily as a debug call to developers. The data returned is a schemaless list of key/bi-value pairs. The first value is the UTC time in seconds of when the entry was last refreshed. The second value is the hostname associated with the IP address.

Example:

**Request:**

GET http://localhost:50026/discovery/dnscache/172.17.3.30

**Response:**

200 OK

<dnscache>

<item>

<key>172.17.3.30</key>

<value>1303748075</value>

<value>vsr15e2oa1.virtsw.net</value>

</item>

</dnscache>

## Eventing

Common services provides a REST interface for retrieving asynchronous event notifications about entities managed by common services. The events may be internal Common Services events (intrinsic events) or SNMP and WBEM events (extrinsic events). Common Services may be configured to translate certain event types into NT eventlog entries. Events that have been converted are not available via the REST interface.

### Retrieving the Event Schema

The event schema may be retrieved from the event service.

Example:

GET http://<host>/event/schema

Description:

Retrieve the schema for Common Service events..

Result (xml or json):

The XML schema for Common Services Events (see share/events.xsd)

### Retrieve All Events

The REST interface provides a method to retrieve all events.

Example:

GET http://<host>/event/all

Result (xml or json):

The list of events.

<meEvents min="15" max="15">

<meEvent>

<eventID>15</eventID>

<eventName>hpcs.event.StartOfDay</eventName>

<eventTimeStamp>2011-03-09T21:27:58.984000</eventTimeStamp>

<eventText>Start of Day</eventText>

<eventCategory/>

<eventType>other</eventType>

<eventSeverity>normal</eventSeverity>

<eventReferencedDataModel/>

<eventReferencedManagedElementID/>

<eventSourceIP/>

<keyValuePairs/>

</meEvent>

</meEvents>

### Retrieve the Last Event

The REST interface provides a method to retrieve the last event.

Example:

GET http://<host>/event/last

Result (xml or json):

The list of events.

<meEvents min="15" max="15">

<meEvent>

<eventID>15</eventID>

<eventName>hpcs.event.StartOfDay</eventName>

<eventTimeStamp>2011-03-09T21:27:58.984000</eventTimeStamp>

<eventText>Start of Day</eventText>

<eventCategory/>

<eventType>other</eventType>

<eventSeverity>normal</eventSeverity>

<eventReferencedDataModel/>

<eventReferencedManagedElementID/>

<eventSourceIP/>

<keyValuePairs/>

</meEvent>

</meEvents>

### Retrieve a sequence of events

The REST interface provides a method to retrieve a sequence of events from an arbitrary event number to the last event.

Example:

GET http://<host>/event/15

Result (xml or json):

The list of events.

<meEvents min="15" max="15">

<meEvent>

<eventID>15</eventID>

<eventName>hpcs.event.StartOfDay</eventName>

<eventTimeStamp>2011-03-09T21:27:58.984000</eventTimeStamp>

<eventText>Start of Day</eventText>

<eventCategory/>

<eventType>other</eventType>

<eventSeverity>normal</eventSeverity>

<eventReferencedDataModel/>

<eventReferencedManagedElementID/>

<eventSourceIP/>

<keyValuePairs/>

</meEvent>

</meEvents>

## Virtual Connect

Common Services can discover and identify Virtual Connect domains and provide access to the Virtual Connect data model.

### Uniquely Identifying Virtual Connect Domains

Since Virtual Connect Domain can be made up of a collection of different Virtual Connect Modules spread across multiple C7000 enclosures, Virtual Connect Domains are uniquely identified by their Domain ID rather than by any specific hardware identifier. The Domain ID is randomly generated by Virtual Connect and will not change over the lifecycle of the domain.

### Items of Interest in the Virtual Connect Data Model

The complete virtual connect data model may be retrieved from the discovery service. The most interesting bits of information are in a structure named data (of type hpcs:VCM\_Data). The information is detailed below:

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Description** |
| info | hpvcm:DomainInformation | A DomainInformation structure containing data about Networks, Uplinks, Downlinks, etc. in this domain. |
| stats | hpvcm:EnetModulePort | Statistics for each of the relevant uplink ports. |
| externalManagerLockState | xs:string | Whether or not an external manager (like VCEM) has locked this domain. Will be one of “LOCKED”, “UNLOCKED”, “SUSPENDED” or “UNKNOWN” |
| externalManager | xs:string | The IP address of the external manager (if one exists). |

### Retrieving the Virtual Connect Data Model

The complete virtual connect data model may be retrieved from the discovery service. Please see section 4.6 for more details.

Operation:

GET http://<host>/discovery/vcdomain/<uuid>

Description:

Retrieve the data model for the managed entity.

Result (xml or json):

The data model for the managed entity. Note: The data model for the managed entity may reference external schemas. The client is responsible for knowing the referenced schema.

Example:

**Request:**

GET http://localhost:50026/discovery/vcdomain/VcD\_0123456789ab

**Response:**

200 OK

<VirtualConnect\_Entity xmlns:ns1="http://hp.com/isb/commonservices/entity" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:iLO\_Entity">

...

</VirtualConnect\_Entity>

### Retrieving a subset of the Virtual Connect data model for a specific host

The Virtual Connect service also allows a client to retrieve a subset of the Virtual Connect data model that applies only to a specific host. This will allow the client to get the list of modules, ports, networks and uplinks that apply only to a given host (e.g. If a Virtual Connect domain has 15 networks defined, but only 4 are mapped to the given server, then only those 4 networks will be returned by this call).

Operations:

GET http://<host>/vcservice/<host-uuid>

GET http://<host>/vcservice/schema

GET http://<host>/vcservice/version

Description:

Retrieve the virtual connect data model relevant to the specified host.

Result (xml or json):

A VCM\_ServerInfo object containing the parts of the Virtual Connect data model that apply to the given host.

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Description** |
| info | hpvcm:DomainInformation | A DomainInformation structure containing data about Networks, Uplinks, Downlinks, etc, relevant to this host. |
| stats | hpvcm:EnetModulePort | Statistics for each of the relevant uplink ports. |

Example:

**Request:**

GET http://localhost:50026/vcservice/ 36363134-3635-5532-5837-313930335233

**Response:**

200 OK

<VCM\_ServerInfo xmlns:ns1="http://hp.com/isb/commonservices/entity" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:iLO\_Entity">

...

</VCM\_ServerInfo>

## Control Verbs

Partner Common Services – by and large – is not a control platform. There are a some instances where control exists to provide solutions to certain problems, and are enumerated here.

### iLO Control

#### Set Device Power State

The power state of an iLO managed entity may be set to a specific state – either ON or OFF.

Operation:

PUT http://<host>/entity/ilo/<uuid>/power/[on|off]

Description:

Set the power state of the entity. Nothing occurs if the device is already in the specified state.

Result (xml or json):

None

Example:

**Request:**

PUT http://localhost:50026/entity/ilo/deadbeef-dead-beef-dead-beefdeadbeef/power/on

**Response:**

200 OK

#### Set UID State

The UID LED indicator can be set to ON or OFF.

Operation:

PUT http://<host>/entity/ilo/<uuid>/uid/[on|off]

Description:

Set the state of the UID indicator.

Result (xml or json):

None

Example:

**Request:**

PUT http://localhost:50026/entity/ilo/deadbeef-dead-beef-dead-beefdeadbeef/uid/on

**Response:**

200 OK

#### Clear Event Logs

Clears the iLO event log. This command takes no paraemeters.

Operation:

PUT http://<host>/entity/ilo/<uuid>/clear\_eventlog

Description:

Clears the iLO event log.

Result (xml or json):

None

Example:

**Request:**

PUT http://localhost:50026/entity/ilo/deadbeef-dead-beef-dead-beefdeadbeef/clear\_eventlog

**Response:**

200 OK

### OA Control

#### Set LCD User Notes

The OA console provides for setting six lines of user notes for the chassis’ LCD display. Up to six lines may be specified, with blank lines being defaulted for unspecified lines.

Operation:

PUT http://<host>/entity/oa/<uuid>/set\_lcd\_user\_notes/line1/line2/.../line6

Description:

Sets the LCD user notes on the OA chassis.

Result (xml or json):

None

Example:

**Request:**

PUT <http://localhost:50026/entity/oa/deafbeef-dead-beef-dead-beefdeadbeef/set_lcd_user_notes/spam> eggs/bacon and spam/

**Response:**

200 OK

# 

# Functional Description

Below is a high level block diagram of Partner Common Services.



## Startup Process

This section describes the startup process for Common Services.

* Logging is configured from share/logging.xml.
* The configuration is read from share/config.xml.
* Any debug options in the configuration are set (e.g. web/debug).
* The SQLite database is configured.
* The password information is read from share/password.xml.
* The event subsystem is configured.
* SNMP and WBEM listeners are started.
* All enabled plugins are loaded.
* Individual plugin configurations are merged with their the plugin default config.
* The configuration is saved back to config.xml.
* All plugins are started.
* The runtime catalog is populated with NullControllers for all known entities in the database. NullControllers handle responding to REST calls while the real controllers are created from their corresponding data model in the database.
* The web server is started (the REST API is ready)
* All data models in the database are restored (this happens in a background thread and may take some time).
* The entity scheduler is started.

## Services and Plugins

This section describes the services and plugins, and provides additional details regarding how they work in order to provide clients with a better understanding of what to expect from the service.

### Configuration Service

The Configuration service allows a Common Services client to query and modify the configuration of the Common Services.

The Common Services configuration is stored in share/config.xml. Common Services will read this file at startup to learn its configuration. If the file is missing, Common Services will assign reasonable default values as indicated below and create config.xml.

A partner application (such as RM or Br) that installs Common Services may pre-populate the config.xml file.

*Placeholder if needed – Above content is duplicated from the Interface section. Any additional info needed here that’s not already covered in the Interface section for Configuration Service?*

### Password Service

The password service allows a Common Services client to manage the password database used by Common Services to log in to other managed devices, such as iLO, OA or Virtual Connect modules.

*Placeholder if needed – Above content is duplicated from the Interface section. Any additional info needed here that’s not already covered in the Interface section for Password Service?*

### Discovery Service

Common Services implements a discovery and identification service. The discovery service allows a client to discover and identify HP management devices at individual IP addresses or all HP management devices within an IP range.

Discovered devices are identified by the various data collection plugins. Devices for which no plugin is loaded will not be identified. For example, if the iLO plugin is not loaded, then iLO devices will not be identified.

There is an “UnknownPlugin” which will claim all devices not claimed by other plugins. The UnknownPlugin also implements a <metype> of “any” which allows a client to request or enumerate managed entities regardless of their entity type.

The discovery service is driven by a discovery engine. Information about discovered and undiscovered nodes is stored in the persistent database and discovery can be retried on undiscovered nodes on a periodic interval. If the client specifies parameters with a call to the discovery API, those parameters are remembered and used on subsequent discovery attempts.

The number of retries and the interval between retries are specified in the <discovery> node of the configuration file. The number of threads employed by the discovery engine is also specified in the configuration.

The discovery engine will attempt DNS lookups on all IP addresses and hostnames that it encounters. Whenever possible, DiscoveryNodes are identified by their fully qualified domain name. DiscoveryEngine will attempt to perform reverse-DNS lookups on nodes encountered by IP address and will store the hostname in the address field of the DiscoveryNode.

The DiscoveryEngine also maintains a DNS cache that maps IP addresses to hostnames. Clients using the REST API may specify either the hostname or IP address to the REST calls. The DiscoveryEngine will automatically map known IP addresses to their hostnames and return the correct node. If an IP address cannot be resolved into a DNS name, the DiscoveryNode will be stored and identified with the IP address.

### Event Service

The event service provides an event queue within Common Services to collect events from the various services and plugins and provides a REST API for clients to retrieve those events.

#### Event Expiration

The event service maintains the event queue in persistent storage and will periodically purge old events. Events are purged after a client-configurable amount of time. The event timeout is specified in the confg.xml file with the <event/timeout> node. The timeout is expressed in minutes.

Example:

<event>

<timeout>15</timeout>

</event>

#### Redirecting events to the NT event log

The event service may be configured to log certain events to the NT event log rather than its own internal queue. When in this configuration, extrinsic events specified by the MessageEntities.xml file are sent to the NT event log rather than the internal event queue. This configuration option is specified in the config.xml file. The <eventRedirect> node specifies to which Windows Event Log to send the events. The <eventSpecFile> node specifies the file path of the event meta-data.

Example:

<event>

<redirect>HP Device Monitor</redirect>

<specfile>c:\path\to\MessageEntities.xml</specfile>

</event>

It is the client’s responsibility to insure that the Windows event sources and message catalog files exist. Common Services will neither create, nor check for the existence if the specified event sources.,

It is also the client’s responsibility to ensure the appropriate fields in the MessageEntities.xml file are correctly populated. Common Services will leverage a subset of the MessageSource content to format the incoming SNMP and WBEM events. Note that he PayloadIDs node can be left unpopulated. If unpopulated Common Services will apply a default set of rules to the SNMP and WBEM events. If populated Common Services will override the default rule set and replace with the ‘Source’ and ‘Name’ content where ‘Source’ represents the Event Source (such as an overriding CIM class) and ‘Name’ represents the event number or a varbind.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **MessageSource SNMP** | | | | **MessageSource WBEM** | | | |
| Facility | | | | Facility | | | |
| Language | | | | Language | | | |
| Message Definitions | LookupIDs | Id (alert number) | | Message Definitions | LookupIDs | Id (CIM Class Name) | |
| Id (Indication Number) | |
| MessageID | | | MessageID | | |
| Severity | | | Severity | | |
| PayloadIDs | id | source | PayloadIDs | id | source |
| name | name |

#### Conversion of events into NT event log entries

When Common Services converts an event into an NT event log entry, it first verifies that the event (SNMP trap or WBEM indication) exists in the event specification XML file. If it does not exist, the event is dropped (unless unknownExtrinsic is set to “keep”).

Events are then converted to NT event with the ID specified in the specification file. The NT event string parameters are used to communicate all of the event data. The string parameters are:

1. The event source IP address.
2. The corresponding entity UUID.
3. The event source hostname (if known), or the IP address. For server type events, this may be the cached hostname from the server data model.
4. The rest of the parameters are SNMP varbinds or WBEM indication data.

### ProLiant Server Plugin

The following sections provide additional detail regarding design and implementation of the ProLiant Server Plugin.

#### ProLiant Server Data Collection

ProLiant Server Data Collection enables the collection and caching of summary, health and inventory data for ProLiant managed nodes that support either SNMP or WBEM agents. This data supports the REST interface defined in Section 3.1 of the manageREST specification.

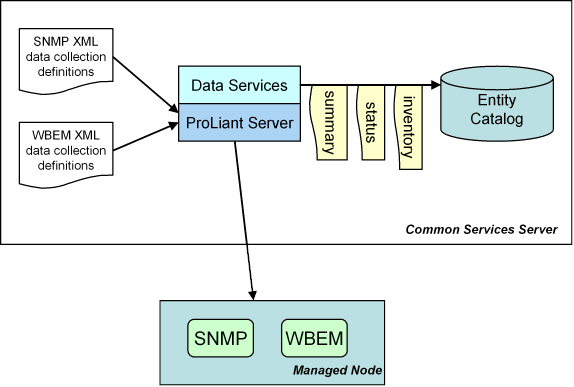


Figure : Common Services Data Collection

##### Normalized Inventory Summary Data

ProLiant Server summary data will be normalized to ensure consistent data values and meaning regardless of the agent source. The following table describes the server summary data that will be collected from SNMP or WBEM agent sources.

|  |  |  |
| --- | --- | --- |
| **Common Services**  **Inventory Summary**  **Property** | **SNMP Inventory**  **Source** | **WBEM Inventory**  **Source** |
| Network name | cpqHoSystemName  (1.3.6.1.4.1.232.11.2.2.12) | SMX\_ComputerSystem. ElementName |
| Manufacturer | Hardcoded to ‘HP’ | SMX\_ComputerSystemChassis.  Manufacturer |
| Model | cpqSiProductName  (1.3.6.1.4.1.232.2.2.4.2) | HP\_ComputerSystem  .model |
| Serial number | cpqSiSysSerialNum  (1.3.6.1.4.1.232.2.2.2.1) | HP\_ComputerSystemChassis  .SerialNumber |
| System firmware | cpqSeSysRomVer  (1.3.6.1.4.1.232.1.2.6.1) | CIM\_SoftwareIdentity?? |
| System type | SNMPv2-MIB::sysDescr  (1.3.6.1.2.1.1.1)  Caculate: if system description indicates OS is x86\_64, return ‘64 bit’. Otherwise, return ‘32 bit’(3). | CIM\_OperatingSystem. Version |
| Physical memory size | cpqSeTotalMem  (1.3.6.1.4.1.232.1.2.3.2)  Calculate:convert memory size from KB to MB | **Calculate**:  HP\_MemoryModule.Capacity |
| Total disk | N/A, hardcoded to -1 | N/A SMX\_SAPrimordialPool.  TotalManagedSpace |
| OS Name | cpqHoDesc(1)  (1.3.6.1.4.1.232.11.2.2.3) | CIM\_OperatingSystem.ElementName |
| OS Type | cpqHoName(1)  (1.3.6.1.4.1.232.11.2.2.1)  cpqHoVersion(1)  (1.3.6.1.4.1.232.11.2.2.2)  Calculate: convert host OS name and version to one of the pre-defined OSType enumeration strings. | CIM\_OperatingSystem.Caption |
| OS Version | cpqHoDesc(1)  (1.3.6.1.4.1.232.11.2.2.3)  sysDescr  (1.3.6.1.2.1.1.1)  Calculate: if ‘VMware’ is in cpqHoDesc, get build number (build-xxxxx) from cpqHoDesc and return; if ‘Linux’ is in cpqHoDesc, get kernel version from sysDescr (2.6.xx-xx) and return(3). | CIM\_OperatingSystem.osVersion |
| HP Management Source | For inband management, hardcoded to ‘Insight Management Agents(SNMP) for HP ProLiant Systems’. For OOB, return iLO4 or iLO4 and AMS product name. | Hardcoded to ‘Insight Management Agents(WBEM) for HP ProLiant Systems’ |
| HP Management Version | cpqHoSwVerAgentsVer(1)  (1.3.6.1.4.1.232.11.2.7.3) | SMX\_SoftwareIdentity (ElementName=”hp-smx”, use RevisionNumber from this instance) |
| HP Management Interface | SNMPv2-MIB:sysDesc (1.3.6.1.2.1.1.1)  Calculate: if ‘Integrated Lights-Out 4’ is in sysDescr, return ‘out-of-band’. Otherwise, return ‘in-band’. | Return ‘INB’. |
| iLO Address | cpqSm2NicIpAddress (1.3.6.1.4.1.232.9.2.5.1.1.5) | SMX\_ManagementProcessor.iLOAddress |
| IPv4 Address | IP-MIB::ipAdEntAddr(1) (1.3.6.1.2.1.4.20.1.1) | SMX\_IPProtocolEndpoint.ipv4Addresses |
| iLO Firmware Version | cpqSm2CntlrRomRevsion  (1.3.6.1.4.1.232.9.2.2)  cpqSm2CntlrRomDate  (1.3.6.1.4.1.232.9.2.1) |  |
| iLO Hostname | In-band: cpqSm2NicRibFullQualDnsName (1.3.6.1.4.1.232.9.2.5.1.1.14)  Out-of-band: SNMPv2-MIB:sysName  (1.3.6.1.2.1.1.5) |  |
| Server Hostname | In-band: SNMPv2-MIB:sysName  (1.3.6.1.2.1.1.5)  Out-of-band:cpqHoSystemName  (1.3.6.1.4.1.232.11.2.2.12) |  |
|  |  |  |
| System UUID | cpqHoGUIDCanonical  (1.3.6.1.4.1.232.11.2.10.6) | HP\_ComputerSystem.  OtherIdentifyingInfo.CIM:GUID |

Table : Inventory Summary Data

1. For Gen8 servers with iLO4 OOB management agent, this data source requires AMS helper running on host OS. Without AMS helper, iLO4 alone is not able to provide this data.
2. RM will treat the MP as another subcomponent. CS must collect the UUID from the iLO and stored as an element on the server object.
3. For Gen8 servers with iLO4 OOB management, SNMPv2-MIB::sysDescr returns iLO4 system info instead of host OS info. As a result, we will not be able to get ‘System type’ (32-bit vs. 64-bit) or ‘OS Version’ (kernel version) for Linux servers through OOB.

##### Normalized Health Status Values

Health values collected from SNMP or WBEM agents will be normalized to a common set of health status values and meanings. The mapping below reflects the agreed upon mapping that will be used across ESSN manageability applications.

|  |  |  |
| --- | --- | --- |
| **Common Services Status** | **SNMP Status** | **WBEM OperationalStatus** |
| Unknown | Not Available (0)  Unknown (1) | Other(1)  Starting(8)  Stopping(9)  In Service(11)  Dormant(15)  Power Mode(18)  Unknown(0)  No Contact(12) |
| Normal | OK (2) | OK(2)  Completed(17) |
| Minor | Degraded (3) | Degraded(3)  Stressed(4)  Stopped(10) |
| Major | Failed (4) | Predictive Failure(5)  Error(6)  Aborted(14)  Supporting Entity in Error(16)  Non-Recoverable Error(7)  Lost Communication(13) |

Table : Normalized Health Status Values for Collections

##### Health Status Data Sources

|  |  |  |  |
| --- | --- | --- | --- |
| **Common Services Subcomponent** | **ProLiant Pre-SS9.0 SNMP Source** | **ProLiant Post-SS9.0 SNMP and Gen8 OOB Source** | **WBEM Source** |
| HPServer | Aggregate subsystem status collected below | cpqHoMibHealthStatusArray Element 0 | HP\_ComputerSystem.  OperationalStatus |
| HP ProLiant  Memory | cpqHeResilientMemCondition (1.3.6.1.4.1.232.6.2.14.4) | cpqHoMibHealthStatusArray Element 2 | HP\_MemoryCollection.  GroupOperationalStatus |
| HPProLiant  Processors | N/A, hardcoded to “unknown” | cpqHoMibHealthStatusArray Element 1 | HP\_ProcessorCollection.  GroupOperationalStatus |
| HPProLiant  Cooling | cpqHeThermalSystemFanStatus (1.3.6.1.4.1.232.6.2.6.4) | cpqHoMibHealthStatusArray Element 3 | HP\_FanCollection.  GroupOperationalStatus |
| HP ProLiant ASR | cpqHeAsrCondition (1.3.6.1.4.1.232.6.2.5.17) | cpqHoMibHealthStatusArray Element 7 | N/A |
| HPProLiantTemperature  Sensors | cpqHeThermalTempStatus (1.3.6.1.4.1.232.6.2.6.3) | cpqHoMibHealthStatusArray Element 4 | HP\_SensorCollection.  GroupOperationalStatus |
| HPProLiant  Power Supplies | cpqHeFltTolPwrSupplyCondition (1.3.6.1.4.1.232.6.2.9.1) | cpqHoMibHealthStatusArray Element 5 | HP\_PowerCollection.  GroupOperationalStatus |
| HPProLiant  Server Storage | Aggregate cpqHoMibStatusArray element 3, 5, 8, 14, 16 for Smart Array, SCSI, external storage, IDE and fibre channel MIB overall condition | Aggregate cpqHoMibHealthStatusArray Element 8, 9, 10, 11, 12 | ?? SAStorageSpecificCollection  FCHBAGroupSystemSpecific  Collection |
| HPProLiant  Server Networks | cpqHoMibStatusArray element 18 for Network MIB overall condition | cpqHoMibHealthStatusArray Element 13 | HP\_EthernetCollection.  GroupOperationalStatus |
| HPProLiant  Management Processors | cpqHoMibStatusArray element 9 for iLO MIB overall condition | cpqHoMibHealthStatusArray Element 14 | HP\_MPCollection.  GroupOperationalStatus |
| HPProLiant  Logs | cpqHeEventLogCondition (1.3.6.1.4.1.232.6.2.11.2) | cpqHoMibHealthStatusArray Element 6 | Not available |

Table : Health Status Sources

HPServer – status array element 0 is the worst-case roll up of the status array and will not reflect mibs not included in the array (cooling, power, etc…). Should we roll-up all status values in the table?

[TR] My recommendation for SNMP is to perform our own roll-up of status. Element 0 of the existing SNMP health array will not reflect the overall status of some of the subsystems we intend to collect above and beyond those that are represented in the array elements.

##### Normalize Inventory Detailed data for Breckenridge

ProLiant Server inventory data will be normalized to ensure consistent data values and meaning regardless of the agent source. The following table describes the server inventory data that will be collected from SNMP or WBEM agent sources for Breckenridge consumption.

|  |  |  |
| --- | --- | --- |
| **Common Services Detailed Inventory Property** | **SNMP and Gen8 OOB Inventory Source** | **WBEM Inventory Source** |
| **FC HBA** |  |  |
| Model | cpqFcaHostCntlrModel(1)  (1.3.6.1.4.1.232.16.2.7.1.1.3) | SMX\_FCHBAFCPortController.  ElementName |
| Manufacturer | Hard code to N/A | SMX\_FCHBAPhysicalPackage.  Manufacturer |
| Serial Number | cpqFcaHostCntlrSerialNumber(1)  (1.3.6.1.4.1.232.16.2.7.1.1.10) | SMX\_FCHBAPhysicalPackage.  SerialNumber |
| Location | cpqFcaHostCntlrSlot(1)  (1.3.6.1.4.1.232.16.2.7.1.1.2) | SMX\_FCHBAPhysicalPackage.  OtherIdentifyingInfo.Location |
| WWPortName | cpqFcaHostCntlrWorldWidePortName(1)  (1.3.6.1.4.1.232.16.2.7.1.1.12) | SMX\_FCHBAFCPort.  PermanentAddress |
| WWNodeName | cpqFcaHostCntlrWorldWideName(1)  (1.3.6.1.4.1.232.16.2.7.1.1.6) | SMX\_FCHBAFCPort.  SystemName |
| Speed | N/A | SMX\_FCHBAFCPort.  Speed |
| **Smart Array** |  |  |
| **SA Controller** |  |  |
| CntlrName | cpqDaCntlrModel  (1.3.6.1.4.1.232.3.2.2.1.1.2)  cpqDaCntlrHwLocation  (1.3.6.1.4.1.232.3.2.2.1.1.20) | SMX\_SAArraySystem.  ElementName |
| Manufacturer | Hardcoded to ‘HP’ | SMX\_SAPhysicalPackage.  Manufacturer |
| Model | cpqDaCntlrModel  (1.3.6.1.4.1.232.3.2.2.1.1.2)  Calculate: map cpqDaCntlrModel (an integer) to HP Smart Array controller marketing name | SMX\_SAPhysicalPackage.  Model |
| Serial Number | cpqDaCntlrSerialNumber  (1.3.6.1.4.1.232.3.2.2.1.1.15) | SMX\_SAPhysicalPackage.  SerialNumber |
| FirmwareVersion | cpqDaCntlrFWRev  (1.3.6.1.4.1.232.3.2.2.1.1.3) | SMX\_SAFirmware.  VersionString |
| **SA Logical Volume** |  |  |
| LogDrvName | ‘Logical Volume’ + cpqDaLogDrvIndex  (1.3.6.1.4.1.232.3.2.3.1.1.1) | SMX\_SAStorageVolume.  ElementName |
| Capacity | cpqDaLogDrvSize  (1.3.6.1.4.1.232.3.2.3.1.1.9) | SMX\_SAStorageVolume.  Capacity |
| StripeSize | cpqDaLogDrvStripeSize(1)  (1.3.6.1.4.1.232.3.2.3.1.1.13), in KB | SMX\_ SAStorageVolume.  StripeSize |
| FaultTolerance | cpqDaLogDrvFaultTol  (1.3.6.1.4.1.232.3.2.3.1.1.3) | SMX\_ SAStorageVolume.  FaultTolerance |
| **SA Physical Drive** |  |  |
| PhysDrvName | cpqDaPhyDrvLocationString  (1.3.6.1.4.1.232.3.2.5.1.1.64) | SMX\_SADiskDrive.ElementName |
| PhysDrvSerialNumber | cpqDaPhyDrvSerialNum  (1.3.6.1.4.1.232.3.2.5.1.1.51) | SMX\_SADiskDrive.SerialNumber |
| Size | cpqDaPhyDrvSize  (1.3.6.1.4.1.232.3.2.5.1.1.45), in MB | SMX\_SAStorageExtent.  NumberOfBlocks |
| Interface | cpqDaPhyDrvType  (1.3.6.1.4.1.232.3.2.5.1.1.60)  Both SNMP and WBEM return an integer for drive interface, need to map integer to a pre-defined enumeration strings | SMX\_SADiskDrive.  ElementName |
| NegotiatedSpeed | cpqDaPhyDrvCurrentSpeed(1)  (1.3.6.1.4.1.232.3.2.5.1.1.54) | SMX\_SADiskDrive.  NegotiatedSpeed |
| **Firmware** |  |  |
| Name | cpqHoFwVerDisplayName (1.3.6.1.4.1.232.11.2.14.1.1.4) | CIM\_SoftwareIdentity.  IdentityInfoValue.softwarefamily |
| Version | cpqHoFwVerVersion (1.3.6.1.4.1.232.11.2.14.1.1.5) | CIM\_SoftwareIdentity.  VersionString |
| Description | N/A | CIM\_SoftwareIdentity.  Caption |
| **Software** |  |  |
| Name | cpqHoSwVerName(2) (1.3.6.1.4.1.232.11.2.7.2.1.4) | CIM\_SoftwareIdentity.  IdentityInfoValue.softwarefamily |
| Version | cpqHoSwVerVersion(2) (1.3.6.1.4.1.232.11.2.7.2.1.8) | CIM\_SoftwareIdentity.  VersionString |
| Description | cpqHoSwVerDescription(2) (1.3.6.1.4.1.232.11.2.7.2.1.5) | CIM\_SoftwareIdentity.  Caption |
| **MP** |  |  |
| Model | cpqSm2CntlrModel (1.3.6.1.4.1.232.9.2.2.21) | SMX\_ManagementProcessor.  ControllerType |
| URL | N/A | SMX\_ManagementProcessor.  URL |
| IPAddress | cpqSm2NicIpAddress (1.3.6.1.4.1.232.9.2.5.1.1.5) | SMX\_ManagementProcessor.  IPAddress |
| SerialNumber | cpqSm2CntlrBoardSerialNumber  (1.3.6.1.4.1.232.9.2.2.15) | SMX\_ManagementProcessor.  UniqueIdentifier |
| **NIC** |  |  |
| Adapter Name | cpqNicIfPhysAdapterName (1.3.6.1.4.1.232.18.2.3.1.1.1.39) | SMX\_EthernetPort.  ElementName |
| MAC Address | cpqNicIfPhysAdapterMACAddress  (1.3.6.1.4.1.232.18.2.3.1.1.4) | SMX\_EthernetPort.  PermanentAddress |
| Firmware Version | cpqNicIfPhysAdapterFWVersion (1.3.6.1.4.1.232.18.2.3.1.1.41) | SMX\_EthernetPortControllerFirmwareIdentity.VersionString |

1. This data source is not supported in iLO4 OOB environment.
2. This data source is supported in iLO4 OOB environment but requires AMS helpers running on host OS.

##### Cached Server Data

Common Services currently cache server host name. The goal is to provide this key information to client software even after management agent has been shutdown. Common Services updates the cached data whenever it gets new valid data from data source. If no new data is available, Common Services will return the cached value back to client.

#### ProLiant Server Configuration Requirement

##### Common Services configuration

Client software needs to modify Common Services configuration file (config.xml) to enable ProLiant server support and optimize performance.

To enable ProLiant server management support, “server” plugin needs to be added in Common Services configuration file:

Example:

<plugins>

<plugin name=”hp.plugins.server”/>

</plugins>

Since target server may support various types of management protocols, client software can modify “primaryprotocol” setting to optimize server discovery process. For example, if the majority of target servers are running ESXi with HP WBEM providers, client can set “primaryprotocol” to “wbem”:

Example:

<plugin name=”hp.plugins.server”>

<server>

<primaryprotocol>wbem</primaryprotocol>

</server>

</plugin>

By default, Common Services starts SNMP trap listener on port 162. Client software can choose to use another port by changing the configurable flag “snmpTrapPort” in config.xml:

Example:

<collector>

<snmpTrapPort>162</snmpTrapPort>

</collector>

NOTE: Common Services can be configured to use the Windows SNMP API to retrieve traps. If you need to use the Windows SNMP API rather than the built-in SNMP implementation for retrieving traps, set the snmpTrapPort to -1.

By default, Common Services starts the WBEM indication listener on port 50028. Client software can choose to use another port by changing the configurable flag “wbemport” in config.xml:

Example:

<collector>

<wbemport>50028</wbemport>

</collector>

If a port other than the default port is selected existing WBEM subscriptions will no longer work. New subscriptions must be created since the subscription embeds the WBEM listener port number.

In Common Services configuration file, there are several configurable polling intervals that control how often Common Services poll each target server node to update server’s summary, status and detailed inventory data. Client software can configure any of the following intervals to optimize polling performance:

Example:

<plugin name=”hp.plugins.server”>

<server>

<summaryinterval>15</summaryinterval>

<stautsinterval>15</statusinterval>

<inventoryinterval>100</inventoryinterval>

</server>

</plugin>

<schedulrer>

<interval>5</interval>

</scheduler>

##### Target SNMP server configuration

For Common Services to detect and monitor Linux and ESX servers through SNMP protocol, the target servers need to set up SNMP community string and trap sink correctly in SNMPconfiguration file (/etc/snmp/snmpd.conf):

Example:

rocommunity <read-only community string> <Common Services IP address>

trapsink <Common Services IP address>

In the above example, the first line of configuration sets up a read-only community string with access limited to the server where Common Services is running on. The second line sets up trap sink to send SNMP traps to Common Services.

##### Target WBEM server configuration

Target WBEM servers do not require any specialized configurations to work with Common Services clients.

#### ProLiant Server Discovery

Client software can use Common Services Discovery API to discover a ProLiant server at a given IP address. Server discovery within an IP range is not officially supported and tested at this time. The discovery API allows user to pass in an optional “protocol” parameter to optimize discovery performance by only using SNMP or WBEM protocol. If no protocol is specified, the discovery process will first try to identity the server with the primary protocol as defined in config.xml. If not successful, the discovery process will try again with the secondary protocols.

##### SNMP server discovery

For Common Services to discover an SNMP server, client software does not need to have target SNMP server’s root credential. Instead, client software needs to use password service to set up SNMP community string as a new “password”:

Example:

Request:

POST <http://localhost:50026/password>?lang=json

{

"username": "\*",

"host": "<Common Servcies IP address>",

"type": "SNMP Community String",

"password": "<read-onnly community string>",

"id": "new"

}

Client software can also set up a default community string for any server that does not have a server specific community string:

Example:

Request:

POST [http://localhost:50026/password](http://hp.com/password)?lang=json

{

"username": "\*",

"host": "\*",

"type": "SNMP Community String",

"password": "<default read-only community string>",

"id": "new"

}

To discover a SNMP server target, the discovery algorithm first detects server’s UUID and management interface:

* Check server UUID (cpqHoGUIDCanonical)
* Check system description (MIB2 sysDescr). If sysDecr contains substring ‘Integrated Lights-Out 4’, then the management interface is out-of-band; otherwise, in-band.

For in-band management, SNMP server discovery process uses the following additional algorithm to ensure that only ProLiant servers running ESX or Linux are detected:

* Check operating System type (cpqHoDesc) and make sure it contains substring ‘VMware’ or ‘Linux’.
* Check server model (cpqSiProductName) and make sure it contains substring ‘ProLiant’.

Current discovery algorithm does not check if OS or SNMP agent versions are in support matrix.

##### WBEM server discovery

WBEM discovery in Common Services will rely on 2 mechanisms to create a CIM connection:

* ESXi hosts in lockdown mode must be managed through a vCenter. During WBEM discovery and data collection, the ProLiant Server plugin will contact the vCenter via the TinyVCenter Plugin and request a security ticket. The security ticket will be used to make the CIM connection.
* ESXi hosts not in lockdown mode can be discovered using OS based login credentials. In order for a WBEM subscription to be successfully created, and WBEM indications received and logged, the root credential must be provided.

Example:

Request:

POST <http://localhost:50026/password>?lang=json

{

"username": <host based username or root>,

"host": "<Common Servcies IP address>",

"type": "WBEM",

"password": "<host based password or root password>",

"id": "new"

}

WBEM server discovery will only run against ESXi servers. WBEM based discovery requires the HP Insight Management WBEM Providers for HP integrated VMWare ESXi. WBEM based discovery will fail if these providers are not present.

#### Server Entity Status

During the discovery and data collection tasks the ProLiant Server entity’s overall status will be set based on ability to complete the task

Server entity status will be set to one of the following status values:

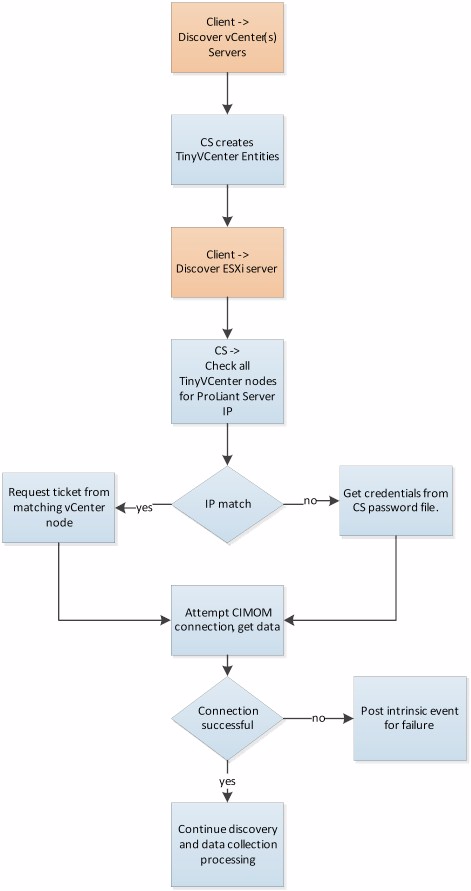
“normal”: Common Services completes the discovery and data collection tasks successfully

“major” : Common Services is unable to contact the managed node’s agents or is unable to collect a complete set of data from the managed node.

### Tiny VCenter Plugin

The Tiny VCenter Plugin helps managed ESXi WBEM based servers that are in lock-down mode. When an ESXi server is in lock-down mode WBEM authentication must be made using a security ticket rather than login credentials. The VCenter host provides SOAP API calls to allow ticket requests for those ESXi servers managed within the vCenter.

The Tiny VCenter Plugin allows the CS client to discover the vCenter installations in their domain. When CIM data requests are made by an ESXi server, the Tiny VCenter Plugin iterates over each of the discovered vCenter servers until a matching IPv4 address is found in the vCenter’s managed host collection or until all discovered vCenter servers have been examined. In the event a managing vCenter server is not identified the default login credentials will be used to attempt a CIM Connection.



#### Tiny VCenter Identification

TinyvCenter plugin discovery relies on the HTML response document to first identify the vCenter serve as a “VMWare” product. If the string “VMWare” is found in the HTML document discovery continues and contacts the vCenter server over SOAP to get the ‘about’ information. In the ‘about’ information the ‘licenseProductName’ is checked for the string “VMware VirtualCenter Server”. If this check passes then a TinyVCenter\_Controller is instantiated and the collection of managed hosts, the managed host UUID and vCenter object references are collected and cached.

#### Ticket requests

Discovery or data collection activities on an ESXi server will result in the search for a managing vCenter server. If a managing vCenter server is identified a request for a security ticket is initiated. The TinyVCenter\_Controller will then contact the managing vCenter server and request the security ticket by calling the SOAP API AcquireCimServicesTicket().

#### vCenter Entity Status

During the discovery and data collection tasks the vCenter entity’s overall status will be set based on ability to complete the task.

vCenter entity status will be set to one of the following status values:

“normal”: Common Services completes the discovery and is able to collect the list of servers managed by the vCenter

“major” : Common Services is unable to contact the managed node’s SOAP API’s and is unable to collect or refresh the list of servers managed by the vCenter

### Entity Reconnect Algorithm

The reconnect algorithm allows an entity to gradually extend the scheduling time when the entity has achieved a status of **CRITICAL** and has exceeded the user configurable reconnect count.

When an entity is not reachable for a configurable number of sequential connection attempts the entity’s status is set to **CRITICAL** and the reconnect algorithm is activated. The reconnect algorithm extends the next schedule time based on the number of attempted data model updates that have failed. Over time the next schedule time becomes progressively longer.

If a user intervenes and takes corrective action on the managed entity and runs \_\_updatenow (see section 3.3.6.2) the reconnect algorithm counters are reset and the entity status is set to **NORMAL**.

The backoff cap is calculated based on the current clock time when the reconnect attempts have been exhausted + the backoffcap parameter (by default the backoffcap is set to 5760 minutes or 4 days). The reconnect algorithm terminates scheduling once the clock has reached the backoff cap and the retryforever flag is set to False.

Once the backoff cap is exceeded, and if the retryforever flag is False, the managed entity is placed in **DISABLED** state. When the user takes corrective action the managed entity must be deleted and rediscovered in order for periodic scheduling to restart.

If the backoff cap is exceeded but the retryforever flag is True then the entity will remain in a **CRITICAL** state and will continue to be scheduled based on the last calculated time interval in the back-off algorithm.

The following describes the flow of control for the reconnect algorithm and for the scheduler:

**Data Model Updates:**



**Scheduling:**



## Error Handling

*TBD*

### Error Messages

*TBD*

#### (Error Name)

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** |  | **Message Type** |  |
| **Error** |  | | |
| **Root Cause** |  | | |
| **Corrective Action** |  | | |

## Security

*TBD - Taken from a comment in a previous version of the spec: “Primary consideration is authentication on “pipe” to services and security of “pipe” (not able to sniff). Also need to consider encrypting when in memory.”*

## Logging

*TBD*

## Boundary Conditions

*TBD*

## Constraints

*TBD*

## Platforms

Partner Common Services will support running on the following host operating systems:

* Windows 2003
* Windows 2008

Partner Common Services will also support running in a Windows Hypervisor VM.

Partner Common Services will support running on the following servers:

* All servers supported by PSP 8.70

Partner Common Services will support management of target nodes running the following operating systems:

* ESX 4.x
* ESXi 4.x
* RHEL 5
* RHEL 6
* SLES 10
* SLES 11

Partner common Services will support management of the following target devices:

* All servers and options supported by PSP 8.70
* 100-series G7 ProLiants
* iLO2, iLO3
* OA
* Virtual Connect

## Internationalization

Partner Common Services does not include any internationalization support.

## Performance

*TBD*

## Portability

Although Partner Common Services currently only supports running on a Windows system, there is an expectation of a future requirement to run on Linux (as part of a Linux appliance). Therefore, consideration should be made such that the product can be easily ported to Linux in the future.

## Expandability

A 2H2011 release of Partner Common Services will need to support the Gen8 platform, particularly support for iLO4’s embedded management functionality (iRaptor).

Another focus of future releases will be enhancements to authentication and discovery. Possible enhancement include use of Kerberos and Windows ACLs for authentication, and automatic SNMP configuration of discovered target nodes.

## Support and Maintenance

*TBD – Consider any functionality and features aimed at improving serviceability and maintainability*

## Migration and Upgrade

*TBD – Cover any considerations that need to be made regarding migrations or upgrade*

# Event Descriptions

This section describes the structure and types of events that may be generated by Common Services

## Event Structure

The event structure is defined by the event schema. A basic description is presented here:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| eventID | unsignedLong | A monotonically increasing integer that uniquely identifies the event. |
| eventName | String | The formalized event name |
| eventTimeStamp | dateTime | The time when the event was created. |
| eventText | String | A free-form text string describing the event. |
| eventCategory | String (enumerated values) | The category of the event (instrinsic,Event, snmpTrap, wbemIndication) |
| eventType | String (enumerated values) | The type of event (create, update, delete, other) |
| eventSeverity | String (enumerated values) | The severity of the event (normal, minor, major, etc) |
| eventReferencedDatamodel | String | The short-name of the entity type that created this event (ilo, oa, vcdomain, etc) |
| eventReferencedManagedElementID | String (uuid) | The entity that created this event. |
| eventSourceIP | String | The IP address of the entity that created this event |
| keyValuePairs | List of keyValuePair items | Additional free-form key-value pairs associated with this event. |
| **keyvaluePair Field Name** | **Type** | **Description** |
| key | String | The key name of the kvpair. |
| value | String | The value of the kvpair. |
| altkey | String | An optional alternate key name. |

## Formalized Events

The following list describes each of the events Common Services can generate.

### Start of Day Event

Name: hpcs.event.StartOfDay

Description: A Start of Day Event sent each time Common Services start.

Required key-value pairs: None

Optional key-value pairs: None

### Test Event

Name: hpcs.event.Test

Description: A Test Event used during debugging

Required key-value pairs: None

Optional key-value pairs: None

### Entity Created

Name: hpcs.event.EntityCreated

Description: A managed entity was created

Required key-value pairs: None

Optional key-value pairs: None

### Entity Modified

Name: hpcs.event.EntityModified

Description: A managed entity was modified

Required key-value pairs: None

Optional key-value pairs:

path: The path inside the entity where data was modified.

alt\_uuid: A different entity that may have been affected by this modification.

### Entity Destroyed

Name: hpcs.event.EntityDestroyed

Description: A managed entity was destroyed

Required key-value pairs: None

Optional key-value pairs: None

### No Credentials

Name: hpcs.event.NoCredentials

Description: No credentials found for this entity

Required key-value pairs: None

Optional key-value pairs:

vcmDomainName: The virtual connect domain related to this event.

### Auth Error

Name: hpcs.event.AuthError

Description: Error authenticating to the managed node.

Required key-value pairs: None

Optional key-value pairs:

vcmDomainName: The virtual connect domain related to this event.

### Invalid Firmware

Name: hpcs.event.InvalidFirmware

Description: Invalid Firmware on the managed node.

Required key-value pairs:

currentFwVersion: The current firmware version on the device.

minimumFwVersion: The minimum firmware version required.

Optional key-value pairs:

vcmDomainName: The virtual connect domain related to this event.

### Failover

Name: hpcs.event.Failover

Description: Primary/Standby failover in progress

Required key-value pairs:

failoverIP: The address that should become the primary.

Optional key-value pairs:

vcmDomainName: The virtual connect domain related to this event.

### Connect Error

Name: hpcs.event.ConnectError

Description: Error connecting to the managed node.

Required key-value pairs: None

Optional key-value pairs:

vcmDomainName: The virtual connect domain related to this event.

### PrivilegeError

Name: hpcs.event.PrivilegeError

Description: The user ID used to log into the managed node does not have sufficient privilege.

Required key-value pairs: None

Optional key-value pairs: None

### C-Series Hardware Event

Name: hpcs.event.cseries.hardware

Description: A hardware event in a C-Series enclosure.

Required key-value pairs:

entity: Which enclosure entity (fan, power supply, blade, interconnect)

bay: Which entity bay (e.g. if entity is ‘fan’, then bay means fan bay)

action: What happened (insert, remove, etc)

Optional key-value pairs:

blades: A comma separated list of which blades are affected by the event.

### C-Series Status Event

Name: hpcs.event.cseries.status

Description: A status event in a C-Series enclosure.

Required key-value pairs:

entity: Which enclosure entity (fan, power supply, blade, interconnect)

bay: Which entity bay (e.g. if entity is ‘fan’, then bay means fan bay)

status: The operational status of the entity.

Optional key-value pairs:

blades: A comma separated list of which blades are affected by the event.

### SNMP Event

Name: hpcs.event.snmp

Description: A SNMP event received from a managed node known by Common Services, transformed into meEventType. If the trap name is known, it is appended to the event name (e.g. hpcs.event.snmp.cpqHoGenericTrap).

Required key-value pairs: None

Optional key-value pairs: The varbinds present in the SNMP event will be included as key-value pairs in the meEvent. The key will be varbind***n***, the altkey will be the OID of the varbind and the value will be the value of the varbind.

### WBEM Indication

Name: hpcs.event.wbem

Description: A WBEM indication received from a managed node known by Common Services, transformed into meEventType. If the indication name is known, it is appended to the event name (e.g. hpcs.event.wbem.fanFailed).

Required key-value pairs: None

Optional key-value pairs: The variables in the wbem indication payload will be added a key-value pairs. Normally, these include Summary, Description and RecommendedActions. The key-value pairs may be modified by the event meta-data (aka the event MasterXML file).

### CSERIES Indication

Name: hpcs.event.cseries

Description: An OA SOAP event received by Common Services, transformed into meEventType. If the event is of a status type, the name is hpcs.event.cseries.status. If the event is of a hardware type, the name is hpcs.event.cseries.hardware.

Required key-value pairs: None

Optional key-value pairs: The variables in the OA SOAP event payload are added as key-value pairs.