Intel®
Data Center
Manager

Intel(R) Datacenter Manager: Energy Director Developer's Guide

Document number: 319148-016US

Revision 3.8

Start Here

www.intel.com

Disclaimer and Legal Information

NOTE

Starting from now, the product name is changed to 'Intel(R) Datacenter Manager: Energy Director' ('Intel DCM: Energy Director'). The notation of 'Intel(R) Datacenter Manager: Energy Director 2.x/3.x' or 'Intel(R) DCM: Energy Director 2.x/3.x' in the document refers to old and new versions of the product.

Table of Contents

Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to:

http://www.intel.com/products/processor_number/

MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, MP3, DV, VC-1, MJPEG, AC3, AAC, G.711, G.722, G.722.1, G.722.2, AMRWB, Extended AMRWB (AMRWB+), G.167, G.168, G.169, G.723.1, G.726, G.728, G.729, G.729.1, GSM AMR, GSM FR are international standards promoted by ISO, IEC, ITU, ETSI, 3GPP and other

organizations. Implementations of these standards, or the standard enabled platforms may require licenses from various entities, including Intel Corporation.

Notational Conventions

This documentation uses the following conventions:

This type style	Indicates an element of syntax, reserved word, keyword, filename, computer output, or part of a program example. The text appears in lowercase unless uppercase is significant.
This type style	Indicates the exact characters you type as input.
This type style	Indicates a placeholder for an identifier, an expression, a string, a symbol, or a value. Substitute one of these items for the placeholder.
[*items*]	Indicates that the items enclosed in brackets are optional.
{ *item* | *item* }	Indicates to select only one of the items listed between braces. A vertical bar (|) separates the items.
... (ellipses)	Indicate that you can repeat the preceding item.

Getting Started

Getting Started

If you are using previous versions, you need to migrate to Intel(R) DCM: Energy Director 3.x before getting started:

- From Intel(R) DCM: Energy Director 3.0 to Intel(R) DCM: Energy Director 3.x, see Migrating from Version 3.0.
- From Intel(R) DCM: Energy Director 2.0 to Intel(R) DCM: Energy Director 3.0, see Migrating from Version 2.0.

The Intel(R) Datacenter Manager: Energy Director (Intel(R) DCM: Energy Director) enables you to monitor and manage power and thermals in a datacenter. You can integrate the Intel(R) Datacenter Manager: Energy Director API into your enterprise software management console.

The following table shows some of the main functions enabled by the Intel(R) DCM: Energy Director API:

Category	Key Functions	Comments
Configure datacenter hierarchy	• Add Node / Group • Associate Nodes to groups • Import and export datacenter hierarchy from and to a file	See Datacenter Modeling Interface
Configure data collection properties	Configure: • Collection state • Collection frequency	See Configuration Interface
Monitor power and thermals	• Monitor node and group level actual power consumption • Monitor Node and group level actual inlet temperature • Log power and thermal	See Query/Metrics Interface

	history trend data • Query trend data using filters • Retrieve raw data for post-processing	
Monitor and Configure Events	Configure and monitor both predefined and user-defined events	See Events Interface
Control	• Configure group level policies • Set policy scheduling • Set custom power capping • Force immediate reduction in power to handle emergencies • Set SLA priority as input • Supports multiple active policies of any type simultaneously • Co-existing policies at multiple hierarchy levels	See Control Policies Interface
General	• Get Version	See General Interface
Real-time Monitoring and Control	Monitor and control nodes and groups information	See Real-time Monitoring and Control Interface
Expert System	Provide knowledge from experts	See Expert System Interface

To get started with Intel(R) DCM: Energy Director, use an application to connect to the Intel(R) DCM: Energy Director web service. For more information, see About the Web Service.

Your Intel(R) DCM: Energy Director installation includes a reference user interface that connects to the web service. For a code sample from the reference user interface that shows how to use getQueryData to monitor power or temperature, see Code Sample: Getting Query Data.

See Also

About the Web Service

Code Sample: Getting Query Data

Reference User Interface

Solution Architecture

Nodes and Groups

API Reference: Overview

Overview

Intel(R) DCM: Energy Director enables you to monitor and manage power consumption and thermals in your datacenter.

Intel(R) DCM: Energy Director: Energy Director manages different platforms through the firmware agents on the platforms, for example, Intel(R) Intelligent Power Node Manager which monitors power consumption, limits power consumption and monitors inlet temperature on the server.

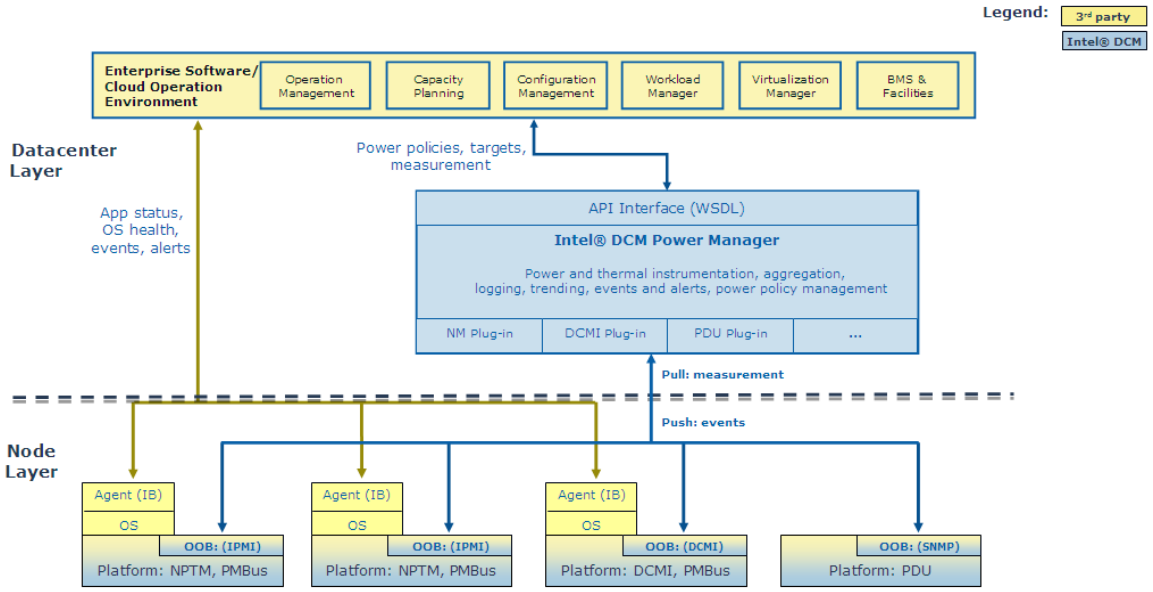
Intel(R) DCM communicates with the firmware agents using different protocols, for example, Intelligent Platform Management Interface (IPMI), Simple Network Management Protocol (SNMP), and Secure Shell (SSH) over LAN. Intel(R) DCM: Energy Director uses out of band management, and therefore has no dependency on the operating system installed on the managed node.

You need to integrate Intel(R) DCM: Energy Director into existing management consoles through the web service interface. Intel(R) DCM: Energy Director supports Transport Layer Security (TLS) for communication with management consoles.

NOTE

The absolute time on the client running the user interface must be synchronized with the absolute time on the Intel(R) DCM: Energy Director server.

The following Intel(R) DCM: Energy Director architecture image shows how it interacts with other parts of the datacenter:



BMS: Building Management System

IB: In Band

OOB: Out of Band

Data Storage

Intel(R) DCM: Energy Director uses a PostgreSQL out-of-process database.

To store passwords in the internal database, Intel(R) DCM: Energy Director uses AES-128 password encryption. When the Intel(R) DCM: Energy Director API receives a password, it encrypts the password for storage. When a communication module uses a password, it decrypts the password immediately before use.

The database stores the following data:

- Datacenter hierarchy structure
- Node and group configuration properties
- Node measurement data
- Intel(R) DCM: Energy Director action log
- Policy configuration data
- Event configuration data

Database Maintenance

Once each day, the database performs maintenance on the data stored.

The DB_MAINTENANCE_HOUR variable sets the time of day at which this maintenance occurs. Here is the database maintenance process:

- One hour before the maintenance starts, Intel(R) DCM: Energy Director sends the predefined event DB_MAINTENANCE_STARTING.
- The database deletes old action log records, as configured by the global property ACTION_LOG_BACK_TRACKING_PERIOD.
- The database deletes old measurement records, as configured by the global property TIME_UNTIL_DB_DELETION.
- The database compresses old measurement records, as configured by the global property TIME_UNTIL_DB_COMPRESSION.

Database compression aggregates all power and thermal measurements for a node over the course of each hour into a single measurement record for that hour. For measurements not related to power and temperature, database compression chooses only the most recent value.

See Also

GlobalProperty

Logging

Nodes and Groups

This topic defines the several terms used throughout this document:

Node. In Intel(R) DCM: Energy Director, the term node refers to a single server or a PDU device. Nodes can be included in groups.

Entity. The term entity refers to either a node or a group. An entity can be included in any number of logical groups. However, an entity can be directly included in only one physical group at one time. For example, to include a node in both a rack and a row, associate the node with the rack and then associate the rack with the row.

Managed Entity. An entity exposing its manageability through certain interface or protocol and managed by Intel(R) DCM: Energy Director through its management interface, for example, a node enabled with Intel(R) Intelligent Power Node Manager.

Group. Each group is either a physical group or a logical group:

- A physical group includes the nodes defined by physical properties. Types of physical groups are:

    - Enclosure: A physical group which models the enclosure devices containing multiple blade servers.
    - Rack: A physical group that includes all nodes in a physical rack in the physical datacenter.
    - Row: A physical group that includes all racks in a physical row in the physical datacenter.
    - Room: A physical group that includes all rows in a physical room in the datacenter.
    - Datacenter
- A logical group is a grouping of servers based on user-defined attributes. For example, you can create a group of all nodes running a web server or all nodes running a specific operating system.

Managing Thousands of Nodes

Intel(R) DCM: Energy Director supports up to 5000 managed devices in one data center through a stand-alone server or a single tier-two server.

Intel(R) DCM: Energy Director supports up to total 10000 managed nodes in two-tier architecture.

If you manage more than 3000 devices, you need to change the following Windows* OS default settings:

- **Maximum dynamic port:** Intel(R) DCM: Energy Director uses a dynamic UDP port to communicate with nodes. By default, up to 3976 ports (port number 1025-5000) can be used as dynamic ports. You need to change the number of maximum dynamic ports so that it is larger than the number of managed devices.
- **TCP/IP wait time:** If a large number of events (-5000) are sent from tier-two server to the central server in a short time, it may keep sockets in "time_wait" status. You need to configure a long enough wait time setting.

To configure these settings, see the following example for Windows* 2003 Server:

1. Open the Windows* OS registry file.
2. Go to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`

3. Modify the value MaxUserPort:
    a. If value MaxUserPort exists, go to step3.c
    b. If value MaxUserPort does not exist, right click mouse, select **New > DWORD Value**, and enter the name **MaxUserPort**.
    c. Select **MaxUserPort** and right click mouse, select **Modify** to edit the value as follows:
    - Value Name: MaxUserPort
    - Value Data: 65534 (Base: decimal)
4. Modify the value TCPTimedWaitDelay:
    a. If value TCPTimedWaitDelay exists, go to step4.c
    b. If value TCPTimedWaitDelay does not exist, right click and select **New > DWORD Value**, enter the name **TCPTimedWaitDelay**.
    c. Right click **TCPTimedWaitDelay** and select **Modify** to edit the value as follows:
    - Value Name: TCPTimedWaitDelay
    - Value Data: 30 (Base: decimal)
5. Click **OK** to save.

On Linux* OS, a small ARP cache Garbage Collector (GC) threshold is used by default. If the node number is bigger than the value of `gc_thresh1`, you need to modify this value.

You can retrieve the value of `default gc_thresh1` by entering the following command:

```
#sysctl -a | grep net.ipv4.neigh.default.gc_thresh1
```

`gc_thresh*` can be permanently modified by adding the following lines in `/etc/sysctl.conf`:

```
net.ipv4.neigh.default.gc_thresh1="value1"
net.ipv4.neigh.default.gc_thresh2="value2"
net.ipv4.neigh.default.gc_thresh3="value3"
```

NOTE

Suggested value: value1 >= maximum node number, value2 = 4*value1, value3 = 2*value2

If you manage more than 1000 devices on Linux* OS, the following configurations should be performed:

1. Add the following line to /etc/security/limits.conf (replace ***value*** with a number which is greater than the number of nodes):

```
* - nofile value
```

2. For 32-bit Linux OS, add the following line to /etc/pam.d/login

```
session required /lib/security/pam_limits.so
```

For 64-bit Linux OS, add the following line to /etc/pam.d/login

```
session required /lib64/security/pam_limits.so
```

The OS should be rebooted to make the changes take effect.

It is also recommended to adjust shared_buffers and checkpoint_segments configuration of PostgreSQL for better performance of data repository. By default, the configuration could be found in <dcm_directory>/pgdata/postgresql.conf and the recommended settings are listed below:

```
shared_buffers = 512MB
checkpoint_segments = 32
```

For some Linux configurations or editions (e.g., RHEL 6 Basic Server), the default value of the kernel parameter SHMMAX might be too small to run PostgreSQL with the configuration above. You need to modify the kernel parameter SHMMAX accordingly (cf., PostgreSQL manuals for details).

See Also

Setting the Datacenter Hierarchy

Datacenter Modeling Interface: Overview

Logging

Intel(R) DCM: Energy Director uses two types of logs:

- Application log
- Action log

Application Log

The application log tracks notifications on occurrences in Intel(R) DCM: Energy Director. You can choose from three levels of application logs:

- **Info**. Provides notifications on errors and other application occurrences. This is the default level.
- **Debug**. Includes all notifications included in Info, and adds notifications detailing the start and end of API calls and system-initiated sequences. For example, this level of notification includes a notification that the method addEntity has started and ended, and the results of the operation.
- **Trace**. Includes all notifications included in Debug, and adds notifications detailing the start and end of each action in Intel(R) DCM. For example, this level of notification includes a notification on each action taken in the process of adding an entity.

To configure the application log level, edit the global property APP_LOG_LEVEL.

Action Log

The action log tracks:

- every action taken by the user through the API
- all custom events
- all predefined events
- activation of an event or policy

The action log is stored in the internal database.

See Also

GlobalProperty

getActionLogRecords

Security

Security: Overview

Intel(R) DCM: Energy Director supports the following security options:

- Communication
- Data Storage

Communication

- **TLS protected Web service API.**

  You can enable TLS as part of the installation. TLS enables:

  - API calling from Enterprise Console
  - Calling between different components. See Web Service Security and Connecting Central Server with Tier-two Server with TLS Enabled.

- **Event integrity verification.** Intel(R) DCM: Energy Director uses digital signature to verify the integrity of event notification, including event notification to management console and event notification between different components. See Event Signature.

- **Communication with nodes.**

  - **IPMI/Node Manager/DCMI nodes.** Intel(R) DCM: Energy Director supports Intel IPMI Cipher Suites, ID 0-3 to communicate with nodes. Intel(R) DCM: Energy Director uses a BMC node to communicate with nodes. This BMC node must have the ADMIN privilege level and it must be configured to enable the ADMIN role to use at least one of the cipher suite levels 0-3. Intel(R) DCM: Energy Director uses the lowest enabled Cipher Suite level.
  - **PDU nodes.** Intel(R) DCM: Energy Director supports SNMP v3 for communication with PDU nodes. The node must be configured to enable the Simple Network Management Protocol (SNMP) v3 User-based Security Model (USM).

Data Storage

- **AES-128 password encryption in the internal database.** When the Intel(R) DCM: Energy Director API receives a password, it encrypts the password for storage. When a communication module uses a password, it decrypts the password immediately before use.

- **User Configure File.** Intel(R) DCM: Energy Director uses OS user access control to protect the confidentiality of information in user configure file.
- **Key-Store File.** Intel(R) DCM: Energy Director uses a Java Key-store (JKS) file for the TLS RSA keys. This file is located under the Intel(R) DCM: Energy Director installation directory. See Key-store File.
- **XML File Security.** An encrypted key encrypts communication between the client and Intel(R) DCM: Energy Director. The encrypted key is added to the XML file. For more information, see the XML schema in Importing or Exporting Hierarchy Files, or see the Hierarchy File Example.

Web Service Security

About Web Service Security

Intel(R) DCM: Energy Director is designed to work with TLS (Transport Layer Security) in case the confidentiality and data integrity need to be enforced for web service calls. There are several authentication options available when TLS is enabled. Generally, Intel(R) DCM: Energy Director does not invent or implement its own security mechanism, instead, it uses existing security options in Tomcat and other third party components.

See Also

Single Box Solution

TLS with Authentication Options

Certificate Management

Single Box Solution

If you use the single box security option, Intel(R) DCM: Energy Director server and the management console are installed on the same server. In this case the management console only calls the Intel(R) DCM: Energy Director API through local web service. Therefore you do not need TLS because the web service calls do not go through the real network. In this case, you can do the following:

- Change the configuration of Tomcat Server service to accept the connection from local address only.

- Change the configuration of event console to accept events sent from local address only.

See Also

TLS with Authentication Options

Certificate Management

TLS with Authentication Options

Intel(R) DCM: Energy Director supports TLS with various authentication options.

You can enable TLS to enforce the confidentiality and data integrity of web service calls when Intel(R) DCM server and management console are deployed in the same network environment. The following authentication options are available when TLS is enabled:

Option 1. Http basic authentication for client (management console), certificate based TLS authentication for server (Intel(R) DCM: Energy Director server).

Use this option to provide basic authentication for the Intel(R) DCM: Energy Director client(s) and certificate based TLS authentication for the Intel(R) DCM: Energy Director server. This is the default option.

Refer to http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html.

Option 2. Certificate based TLS mutual authentication for client and server.

Use this option for better security. You can enable TLS connection based authentication by modifying the configuration of Tomcat server that installed by Intel(R) DCM: Energy Director.

Refer to http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html.

Option 3. Kerberos authentication for client, certificate based TLS authentication for server.

For more information on how to enable Kerberos authentication in Tomcat through SPNGEO filter, refer to http://spnego.sourceforge.net/spnego_tomcat.html.

NOTE

For better security, certificate based mutual authentication is recommended.

See Also

Single Box Solution

Certificate Management

Certificate Management
Intel(R) DCM: Energy Director uses a key-store file to manage the certificates, including server certificates, trusted CA certificate chains and trusted client certificates.

Key-store File

This key-store file is a Java Key Store <JKS> format and is saved as *keystore.ssl* under the Intel(R) DCM: Energy Director installation directory. Its password is saved in Intel(R) DCM user configuration file. You can manage the key-store file using keytool provided by Sun* JDK.

For more information on the keytool, refer to http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html.

See Also

Single Box Solution

TLS with Authentication Options

Connecting Central server with Tier-two server communication with TLS enabled

Intel(R) DCM: Energy Director supports two-tier architecture, that is, one central server can manage multiple tier-two servers. This enables you to develop your own solution for managing many nodes. The communication between the central server and tier-two servers can be protected by TLS. In this case, the central server calls the tier-two servers over *https* (instead of *http*). Two tier architecture secure communication must use certificate-based TLS mutual authentication between the central server and the tier-two servers. That is, http basic authentication or Kerberos authentication are not supported.

To enable TLS between central server and tier-two server, perform the following steps before adding the tier-two servers to the central server:

1. Enable TLS options in Tomcat server of the central server and tier-two server. Refer to http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html.

2. If the tier-two server uses CA signed certificate, import the CA certificate into central server's key-store file as a trusted CA certificate.
3. If the tier-two server does not use CA signed certificate, do the following:
   - Export the certificate of a tier-two server from its key-store file
   - Import the key-store file to the central server's key-store file as a trusted certificate
   - Export the certificate of the central server from its key-store file
   - Import the key-store file to the tier-two server key-store file as a trusted certificate

For large-scale certificate deployment case, CA based certificate exchange is recommended.

See Also

Certificate Management

Using Event Signature

About Event Signature

Intel(R) DCM: Energy Director server uses a HMAC-SHA-256 algorithm to sign the events notification it sends out. The management console can use the event signature for event message authentication and integrity check If the management console does not require event source identification and message integrity, the management console can skip the signature tag in the event message.

See Also

getEventMessageAuthenticationKey

Event Message Authentication Key

Intel(R) DCM: Energy Director uses a security key for event message authentication. The security key is a base64 encoding binary key used for the HMAC-SHA-256 algorithm for signing event notification.

The process for securing messages includes the following stages:

1. During the subscribeEventHandler stage, Intel(R) DCM: Energy Director generates an event message authentication key (256 bits) and signs the event message.
2. Intel(R) DCM: Energy Director subscribes the event handler.
3. The management console gets the security key with the following API:

```
String getEventMessageAuthenticationKey (int handlerId)
```

The returned string is encoded in base64 encoding. The management console uses a base64 decoder to restore the key to binary mode.

For example, if the returned security key is:

```
"rqYbukgOe5hXrndqR9FrZt3AiSZ1p/sgiMw40p4xS74="
```

After base64 decoding, the 32 bytes raw binary key is:

```
[0xae, 0xa6, 0x1b, 0xba, 0x48, 0x0e, 0x7b, 0x98, 0x57, 0xae, 0x77, 0x6a,
0x47, 0xd1, 0x6b, 0x66, 0xdd, 0xc0, 0x89, 0x26, 0x75, 0xa7, 0xfb, 0x20,
0x88, 0xcc, 0x38, 0xd2, 0x9e, 0x31, 0x4b, 0xbe]
```

NOTE

The authentication keys are stored in database (encrypted) which is connected to Intel(R) DCM: Energy Director server. Therefore, the database security is important. You shall make sure the database is safe when developing your own solution.

See Also

getEventMessageAuthenticationKey

Key Expiration and Update

If cryptographic keys are used, a key replacement policy is required. In Intel(R) DCM: Energy Director, an event authentication key is generated every time an event is subscribed. If the management console needs to update event authentication key before key expiration, it needs to unsubscribe and re-subscribe that event first, management console will query the new key upon each subscription.

See Also

subscribeEventHandler

Message Integrity Verification

Intel(R) DCM: Energy Director uses digital signature to verify the integrity of event notification. The following example shows how to verify it.

This is the original event message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:dcm="http://wsdl.intf.dcm.intel.com/events">
<s12:Header>
<wsa:Action>http://wsdl.intf.dcm.intel.com/event</wsa:Action>
<wsa:MessageID>uuid:dbdcb11e-a997-4a86-bcf9-
1e1e501e9253</wsa:MessageID>
</s12:Header>
<s12:Body>
<dcm:Event>
<dcm:Url>tcp://yding9-
MOBL1.ccr.corp.intel.com:8686/42/WsEventListener/</dcm:Url>
<dcm:PredefinedEventType>CONFIGURATION_CHANGED</dcm:PredefinedEventType
>
<dcm:PolicyId>-1</dcm:PolicyId>
<dcm:EntityID>-1</dcm:EntityID>
<dcm:EventTime>2010-01-18T6:21:49Z</dcm:EventTime>
<dcm:SeverityLevel>INFORMATIVE</dcm:SeverityLevel>
<dcm:Info>The value of property: NODE_POWER_MEASUREMENT_GRANULARITY has
changed from: 60 to: 180</dcm:Info>
<dcm:Data></dcm:Data>
</dcm:Event>
<dcm:Signature>Lldyn1HJ0Rkn/9oNKgDAanWB6FNhA3KyUfrJSwOCOo4=</dcm:Signat
ure>
</s12:Body>
</s12:Envelope>
```

To verify message integrity, the event receiver removes the signature file from the original message as follows, and then generates the signature from it.

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:dcm="http://wsdl.intf.dcm.intel.com/events">
<s12:Header>
<wsa:Action>http://wsdl.intf.dcm.intel.com/event</wsa:Action>
<wsa:MessageID>uuid:dbdcb11e-a997-4a86-bcf9-
1e1e501e9253</wsa:MessageID>
</s12:Header>
<s12:Body>
<dcm:Event>
<dcm:Url>tcp://yding9-
MOBL1.ccr.corp.intel.com:8686/42/WsEventListener/</dcm:Url>
<dcm:PredefinedEventType>CONFIGURATION_CHANGED</dcm:PredefinedEventType
>
<dcm:PolicyId>-1</dcm:PolicyId>
<dcm:EntityID>-1</dcm:EntityID>
<dcm:EventTime>2010-01-18T6:21:49Z</dcm:EventTime>
```

```
<dcm:SeverityLevel>INFORMATIVE</dcm:SeverityLevel>
<dcm:Info>The value of property: NODE_POWER_MEASUREMENT_GRANULARITY has
changed from: 60 to: 180</dcm:Info>
<dcm:Data></dcm:Data>
</dcm:Event>
<dcm:Signature></dcm:Signature>
</s12:Body>
</s12:Envelope>
```

Now you can verify the original signature by comparing with newly created signature.

Here is the code sample to verify the signature:

```
public boolean verifySignature(String message, String signature, byte[]
key)
      throws DcmSecurityException{
            SecretKeySpec signingKey = new SecretKeySpec(key,
"HmacSHA256");
            Mac mac;

            try {
                mac = Mac.getInstance("HmacSHA256");
                mac.init(signingKey);
            }

            catch(NoSuchAlgorithmException e) {
                e.printStackTrace();
            }

            catch(InvalidKeyException e) {
                  e.printStackTrace();
            }

            byte[] rawHmac = mac.doFinal(message.getBytes());

            BASE64Encoder encoder = new BASE64Encoder();

            String strSignature = encoder.encode(rawHmac);

            return strSignature.equals(signature);
```

See Also

getEventMessageAuthenticationKey

Protecting User Configuration File

User Configuration File Security

Intel(R) DCM: Energy Director saves the following sensitive information in the configuration file on the server:

- Encrypted database password
- Encrypted key-store password

- Database password encryption key
- Key for protecting other sensitive information on Intel(R) DCM: Energy Director server

You must enforce the integrity and confidentiality of the user configuration file to protect:

- The integrity of user configuration file in the implementation
- The confidentiality of user configuration file through user access control in the OS

Using InstallTool

Intel(R) DCM: Energy Director provides an external tool *InstallTool.jar* to manage the user configuration file.

Enter the following command to use InstallTool on Windows* OS:

```
java -cp DCM.jar;InstallTool.jar[;postgresql-8.3-603.jdbc4.jar]
com.intel.dcm.install.Security [-create –option| -read -option | -write
-option=value| -help] –file <configuration file>
```

Enter the following command to use InstallTool on Linux* OS:

```
java -cp DCM.jar:InstallTool.jar[:postgresql-8.3-603.jdbc4.jar]
com.intel.dcm.install.Security [-create –option| -read -option | -write
-option=value| -help] –file <configuration file>
```

InstallTool.jar is located in `INSTALL_DIR\bin`

DCM.jar is located in `INSTALL_DIR\bin\plugins\`

postgresql-8.3-603.jdbc4.jar is located in `INSTALL_DIR\tool` (only exists in the zip installation, the version may change for each Intel(R) DCM release).

Option	Parameter	Description
-Create	[dbpasswdkey]	Create a database password key and write into configuration file.
	[rootkey]	Create a root key and write into configuration file.
-Read	[dbpasswd]	Return the database password from the configuration file (decrypted with database password

		key). If database password or database password key does not exist, return *failure*.
	[keystorepasswd]	Return key store password from the configuration file (decrypted with root key). If key store password or root key does not exist, return *failure*.
-Write	[dbpasswd <DB password >]	Write database password into configuration file (encrypted with database password key). If database password key does not exist, return *failure*.
	[keystorepasswd <key store password >]	Write key store password into configuration file (encrypted with root key). If root key does not exist, return *failure*.
-Update	[dbpasswd <DB password>]	Update database password into configuration file (encrypted with database password key). If database password key does not exist, return *failure*.
	[keystorepasswd <key store password>]	Update key store password into configuration file (encrypted with *Root* key). If root key does not exist, return *failure*.
	[rootkey]	Update root key in the configuration file, as well as re-encrypt all encrypted data in database and configuration file. You must stop all Intel(R) DCM services before calling this command. Updating root key

		requires access to database, therefore, you shall include *postgresql-8.3-603.jdbc4.jar* in the command line.
	[dbpasswdkey]	Update database password key in the configuration file, and re-encrypt database password key.

The following steps show how to use InstallTool in Windows*:

1. Create a root key using the following command line:

   ```
   java -cp DCM.jar;InstallTool.jar com.intel.dcm.install.Security –create rootkey –file <user.config.xml file path>
   ```

2. Create a database password key using the following command line:

   ```
   java -cp DCM.jar;InstallTool.jar com.intel.dcm.install.Security –create dbpasswdkey –file <user.config.xml file path>
   ```

3. When the database password is entered, write it into configuration file using the following command line:

   ```
   java -cp DCM.jar;InstallTool.jar com.intel.dcm.install.Security –write dbpasswd “db password” –file <user.config.xml file path>
   ```

4. If TLS is enabled, when a user enters the key store password, write it into the configuration file using the following command line:

   ```
   java -cp DCM.jar;InstallTool.jar com.intel.dcm.install.Security –write keystorepasswd “keystrore password” –file <user.config.xml file path>
   ```

You can skip step 4 if TLS is not enabled. Refer to the *Intel(R) DCM: Energy Director Installation Guide* for more information.

Web Service

About the Web Service

You interact with Intel(R) DCM: Energy Director through a web service interface. The web service interface uses the JAX-WS web service engine installed on a Tomcat* 6 application server. To communicate with client systems, Intel(R) DCM: Energy Director supports TLS. You can enable using TLS as part of the Intel(R) DCM: Energy Director installation. Refer to the *Intel(R) Datacenter Manager: Energy Director Installation Guide* for more information.

The Intel(R) DCM: Energy Director web service description language (WSDL) connector class contains all Intel(R) DCM: Energy Director API methods, and passes them to the API implementation.

NOTE

To the web service, a null value and an empty array are equivalent.

The following is the default URL for the WSDL schema, without enabling TLS:

http://<server_name>:8688/DCMWsdl/dcm.wsdl

The default URL for the WSDL schema, using TLS:

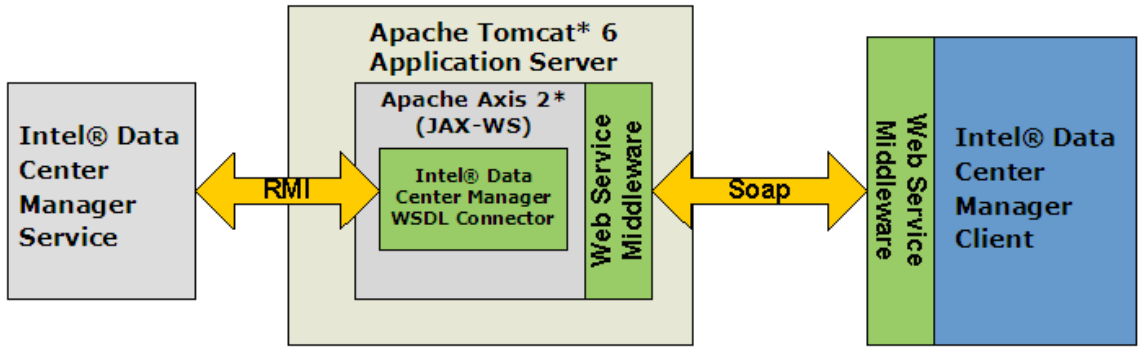
https://<server_name>:8643/DCMWsdl/dcm.wsdl

The *<server_name>* is the name of the server on which you installed Intel(R) DCM: Energy Director. If the client is running on the same machine as is Intel(R) DCM: Energy Director, you can use *localhost* for the *<server_name>*.

NOTE

The absolute time on the client running the user interface must be synchronized with the absolute time on the Intel(R) DCM: Energy Director server.

The web service interface is deployed as follows:



See Also

Code Sample: Getting Query Data

Reference User Interface: Overview

Invoke the Web Service

Intel(R) DCM: Energy Director provides web service interface based on JAX-WS that includes related web services utilities that are already configured to use the required libraries in the class path.

To create the Intel(R) DCM: Energy Director client in Java, do the following:

1. Use the wsimport utility to generate stub classes source. Use the following command line format:

   ```
   <jdk Directory>\bin\wsimport.exe –s <sourcedir> <WSDL URL>
   ```

   For example:

   ```
   C:\Program Files\Java\jdk1.6.0_13\bin>wsimport.exe –s source
   http://localhost:8688/DCMWsdl/dcm.wsdl
   ```

   NOTE

   The generated stub class source code is located in the source directory.

2. Create your client application to invoke the web service interface:

   1) Get the endpoint stub class by initializing a stub service instance.

   2) Invoke the web service interface like a local method.

Code sample to invoke getVersion interface.

```
import java.net.MalformedURLException;
import java.net.URL;

import javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.namespace.QName;

public class SampleClient {

    private static QName qName = new
QName("http://wsdl.intf.dcm.intel.com/", "Dcm");
    private static URL url = null;
    /**
     * @param args
     */
    public static void main(String[] args) {
       // TODO Auto-generated method stub
       try{
```

```
            url = new URL("http://localhost:8688/DCMWsdl/dcm.wsdl");
            Dcm_Service dcmService = new Dcm_Service(url, qName);
            Dcm dcmWsdl = dcmService.getDcmPort();
            str = dcmWsdl.getVersion();
        }catch (Exception_Exception e)
        {
            //handle web service invocation exception
        }
        catch (MalformedURLException e1)
        {
            //handle new URL generated exception
        }
        finally{
            System.out.print(str);
        }
    }
}
```

Constructing bindings to web service interfaces is language specific. Though Intel(R) DCM: Energy Director tries to maintain the backward compatibility of web service interfaces with the previous versions, it is possible that some newly added features involve new members in objects which may not be recognized by old web service clients. In these cases, you need to reconstruct the bindings to the web service interfaces (For example, run step 1 again in the sample above for Java). It is common that script languages support dynamic binding of web services, which avoids the binding reconstruction in these cases.

See Also

Code Sample: Getting Query Data

Reference User Interface: Overview

Faults

Intel(R) DCM returns every exception as part of an HTTP status code *Internal Error 500*. The exception string includes the exception name and exception message, as follows:

exception_name: exception message...

See Also

Faults

Sample User Interface Implementation

Sample User Interface: Overview

The Intel(R) DCM: Energy Director reference user interface (reference UI) uses the Intel(R) DCM: Energy Director web service. The reference UI is written in ActionScript* language.

Prerequisites

To enable the reference UI to work, you need to do the following:

- Install the reference UI during the Intel(R) DCM: Energy Director installation
- Install the Adobe Flash Player*
- Synchronize the absolute time on the client running the user interface with the absolute time on the Intel(R) Data Center Manager server.

If you installed the reference UI as part of the Intel(R) DCM: Energy Director installation, you can view it by entering the following address in your web browser: http://<server_name>:8688/DataCenterManager/

By default, <*server_name*> is the name of your computer.

The reference UI does not encrypt communications with the Intel(R) Data Center Manager server.

Samples

As part of the Intel(R) DCM installation, you can install all source code samples for the Intel(R) DCM: Energy Director reference UI. By default, the samples are installed at:

- On Windows* OS: `c:\Program Files\Intel\DataCenterManager\samples`
- On Linux* OS: `/opt/intel/datacentermanager/samples`

In Intel(R) DCM: Energy Director reference UI, the term Managed Entity refers to either a managed node or a managed group (for example, an enclosure).

Following are five different use cases for the Intel(R) DCM: Energy Director UI:

- Adding a managed entity
- Adding a group
- Associating a managed entity with a group

- Monitoring power and inlet temperature
- Monitoring key metrics

Use Case 1: Adding a Managed Entity

1. Go to **Configuration** > **Add Managed Entity**
2. The **Managed Entity Properties** dialog box opens. In this box:
   - Enter a meaningful name to designate the managed entity name.
   - Enter the Baseboard Management Controller (BMC) IP address. For example, *12.34.57.12*. Ensure that the BMC does not have any administrator name or password set.
   - Enter a numerical value for De-rated Power in Watts. For example, *600*. De-rated Power is typically a percentage off the nameplate value of the server power supply.
3. Click **Save** to add the managed entity.

Use Case 2: Adding a Group

1. Go to **Configuration** > **Add Group**
2. The **Group Properties** dialog box opens. In this box:
   - From the **Group Type** drop down list, select the appropriate group type.
   - Enter a meaningful name to designate the group name.
3. Click **Save** to add the group.

Use Case 3: Associating a Managed Entity with a Group

1. Go to **Configuration** > **Find Managed Entity/Group**.
2. The **Find Managed Entity/Group** dialog box opens. In this box:
   - From the list of entities, select the managed entity you want to associate with a group.
   - Click **Associate**. A window opens called **Associate-<your_managed entity>**.
   - From the list of groups at the top of the **Associate-<your_managed entity>** window, select the group to which you want to associate the managed entity. You can select multiple groups.
   - Click **Associate**. The group moves to the lower list, meaning that the managed entity has been associated with that group.

Use Case 4: Monitoring Power and Inlet Temperature

1. To start monitoring, click the icon to the right of the Help menu. Roll over this icon, if it is currently monitoring it says **Resume Monitoring**. If the icon says **Suspend Monitoring**, monitoring is already on.
2. Go to **Configuration** > **Trending**. The **Trending Options** dialog opens.
3. In the **Trending Data** list, check the data that you want to see in the **Trending** graph.
4. Click **Save**.
5. To see the monitored data from a managed entity or group, click that managed entity or group on the left side under **Datacenter Hierarchy**.

Use Case 5: Monitoring Key Metrics

1. To see the monitored data from a managed entity or group, click that managed entity or group on the left side under **Datacenter Hierarchy**.
2. Under **Metrics** on the right side, choose the metrics you want to see in the time frame in the **Trending** graph.

NOTE

Only maximum, minimum and average for power and inlet temperature are implemented.

See Also

Setting the Datacenter Hierarchy

GlobalProperty

Code Sample: Getting Query Data

Query/Metrics Interface: Overview

Sample User Interface Code Samples

Code Sample: Overview

This section discusses the code samples, which you can install on your system as part of the Intel(R) DCM: Energy Director installation:

- Code Sample: Getting Query Data
- Code Sample: Adding a Managed Entity
- Code Sample: Associating a Managed Entity with a Group
- Code Sample: Configuring Monitoring

- Code Sample: Initializing the Web Service
- Code Sample: Subscribing Predefined Events
- Code Sample: Getting Predefined Events
- Code Sample: HTTP Event Handler
- Code Sample: Creating a Control Policy
- Code Sample: Scheduling a Policy
- Code Sample: Adding a Custom Event
- Code Sample: Updating a Custom Event
- Code Sample: Importing Hierarchy from File
- Code Sample: Exporting Hierarchy to File

These code samples are written in the ActionScript* language. In this document, we show only the most basic and most critical parts of the code.

During Intel(R) DCM installation, you can choose to install all sample source code for the Intel(R) DCM: Energy Director reference UI. By default, these samples are installed in:

- On Windows* OS: c:\Program Files\Intel\DataCenterManager\samples
- On Linux* OS: /opt/intel/datacentermanager/samples

See Also

About the Web Service

Reference User Interface

Code Sample: Initializing the Web Service

This code sample shows how to initialize the web service. This code sample comes from the file *DcmWebService.as*.

The following code:

1. creates the web service object
2. adds event listeners for load, faults and successful web service results:

```
service = new WebService();

// Add event listener for the successful loading of the WSDL scheme
service.addEventListener(LoadEvent.LOAD, loadHandler);
```

```
// Add event listener for faults.
service.addEventListener(FaultEvent.FAULT, requestResultHandler);
// Add event listener for successful results.
service.addEventListener(ResultEvent.RESULT, requestResultHandler);
```

Here is the result handler method for faults and for successful web service results:

```
private function requestResultHandler(event:AbstractEvent):void {
        var resType:String =
requestsDictionary[event.token.message.messageId] as String;
        // If the token was filed, then it is a response to a method.
        if (resType) {
               delete  requestsDictionary[event.token.message.messageId];
                var e:DcmWsReturnValueEvent = new
DcmWsReturnValueEvent(resType, event);
                dispatchEvent(e);
        } else {
                dispatchEvent(event);
        }
}
```

The following code shows a method that calls the web service. It gets a response from the web service and tries to match it to the response. The code adds all requests to the requestsDictionary container. This container maps between the token ID and the type of DcmWsReturnValueEvent to dispatch. Once the UI receives the response, it dispatches the appropriate DcmWsReturnValueEvent.

```
private function fileToken(token:AsyncToken,
eventType:String):AsyncToken {
        requestsDictionary[token.message.messageId] = eventType;
        return token;
}

public function getVersion():AsyncToken
{
        var token:AsyncToken = service.getVersion();
        return fileToken(token,
DcmWsReturnValueEvent.GET_DCM_VERSION_RESPONSE);
}
```

Code Sample: Getting Query Data

This code sample shows how to monitor power and temperature through the web service. This sample uses the method getQueryData through the web service.

This code sample comes from the file *QueryType.as*.

The following section of code defines an internal method that wraps the call to the web service method.

All arguments except QueryType are native to the language. QueryType is a class that holds the string value of all possible query types.

The getQueryData method provides data on the selected entity (managed entity or group) and displays the latest measurements. It always uses the *SELF* type as AggregationLevel.

From *DcmWebService.as*:

```
public function getQueryData(entityId:int, queryType:QueryType,
aggregationLevel:String,
           startTime:Date, endTime:Date, aggPeriod:int):AsyncToken
```

The following section of code adds the listener for the event representing the response from getQueryData. From *DcmTrendingManager.as*:

```
DcmUICore.instance.dcmWS.addEventListener(
  DcmWsReturnValueEvent.GET_QUERY_DATA_RESPONSE,
getQueryDataEventHandler);
```

The following section of code calls the wrapper method. From *DcmTrendingManager.as*:

```
var tok:AsyncToken = DcmUICore.instance.dcmWS.getQueryData(
        _monitoredEntity.id,
        queryType,
        aggLevel,
        startDate,
        endDate,
        aggPeriodSec);

++requestsLength;
requestsDictionary[tok.message.messageId] = queryType;
}
```

The following section of code handles the result of this query. From *DcmTrendingManager.as*:

```
var queryType:QueryType = requestsDictionary[event.messageId];
if (queryType == null) {
        return;
}
```

```
if (!event.isError) {
	// Handle single result and multiple results the same,
	// put the single result in an array.
	var arr:ArrayCollection;
	if (event.result.queryData is ArrayCollection) {
		arr = event.result.queryData as ArrayCollection;
	} else {
		arr = new ArrayCollection([event.result.queryData]);
	}
	// For each of the results, add it's data to the
tempTrendingPoints
	// array, to the right cell and the right trending data field
	for each (var obj:Object in arr) {
		if (obj != null && obj.value is int && obj.time is Date)
{
			queryType.addTrendingPoint(obj);
		}
	}
	if (event.result.enumerationHandle != 0) {
		var tok:AsyncToken =

	DcmUICore.instance.dcmWS.getNextData(event.result.enumerationHan
dle);
		++requestsLength;
		requestsDictionary[tok.message.messageId] = queryType;
	}
} else {
	Dumper.error(event.toString());
}
delete requestsDictionary[event.messageId];
++resultsCount;
DcmUI.instance.callLater(DcmUI.instance.setTrendingProgress,
		[100 * resultsCount / requestsLength]);
// If we got all the responses, finish the trending cycle.
if (resultsCount == requestsLength) {
	finishTrendingCycle();
}
```

This example uses the enumerationHandle field to check if there is more data waiting for fetching. If this field is not 0, you need to retrieve the next chunk of data.

After getting the data, you can display it or conduct other calculations on it. The Intel(R) DCM: Energy Director displays the data after it gets data for all query types.

Code Sample: Adding a Managed Entity

This code sample shows how to add a managed entity using addEntity. The code for adding a group uses the same method and is very similar.

The code uses a base class called Entity.

This code sample begins by saving the properties you enter on the managed entity object. The managed entity object encapsulates the new managed entity.

From *DcmNodePropertiesWindow.mxml*:

```
node.bmcAddress = nodeBMCAddress.text;
node.bmcUserName = nodeBMCUser.text;
node.bmcPassword = nodeBMCPassword.text;
node.bmcKey = nodeBMCKey.text;
node.name = nodeName.text;
node.description = nodeDesc.text;
node.deratedPower = nodeDeratedPwr.text;
node.namePlatePower = nodeNameplatePwr.text;
node.dimension = nodeSize.text;
node.location = nodeLocation.text;
```

DcmNodePropertiesWindow.mxml also updates the managed entity properties. Therefore, you need to check if the managed entity already has an ID. From *DcmNodePropertiesWindow.mxml*:

```
updateNode = node.isValid();
if (updateNode) {...
```

After confirming that the managed entity has no pre-existing ID, add the new ID. From *DcmNodePropertiesWindow.mxml*:

```
} else {
        DcmPopUpManager.setBusyCursor(this);
        DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.
ADD_ENTITY_RESPONSE,
                addNodeEventHandler);
        DcmUICore.instance.dcmWS.addEntity(EntityType.NODE,
node.properties, false);
}
```

Next, call the method in the *DcmWebService* class that adds the entity. This method uses the managed entity property *properties*. First, it list the generic implementation of the *Entity* class, and then the override implementation of the *Node* class.

Here is the generic implementation of the *Entity* class. From *Entity.as*:

```
public function get properties():ArrayCollection {
        var arr:ArrayCollection = new ArrayCollection();
        arr.addItem(new Property(Property.NAME, _name));
        arr.addItem(new Property(Property.DESCRIPTION, description));
        arr.addItem(new Property(Property.LOCATION, location));
```

```
        arr.addItem(new Property(Property.DIMENSION, dimension));
        return arr;
}
```

Here is the override implementation of the *Node* class. From *Node.as*:

```
override public function get properties():ArrayCollection {
        var arr:ArrayCollection = super.properties;
        arr.addItem(new Property(Property.BMC_ADDRESS, bmcAddress));
        arr.addItem(new Property(Property.BMC_USER, bmcUserName));
        arr.addItem(new Property(Property.BMC_PASSWORD, bmcPassword));
        arr.addItem(new Property(Property.BMC_KEY, bmcKey));
        arr.addItem(new Property(Property.NAMEPLATE_PWR,
namePlatePower));
        arr.addItem(new Property(Property.DERATED_PWR, deratedPower));

        return arr;
}
```

The method addEntity in *DcmWebService* removes empty properties and sends all other properties to the addEntity web service method:

```
public function addEntity(entityType:EntityType,
properties:ArrayCollection, forceAddition:Boolean):AsyncToken
{
        var cursor:IViewCursor = properties.createCursor();
        while (!cursor.afterLast) {
                var prop:Property = cursor.current as Property;
                if (prop.value == null || prop.value.length == 0) {
                        cursor.remove();
                } else {
                        cursor.moveNext();
                }
        }
        var token:AsyncToken = service.addEntity(entityType.name,
properties, forceAddition);
        return fileToken(token,
DcmWsReturnValueEvent.ADD_ENTITY_RESPONSE);
}
```

Upon success, Intel(R) DCM returns an entity ID. Upon failure, Intel(R) DCM sends an error.

The following code handles the response. From *DcmNodePropertiesWindow.mxml*:

```
private function addNodeEventHandler(event:DcmWsReturnValueEvent):void
{
```

```
        DcmUICore.instance.dcmWS.removeEventListener(DcmWsReturnValueEve
nt.ADD_ENTITY_RESPONSE,
                addNodeEventHandler);
        if (!event.isError) {
                node.id = event.result as int;
                // Get the node's properties from DCM, to get the DCM
generated properties

        DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.
GET_ENTITY_PROPERTIES_RESPONSE ,
                        getNodePropertiesResponse);
                DcmUICore.instance.dcmWS.getEntityProperties(node.id);
        } else {
                DcmPopUpManager.removeBusyCursor(this);
                if (event.exception ==
DcmException.DcmCommunicationException) {
                        var msg:String = StringUtil.substitute(
                                resourceManager.getString("dcmui",
"NODEPROP_NODENOTREACHABLE_MSG"),
                                event.errorMessage);

                        forceAdditionMsgBox = DcmAlert.message(msg,
                                resourceManager.getString("dcmui",
"NODEPROP_NODENOTREACHABLE_TITLE"),
                               Alert.YES | Alert.NO | Alert.CANCEL, this,
forceAdditionMsgComplete, Alert.NO);
                        return;
                }
                DcmAlert.error(event.errorMessage, this);
                node.setProperties(origPropArr);
        }
}
```

Upon success, assign the new ID to the managed entity object and get the managed entity properties, including the read-only properties Intel(R) Datacenter Manager: Energy Director added to the managed entity.

Code Sample: Associating a Managed Entity with a Group

This code sample shows how to use associateEntity associate an existing managed entity with an existing group. To use this API, you supply the ID for the group and the ID for the managed entity, and call the web service.

The following code calls the hierarchy manager and associates the managed entity to the group. From *DcmAssociateWindow.mxml*:

```
private function associateToGroup(group:Group, entity:Entity):void {
        ++requestsLength;
        var tok:AsyncToken =
DcmUICore.instance.hierarchyManager.associateEntity(entity, group);
```

```
        requestsDictionary[tok.message.messageId] = [group, entity];
}
```

The following code calls the web service with the managed entity ID and the group ID. From *DcmHierarchyManager.as*:

```
public function associateEntity(child:Entity, parent:Group):AsyncToken
{
        var tok:AsyncToken =
DcmUICore.instance.dcmWS.associateEntity(parent.id, child.id);
        requestsDictionary[tok.message.messageId] = [parent, child];
        return tok;
}
```

Choose the managed entity ID and group ID to associate together. The following code shows the response handler that retrieves the group and managed entity objects that the user chose. From *DcmHierarchyManager.as*:

```
var parent:Group = (requestsDictionary[event.messageId])[0];
var child:Entity = (requestsDictionary[event.messageId])[1];
```

After getting the IDs for the managed entity and the group selected, add the managed entity to the group. Adding the managed entity to the group also updates the runtime hierarchy displayed. From *DcmHierarchyManager.as*:

```
if (event.isError) {
        DcmAlert.error(

        StringUtil.substitute(ResourceManager.getInstance().getString("d
cmui", "ASSOCIATEWND_ERRORASSOCIATE")
                        ,child.type.label ,child.name, parent.type.label,
parent.name), DcmUI.instance);
} else {
        parent.addEntity(child);
}
dispatchEvent(event.clone());
}
```

Code Sample: Configuring Monitoring

This code sample shows how to:

- enable or disable monitoring
- set global properties

Enabling or Disabling Monitoring

To enable or disable monitoring, call setCollectionState with a boolean value. *True* indicates monitoring enabled, *false* indicates monitoring disabled. From *DcmUICore.as*:

```
dcmWS.setCollectionState(_monitoringStatus);
```

Setting Global Properties

The following samples shows how to use setGlobalProperty to change any of the GlobalProperty. For example, enter the parameter NODE_POWER_SAMPLING_FREQUENCY to change the power sampling frequency. You can change multiple properties simultaneously. prop represents the global property type. changedGlobalProperties[prop] represents your new value. From *DcmUISettings.as*:

```
var token:AsyncToken = DcmUICore.instance.dcmWS.setGlobalProperty(prop,
changedGlobalProperties[prop]);
```

Intel(R) DCM: Energy Director validates the values of the global properties and tells you whether the call succeeded.

Here is the handler of the response to setGlobalProperty. If the call succeeded, it saves the new value instead of the old one. If an error occurred, it display the error. From *DcmUISettings.as*:

```
var prop:GlobalProperty = requestsDictionary[event.messageId];
if (event.isError) {
        if (event.errorMessage != null) {
                if (event.exception !=
DcmException.DcmNotSupportedException) {
                        var vRes:ValidationResult = new
ValidationResult(true, null, event.exception, event.errorMessage);
                        var vEvent:ValidationResultEvent = new
ValidationResultEvent(ValidationResultEvent.INVALID
                                , false, false, prop.label, new
Array(vRes));
                        dispatchEvent(vEvent);
                }
        } else {
                if (prop != null) {
                        --requestsLength;
                        setGlobalProp(prop);
                        return;
                }
```

```
        }
} else {
        // The process was successful. Commit the new value.
        globalProperties[prop] = changedGlobalProperties[prop];
}
++resultsCount;
delete changedGlobalProperties[prop];
delete requestsDictionary[event.messageId];
if (resultsCount == requestsLength) {
        dispatchEvent(new Event(SAVE_GLOBAL_PROPS_COMPLETE));
        if (!trendingSettings.validateTrendingFreq()) {
                DcmUI.instance.refreshTrendingView(true);
        }
}
```

Code Sample: Subscribing Predefined Events
This code sample shows how to subscribe and unsubscribe to predefined events.

The code carries out these steps:

- Adds all enabled predefined event types to the subscribed list, and adds all disabled predefined event types to the unsubscribed list.
- Adds event listeners for the subscribePredefinedEvent and unsubscribePredefinedEvent methods.
- Calls subscribePredefinedEvent and unsubscribePredefinedEvent methods.

From *DcmPredefinedEventsWindow.as*:

```
private function savePredefinedEvents():void {
        var subscribeEventsList:Array = [];
        var unsubscribeEventsList:Array = [];
        for each (var preDefEvent:PredefinedEventType in
PredefinedEventType.values) {
                if (preDefEvent.enabled) {
                        subscribeEventsList.push(preDefEvent.name);
                } else {
                        unsubscribeEventsList.push(preDefEvent.name);
                }
        }
        DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.
UNSUBSCRIBE_PREDEFINED_EVENT_RESPONSE,
                savePredefinedEventsResponseHandler);

        DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.
SUBSCRIBE_PREDEFINED_EVENT_RESPONSE,
                savePredefinedEventsResponseHandler);

        preDefEventsResponseCounter = 0;
```

```
        errorOccuredInPredefinedEventsUpdate = false;
        DcmPopUpManager.setBusyCursor(this);

        if (subscribeEventsList.length > 0) {
                ++preDefEventsResponseCounter;

        DcmUICore.instance.dcmWS.subscribePredefinedEvent(subscribeEvent
sList);
        }
        if (unsubscribeEventsList.length > 0) {
                ++preDefEventsResponseCounter;

        DcmUICore.instance.dcmWS.unsubscribePredefinedEvent(unsubscribeE
ventsList);
        }
}
```

Then you just need to wait for both responses to remove the events listeners and close the window.

From *DcmPredefinedEventsWindow.as*:

```
private function
savePredefinedEventsResponseHandler(event:DcmWsReturnValueEvent):void {
        --preDefEventsResponseCounter;

        errorOccuredInPredefinedEventsUpdate =
errorOccuredInPredefinedEventsUpdate || event.isError;

        if (event.isError) {
                Dumper.error(event);
        }

        if (preDefEventsResponseCounter == 0) {

        DcmUICore.instance.dcmWS.removeEventListener(DcmWsReturnValueEve
nt.UNSUBSCRIBE_PREDEFINED_EVENT_RESPONSE,
                        savePredefinedEventsResponseHandler);


        DcmUICore.instance.dcmWS.removeEventListener(DcmWsReturnValueEve
nt.SUBSCRIBE_PREDEFINED_EVENT_RESPONSE,
                        savePredefinedEventsResponseHandler);

                if (errorOccuredInPredefinedEventsUpdate) {

        DcmUICore.instance.eventsManager.initPredefinedEvents();
                        DcmAlert.error(resourceManager.getString("dcmui",
"EVENTSLIST_ERRORSETPREDEF"));
                }
```

```
                closeWindow();
        }
}
```

Code Sample: Getting Predefined Events

This code sample shows how to get the list of all subscribed predefined events, using the method getPredefinedEventTypes. This code sample comes from the file *DcmEventsManager.as*.

Here is the sample to call the method:

```
public function initPredefinedEvents():void {
                DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnVa
lueEvent.GET_PREDEFINED_EVENT_TYPES_RESPONSE,
                                getPredefinedEventTypesHandler);
                DcmUICore.instance.dcmWS.getPredefinedEventTypes();

}
```

And here we handle the response. Intel(R) DCM returns the list of predefined event types as a list of strings:

```
private function
getPredefinedEventTypesHandler(event:DcmWsReturnValueEvent):void {
        DcmUICore.instance.dcmWS.removeEventListener(DcmWsReturnValueEve
nt.GET_PREDEFINED_EVENT_TYPES_RESPONSE,
                getPredefinedEventTypesHandler);

        if (!event.isError) {
                var tmpArr:ArrayCollection;
                // We want a generic handling for one or more results
                // So if we got just one result, we will insert it into
                // an array and iterate over the array.
                if (event.result is ArrayCollection) {
                        tmpArr = event.result as ArrayCollection;
                } else {
                        tmpArr = new ArrayCollection();
                        if (event.result != null) {
                                tmpArr.addItem(event.result);
                        }
                }
                for each (var pEventT:PredefinedEventType in
PredefinedEventType.values) {
                        pEventT.enabled = false;
                }
                for each (var predefEvent:String in tmpArr) {
                        var pEventType:PredefinedEventType =
PredefinedEventType.fromString(predefEvent);
```

```
                        if (pEventType) {
                                pEventType.enabled = true;
                        }
                }
        }
        dispatchEvent(new FlexEvent(INIT_COMPLETE));
}
```

Code Sample: HTTP Event Handler
Event handlers must open an HTTP server.

Here is an example for opening an HTTP server in Java* language. Intel(R) DCM: Energy Director defines an anonymous *HttpHandler* class that handles HTTP requests for the HTTP path: http://localhost:8688/DcmMW/WsEventListener/

```
       HttpServer hServer = HttpServer.create(new
InetSocketAddress(8688), 0);
       hServer.createContext("/DcmMW/WsEventListener/", new
HttpHandler() {
           @Override
           public void handle(HttpExchange httpCon) throws IOException
{
              int responseCode = 200; // HTTP_OK
              StringBuilder sb = new StringBuilder();
              try {
                  BufferedReader in = new BufferedReader(
                         new
InputStreamReader(httpCon.getRequestBody()));
                  int lastChar;
                   while ((lastChar = in.read()) != 0) {
                     sb.append((char) lastChar);
                  }
                  // send response headers with response code
                  httpCon.sendResponseHeaders(responseCode, -1);
              } finally {
                  httpCon.close();
              }
               // Do some work on the event XML string
              //System.out.println(sb.toString());
           }
       });
       hServer.start();
```

To close the HTTP server, call:

```
hServer.stop(2); // wait for 2 seconds&ldots;
```

See Also

Handling Events

Event Message Example

Code Sample: Creating a Control Policy

This code sample shows how to create a new control policy. This sample comes from *DcmPolicyDetailsWindow.mxml*.

The sample begins by saving the properties the user enters on the policy object. The policy object encapsulates the new policy. Set the policy enable flag if the you do not schedule the policy. If the entity is a group, list high and low priority entities:

```
private function createPolicy():void {
        // Init the policy fields (not the schedule fields) first
        policy.entity = entity;
        if (permanentPolicy.selected) {
                policy.enabled = policyEnabled.selected;
        }
        policy.type = policyType.selectedItem as PolicyType;
        policy.threshold = getThreshold();
        policy.policyDescription = policyDescription.text;
        policy.lowPriorityList.length = 0;
        policy.highPriorityList.length = 0;
        if (childrenDataGrid.enabled) {
                // Fill the priority lists according to the user
selection.
                // Fill the lists only if the policy needs the
priorities list.
                for each (var entPrioriry:EntityPriorityRow in
childrenArray) {
                        if (entPrioriry.priority == EntityPriority.HIGH)
{

        policy.highPriorityList.push(entPrioriry.entity.id);
                        } else if (entPrioriry.priority ==
EntityPriority.LOW) {

        policy.lowPriorityList.push(entPrioriry.entity.id);
                        }
                }
        }
        DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.
SET_POLICY_RESPONSE
                , setPolicyResponseHandler);
        // Call the web service with the new policy
        DcmUICore.instance.dcmWS.setPolicy(policy);
}
```

Then call the setPolicy method in the web service class, as follows:

```
var token:AsyncToken = service.setPolicy(policy.entity.id,
policy.type.name,
                policy.threshold, policy.policyDescription,
policy.enabled, policy.lowPriorityList, policy.highPriorityList);
```

Then handle the response from Intel(R) DCM: Energy Director. If the call succeeds, set the policy ID and check to see if you need to schedule the policy:

```
if (event.error) {
                errorOccured(event.errorMessage);
} else {
                // If there was no error, just assign the returned
policy id to the policy.
                policy.id = event.result as int;
                // If the user didn't config the schedule, or the
schedule
                // is an empty schedule clode the window.
                // If the startSchedulePolicy returns true then it
means that the schedule
                // was changed so the schedule change has started and
we need to return and do nothing.
                // Otherwise, we need to close the window because all
the actions were finished.
                if (permanentPolicy.selected || !startSchedulePolicy())
{
                                closeWindow();
                }
}
```

Code Sample: Scheduling a Policy
This code sample shows you how to schedule a policy.

NOTE

You can only schedule a policy if the policy is disabled.

This code sample carries out the following steps:

1. Makes sure the policy is disabled.
2. Updates the schedule.
3. Calls schedulePolicy to schedule the policy.

Make sure the policy is disabled:

```
// If the policy is enabled, we need to disable it before setting the
schedule.
if (policy.enabled) {
```

```
        DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.
SET_POLICY_STATE_RESPONSE
                , disablePolicyBeforeUpdateResponseHandler);

        DcmUICore.instance.dcmWS.setPolicyState(policy.id, false);
} else {
        updatePolicy();
}
```

Update the schedule:

```
policy.enabled = policyEnabled.selected;
policy.clearSchedule();
if (!permanentPolicy.selected) {
        if (enaApplyBetween.selected) {
                policy.startHour = dateToMinuts(startHour.time);
                policy.endHour = dateToMinuts(endHour.time);
        }
        if (enaStartTime.selected) {
                policy.startTime = startTime.time;
        }
        if (enaEndTime.selected) {
                policy.endTime = endTime.time;
        }
        policy.days.length = 0;
        for each (var cBox:CheckBox in daysTile.getChildren()) {
                if (cBox.selected) {
                        policy.days.push(cBox.name);
                }
        }
}
DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.SCHEDUL
E_POLICY_RESPONSE
        , schedulePolicyResponseHandler);

DcmUICore.instance.dcmWS.schedulePolicy(policy);
```

Call `schedulePolicy`:

```
public function schedulePolicy(policy:Policy):AsyncToken
{
      var token:AsyncToken = service.schedulePolicy(policy.id,
policy.startTime,
            policy.endTime, policy.startHour, policy.endHour,
policy.days);
      return fileToken(token,
DcmWsReturnValueEvent.SCHEDULE_POLICY_RESPONSE);
}
```

Code Sample: Adding a Custom Event

The following code adds a custom event.

From *DcmCustomEventDetailsWindow.as*:

```
this.customEvent.entity = entity;
this.customEvent.type = customEventType.selectedItem as CustomEventType;
this.customEvent.threshold = parseInt(threshold.text);
this.customEvent.conditionOp = conditionOp.selectedItem as
ConditionOperator;
this.customEvent.evalPeriod = parseInt(evalPeriod.text);
this.customEvent.eventDescription = eventDescription.text;
...
// New custom event.
this.customEvent.enabled = customEventEnabled.selected;

DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.DEFINE_
CUSTOM_EVENT_RESPONSE
      , defineCustomEventResponseHandler);

DcmUICore.instance.dcmWS.defineCustomEvent(customEvent);

In DcmWebService.as:
public function defineCustomEvent(customEvent:CustomEvent):AsyncToken
{
       var token:AsyncToken =
service.defineCustomEvent(customEvent.entity.id, customEvent.type.name,
               customEvent.conditionOp.name, customEvent.threshold,
customEvent.evalPeriod,
               customEvent.eventDescription, customEvent.enabled);
       return fileToken(token,
DcmWsReturnValueEvent.DEFINE_CUSTOM_EVENT_RESPONSE);
}
```

If the event was successfully added, assign the custom event ID to the custom event data and close the window:

```
customEvent.id = event.result as int;


closeWindow();
```

Code Sample: Updating a Custom Event

This code sample shows you how to update an event.

NOTE

You can only update an event if the event is disabled.

This code sample carries out the following steps:

1. Makes sure the event is disabled.
2. Updates the enabled flag for the UI.
3. Calls updateCustomEvent.

Make sure the event is disabled:

```
if (this.customEvent.enabled) {
      DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.S
ET_CUSTOM_EVENT_STATE_RESPONSE
            , disableEventBeforeUpdateResponseHandler);
      DcmUICore.instance.dcmWS.setEventState(customEvent.id, false);
} else {
      updateCustomEvent();
}
```

Set the state as disabled, and then call the update API:

```
// Set the new enabled state
this.customEvent.enabled = customEventEnabled.selected;

DcmUICore.instance.dcmWS.addEventListener(DcmWsReturnValueEvent.UPDATE_
CUSTOM_EVENT_RESPONSE
      , updateCustomEventResponseHandler);

DcmUICore.instance.dcmWS.updateCustomEvent(customEvent);

In DcmWebService.as:
public function updateCustomEvent(customEvent:CustomEvent):AsyncToken
{
       var token:AsyncToken = service.updateCustomEvent(customEvent.id,
              customEvent.threshold, customEvent.evalPeriod,
customEvent.eventDescription);
       return fileToken(token,
DcmWsReturnValueEvent.UPDATE_CUSTOM_EVENT_RESPONSE);
}
```

Code Sample: Importing Hierarchy from File

This code sample shows how to import the datacenter hierarchy from a file.

The reference UI downloads the hierarchy file from the computer on which Intel(R) DCM: Energy Director is installed.

Here it calls the import method with the filename appended to the local path from the UI settings. From *DcmImportHierarchyWindow.as*:

```
DcmUICore.instance.dcmWS.addEventListener(
```

```
        DcmWsReturnValueEvent.IMPORT_HIERARCHY_RESPONSE,
importResponseHandler);

DcmUICore.instance.eventsManager.addEventListener(
        DcmEvent.INTERNAL_EVENT, importReportHandler);
DcmUICore.instance.dcmWS.importHierarchy(
        DcmUICore.instance.dcFilesLocation + "/" + fileName,
forceAddition.selected);
```

Then wait for the response. If the call succeeded, load the new hierarchy from Intel(R) DCM: Energy Director and wait for the event with the import result. If Intel(R) DCM: Energy Director receives this event, it is shown. If an error occurred, it shows the error:

```
DcmUICore.instance.dcmWS.removeEventListener(
        DcmWsReturnValueEvent.IMPORT_HIERARCHY_RESPONSE,
importResponseHandler);
if (event.isError) {
        DcmUICore.instance.eventsManager.removeEventListener(
                DcmEvent.INTERNAL_EVENT, importReportHandler);

        DcmAlert.error(event.errorMessage);
        closeWindow();
} else {
        progressBar.mode = ProgressBarMode.MANUAL;
        progressBar.indeterminate = false;
        progressBar.label = resourceManager.getString("dcmui",
"INIT_LOADHIERARCHY");
        progressBar.setProgress(0, 3);
        DcmUICore.instance.hierarchyManager.addEventListener(DcmHierarch
yManager.INIT_COMPLETE, hierarchyInitComplete);
        DcmUICore.instance.hierarchyManager.addEventListener(FlexEvent.I
NIT_PROGRESS, hierarchyInitProgress);
        DcmUICore.instance.hierarchyManager.init();
}
private function hierarchyInitComplete( event:Event ):void
{
        DcmUICore.instance.eventsManager.removeEventListener(
                DcmEvent.INTERNAL_EVENT, importReportHandler);
        DcmUICore.instance.hierarchyManager.removeEventListener(
                DcmHierarchyManager.INIT_COMPLETE,
hierarchyInitComplete);
        DcmUICore.instance.hierarchyManager.removeEventListener(
                FlexEvent.INIT_PROGRESS, HierarchyInitProgress);

        DcmUI.instance.dcmHierarchyTree.expandTree();

        if (importReport != null) {
                DcmAlert.message(importReport,
```

```
                        resourceManager.getString("dcmui",
"IMPORT_REPORT_TITLE")
                        , Alert.OK, DcmUI.instance);
        }

        closeWindow();
}
```

See Also

Importing or Exporting Hierarchy Files

Hierarchy File Example

Code Sample: Exporting Hierarchy to File

This code sample shows how to export the datacenter hierarchy to a file. This sample carries out the following steps:

1. tells Intel(R) DCM: Energy Director where to save the file
2. downloads the file
3. saves the file to the client computer

First, Intel(R) DCM: Energy Director creates a new UUID for the file name. Intel(R) DCM: Energy Director appends this file name to the configured path and call exportHierarchy:

```
fileName = UIDUtil.createUID() + ".xml";

DcmUICore.instance.dcmWS.addEventListener(
      DcmWsReturnValueEvent.EXPORT_HIERARCHY_RESPONSE,
exportResponseHandler);
DcmUICore.instance.dcmWS.exportHierarchy(DcmUICore.instance.dcFilesLoca
tion + fileName);
```

When the response successfully returns, Intel(R) DCM: Energy Director downloads the file from the configured URL:

```
private function
exportResponseHandler(event:DcmWsReturnValueEvent):void {
        DcmUICore.instance.dcmWS.removeEventListener(
                DcmWsReturnValueEvent.EXPORT_HIERARCHY_RESPONSE,
exportResponseHandler);

        if (event.isError) {
                DcmAlert.error(event.errorMessage);
                closeWindow();
                return;
```

```
        }

        DcmPopUpManager.removeBusyCursor(this);

        progressBar.mode = ProgressBarMode.MANUAL;
        progressBar.indeterminate = false;
        progressBar.label = resourceManager.getString("dcmui",
"EXPORT_WINDOW_FILE_READY");

        downloadButton.enabled = true;
   }
private function downloadButtonClick():void {
        DcmPopUpManager.setBusyCursor(this);

       progressBar.label = StringUtil.substitute(progressBarMessage, 0);

        // If there was no error, download the file.
        fileRef = new FileReference();
        fileRef.addEventListener(Event.COMPLETE, downloadComplete);
        fileRef.addEventListener(ProgressEvent.PROGRESS,
progressHandler);
        fileRef.addEventListener(Event.CANCEL, cancelHandler);
        fileRef.addEventListener(IOErrorEvent.IO_ERROR, errorHandler);
        fileRef.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
errorHandler);

        var request:URLRequest = new
URLRequest(DcmUICore.instance.dcFilesURL + "/" + fileName);
        request.method = URLRequestMethod.GET;


        var time:Date = new Date();

        var localFileName:String = "dchierarchy_";
       localFileName += time.getFullYear().toString() + (time.getMonth()
+ 1).toString() + time.getDate().toString();
        localFileName += "_" + time.getHours().toString() +
time.getMinutes().toString();
        localFileName += ".xml";

        // This will open a browse window for the user to select the
location
        // to save the downloaded file.
        fileRef.download(request, localFileName);
}
```

See Also

Importing or Exporting Hierarchy Files

Hierarchy File Example

Sample UI Context-Sensitive Help

Sample UI Context-Sensitive Help: Overview

This section contains context-sensitive help for the sample user interface. When you click on the **?** button from the sample user interface, a new window opens with the appropriate page of the help document.

See Also

Sample User Interface: Overview

Trending Graph

The trending graph appears in the middle of the reference UI main page. The trending graph shows the measurements for the power and thermal metrics you choose to display. The **Legend** to the right of the trending graph defines each color on the graph.

To choose the trending data to show in the graph, go to **Configuration** > **Trending...** in the upper left of the reference UI window.

See Also

Trending Options

Query/Metrics Interface: Overview

Power Recommendation Graph

The power recommendation graph shows the measurements for power metrics, and the constant line shows the recommendations. You can specify a history period to query and it will show you the recommendations.

To change the options for the power recommendation graph, go to the upper right of the reference UI window to change the history period.

See Also

Query/Metrics Interface: Overview

Datacenter Hierarchy

On the left side of the reference UI main page is a section titled **Datacenter Hierarchy**. The **Datacenter Hierarchy** section displays all managed entities and groups in Intel(R) DCM: Energy Director. Click on a group name to see all managed entities and groups included in that group.

To add a managed entity or group, right-click underneath **Datacenter Hierarchy**. When you choose either **Add Managed Entity** or **Add Group**, the respective dialog box opens.

To modify the data center hierarchy, drag and drop managed entities or groups in the Datacenter Hierarchy section.

To perform any action on an existing entity, right click on that entity and choose the action from the pop-up menu.

See Also

Datacenter Modeling Interface: Overview

Setting the Datacenter Hierarchy

Metrics Pane
In the metrics pane, you can choose to show metrics returned by Intel(R) Data Center Manager.

To add a metric, press **+** and choose the desired metric from the drop-down list.

To remove a selected metric, press **-**.

See Also

MetricType

Query/Metrics Interface: Overview

Managed Entity Details Pane
The Managed Entity Details pane includes three tabs:

- **'<your_entity>' Details** shows all properties of your managed entity.
- **'<your_entity>' Events** shows details about all events on your managed entity.
- **System Events** shows system events.
- **Action Log** shows all entries in the Action Log, which includes:
  - Every action taken by the user through the API
  - All custom events
  - All predefined events
  - Activation of an event or policy

See Also

ActionLogType

EntityProperty

Events Interface: Overview

Policy Details
To create a new policy, enter the following properties that define your policy. Unless noted, all properties are mandatory:

- **Description** (optional). A string describing your policy.
- **Policy type**. The type of policy.
- **Entity**: The entity to which to apply the policy.
- **Threshold**. Can be used in two ways:
  - For policy type CUSTOM_PWR_LIMIT, threshold sets the power limit.
  - For policy type MIN_PWR_ON_INLET_TEMP_TRIGGER threshold sets the temperature beyond which the policy is triggered.
- **Policy Enabled**. Sets whether to enable the policy.
- **Children Priorities**. Only for policy type CUSTOM_PWR_LIMIT. The priority levels for each entity to which this policy applies.

See Also

PolicyType

setPolicy

Policy Priority Levels

Custom Events
The **Custom Events** dialog lists all custom events. You can sort the events by:

- Description
- Entity name
- Event type
- Enabled/disabled status

From the **Custom Events** dialog, you can take the following actions:

- Create a new event, by pressing the **New...** button.

- Create a new event based on an existing event, by pressing the **Copy...** button.
- Edit an existing event, by pressing the **Edit...** button.
- Delete an existing event, by pressing the **Delete** button.

See Also

CustomEventType

Custom Event Details

When you create a new custom event, or edit an existing event. Enter the following properties that define your custom event. Unless noted, all properties are mandatory:

- **Description** (optional). A string describing your event.
- **Custom event type**. The type of custom event. See the Developer's Guide for more details on the event types.
- **Entity**. The entity to which to apply the custom event.
- **Condition operator**. Sets whether an event is triggered when the value measured is **Greater Than** or **Less Than** the threshold. The value measured is determined by the Custom event type.
- **Threshold**:.The threshold beyond which the event is triggered.
- **Evaluation Period**. The amount of time, in seconds, that Intel(R) DCM: Energy Director requires the event condition to persist before triggering the event.

See Also

CustomEventType

Associate Entities

The **Associate <your_entity>** dialog enables you to associate the selected entity with other entities. You can also associate entities from the Datacenter Hierarchy window.

If the selected entity is a managed entity, you can only associate or disassociate your managed entity to available groups.

If the selected entity is a group, you can choose between two types of association:

- Associate your group to available groups. (Default)

- Associate available entities to your group.

Choose a type of association from the **Associate** drop-down list at the top of the screen.

To associate your entity to an available group:

1. From the **Available Group List**, select the group to which you want to associate your entity.
2. Click **Associate**.

To disassociate your entity from a group:

1. From the list **<your_entity> is Associated to:**, click the group from which you want to disassociate.
2. Click **Disassociate**.

To associate entities to your group:

1. From the **Available Entities List**, select the entity to associate with your group.
2. Click **Associate**.

To disassociate entities from your group:

1. From the list **Children of <your_group>**, select the entity to disassociate from your group.
2. Click **Disassociate**.

See Also

Datacenter Modeling Interface: Overview

associateEntity

disassociateEntity

Predefined Events Dialog
The Predefined Events dialog enables you to select:

- The predefined event types for which you want to receive notification
- The severity levels for which you want to be notified of events

See Also

PredefinedEventType

SeverityLevel

Group Properties Dialog
The **Group Properties** dialog shows all properties for a group. You can use this dialog to do the following:

- Add a group
- Update group properties
- See group properties

See the description of all group properties here: EntityProperty.

Find Managed Entity/Group Dialog
The **Find Managed Entity/Group** dialog shows all nodes and groups in Intel(R) DCM: Energy Director. You can sort managed entities and groups by entity type or by name.

Right-click an entity from the list, to do, or view, the following:

- **Delete.** Deletes the selected entity from Intel(R) DCM: Energy Director.
- **Properties.** Shows you the properties of the selected entity (either Managed Entity Properties Dialog or Group Properties Dialog).
- **Associate.** Enables you to associate the selected entity with other entities, through the Associate Entities dialog.

See Also

Associate Entities Dialog

Managed Entity Properties Dialog

Group Properties Dialog

Datacenter Modeling Interface: Overview

EntityType

Managed Entity Properties
The **Managed Entity Properties** window shows all properties for a managed entity.

Right-click an entity from the list, to do, or view, the following:

- Add a managed entity
- Update managed entity properties
- View managed entity properties

For description of all managed entity properties, see EntityProperty.

Options Dialog
The Options dialog enables you to set the Intel(R) DCM: Energy Director global properties. For descriptions of the properties, see GlobalProperty.

See Also

Configuration Interface: Overview

Policies List
The **Policies List** dialog shows a list of all policies. You can sort the policies by policy type or by entity name. From this window, you can take the following actions:

- Create a new policy, by pressing the **New...** button.
- Create a policy based on an existing policy, by pressing the **Copy...** button.
- Edit an existing policy, by pressing the **Edit...** button.
- Delete an existing policy, by pressing the **Delete** button.

See Also

Control Policies Interface: Overview

Trending Options
From the Trending Options dialog, you can configure the following options:

- Dynamic or Static trending
- Graph refresh frequency
- Temperature units
- Trending data to show
- Policy data to show

See Also

Query/Metrics Interface: Overview

Control Policies Interface: Overview

Setting the Datacenter Hierarchy

Setting the Datacenter Hierarchy

You can set the datacenter hierarchy in two ways:

- By calling a method for each change
- By importing the hierarchy from a file

See Also

Manually Setting Hierarchy

Importing or Exporting Hierarchy Files

Control Policies and Datacenter Hierarchy

Nodes and Groups

Manually Setting Hierarchy

You can manually set the Intel(R) DCM: Energy Director hierarchy. Specifically, you can:

- Add or remove nodes and groups
- Associate existing entities with groups, or disassociate entities from groups
- Get and set entity properties

Each node can be included in any number of logical groups. Each node can be included in one physical group of each type: one enclosure, one rack, one row, one room and one datacenter.

Each group can be included in any number of other logical groups. Logical groups cannot be included in physical groups. Each physical group can be included in one higher-level physical group of each type. For example, each rack can be included in one row and one room, each row can be included in one room.

When you add a node or an enclosure which corresponds to a physical device, you need to specify a matching connector, which is used to communicate with the device. The table below lists the connectors along with the devices supported.

Connector	Protocol	Device
`com.intel.dcm.plugin.Nm15Plugin`	IPMI	Intel(R) Intelligent Power Node Manager 1.5

		servers
com.intel.dcm.plugin.Nm20Plugin	IPMI	Intel(R) Intelligent Power Node Manager 2.0 servers
com.intel.dcm.plugin.Dcmi10Plugin	IPMI	DCMI 1.0 servers, for example, Cisco* rack servers, HP* iLO2/iLO3/iLO4 servers/blades, Fujitsu servers with DCMI interface exposed
com.intel.dcm.plugin.hpiLOPlugin	IPMI (with SSH for auxiliary tasks)	HP* iLO2/iLO3/iLO4 servers with IPMI interface and SSH interface exposed
com.intel.dcm.plugin.dellIdrac6Plugin	IPMI	Dell* iDRAC6 servers
com.intel.dcm.plugin.dellIdrac7Plugin	IPMI	Dell* iDRAC7 servers
com.intel.dcm.plugin.Ipmi20Plugin	IPMI	Generic IPMI 2.0 servers, for example, common IBM* rack servers with IPMI interface exposed
com.intel.dcm.plugin.apcPduPlugin	SNMP	APC* PDUs
com.intel.dcm.plugin.avocentPduPlugin	SNMP	Avocent* PDUs

com.intel.dcm.plugin.dellPduPlugin	SNMP	Dell* PDUs
com.intel.dcm.plugin.eatonPduPlugin	SNMP	Eaton* PDUs
com.intel.dcm.plugin.emersonPduPlugin	SNMP	Emerson* PDUs
com.intel.dcm.plugin.serverTechPduPlugin	SNMP	ServerTech* PDUs
com.intel.dcm.plugin.raritanPduPlugin	SNMP	Raritan* PDUs
com.intel.dcm.plugin.chatsWorthPduPlugin	SNMP	Chatsworth PDUs
com.intel.dcm.plugin.apcUpsPlugin	SNMP	APC* UPSes
com.intel.dcm.plugin.dellUpsPlugin	SNMP	Dell* UPSes
com.intel.dcm.plugin.eatonUpsPlugin	SNMP	Eaton* UPSes
com.intel.dcm.plugin.emersonUpsPlugin	SNMP	Emerson* UPSes
com.intel.dcm.plugin.ciscoSwitchPlugin	SNMP	Cisco* switches with Cisco* EnergyWise Technology enabled
com.intel.dcm.plugin.snmpPlugin	SNMP	Generic SNMP devices managed according to rules in the configuration file
com.intel.dcm.plugin.ibmAMMPlugin	SSH	IBM* blade enclosures with AMM interface exposed
com.intel.dcm.plugin.ibmBladePlugin	SSH	IBM* blade servers inside an IBM* enclosure
com.intel.dcm.plugin.hpBladeOAPlugin	SSH	HP* blade enclosures with OA interface

		exposed
com.intel.dcm.plugin.hpBladeServerPlugin	SSH	HP* blade servers inside an HP* enclosure
com.intel.dcm.plugin.dellCMCSSHPlugin	SSH	Dell* CMC enclosures with SSH opened for connection
com.intel.dcm.plugin.dellBladePlugin	SSH	Dell* blade servers inside a Dell* enclosure
com.intel.dcm.plugin.hpSLAPMPlugin	SSH	HP* SL enclosures with management interface exposed from HP* SL APM
com.intel.dcm.plugin.sshPlugin	SSH	Generic SSH devices managed according to rules in the configuration file
com.intel.dcm.plugin.dellCMCPlugin	HTTPS/ WS-MAN	Dell* CMC enclosures with HTTPS/WS-MAN interface exposed
com.intel.dcm.plugin.ciscoUCSChassisPlugin	HTTPS/ XML	Cisco* UCS chassis/enclosures with HTTPS/ XML interface exposed from Cisco* UCS Manager

com.intel.dcm.plugin.ciscoUCSPlugin	HTTPS/ XML	Cisco* UCS servers with HTTPS/ XML interface exposed from Cisco* UCS Manager
com.intel.dcm.plugin.WMIWindowsPlugin	WMI	Microsoft* Windows servers with WMI exposed
com.intel.dcm.plugin.SSHLinuxPlugin	SSH on Linux	Red Hat Enterprise Linux Servers* and Novell SUSE Linux Enterprise Servers* with SSH opened for connection
com.intel.dcm.plugin.SSHESXPlugin	SSH on ES/EXSi	VMWare* ESX/ESXi Servers with SSH opened for connection
com.intel.dcm.plugin.SSHXenPlugin	SSH on Xen	Xen servers with SSH opened for connection
com.intel.dcm.plugin.DummyPlugin	N/A	Dummy entities in hierarchy for servers which do not have real communication with Intel(R) DCM
com.intel.dcm.plugin.DummyEnclosurePlugin	N/A	Dummy entities in hierarchy for enclosure which

		do not have real communication with Intel(R) DCM

Intel(R) DCM: Energy Director also provides auxiliary APIs to help management software build the hierarchy, for example, identify the appropriate connectors to communicate with devices in an IP range by specifying the protocols as well as the credentials.

Through Intel(R) DCM: Energy Director, management software is able to communicate with a blade enclosure to retrieve the individual blade server information under the enclosure so that the individual blade servers can be added into Intel(R) DCM: Energy Director and associated with the enclosure. In the standalone deployment of Intel(R) DCM: Energy Director, management software is also able to build and recover the physical hierarchy under IBM*, HP*, Dell* (managed with the connector `com.intel.dcm.plugin.dellCMCSSHPlugin`), or Cisco* UCS enclosures/chassis automatically.

NOTE

To secure the remote WMI connection in managing Microsoft* Windows servers through WMI, please refer to http://msdn.microsoft.com/library/aa393266.aspx.

NOTE

Intel(R) DCM does not support identifying the connector for Cisco* UCS devices.

See Also

addEntity

removeEntity

associateEntity

disassociateEntity

rediscoverEntity

setEntityProperties

getEntityProperties

identifyEntity

IdentifyEntityByProtocol

discoverEntities

getEnclosureAndBladeInfo

Importing or Exporting Hierarchy Files

Instead of manually setting each element of the Intel(R) Datacenter Manager: Energy Director hierarchy, you can use `importHierarchy` to import the hierarchy from a file.

You can use `exportHierarchy` to export the Intel(R) DCM: Energy Director hierarchy to a file.

The import and export methods get a string for the path of the XML hierarchy file. Intel(R) DCM: Energy Director should be able to access this path. For example, you can put the files on the local computer on which Intel(R) Datacenter Manager: Energy Director is running.

The hierarchy file is an XML file that lists entities and the associations between those entities. Each entity listed includes `EntityProperty`.

The *UNIQUE_ID* attribute is mandatory and must be unique in the file. Once an entity is added to Intel(R) DCM: Energy Director, Intel(R) DCM: Energy Director provides it with an entityId that is not related to the *UNIQUE_ID*.

The following is the schema for the hierarchy file:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="EntityType">
<xs:restriction base="xs:string">
<xs:enumeration value="DATACENTER" />
<xs:enumeration value="ROOM" />
<xs:enumeration value="ROW" />
<xs:enumeration value="RACK" />
<xs:enumeration value="ENCLOSURE" />
<xs:enumeration value="NODE" />
<xs:enumeration value="LOGICAL_GROUP" />
<xs:enumeration value="DCM_SERVER" />
</xs:restriction>
</xs:simpleType>
<xs:element name="hierarchy">
<xs:complexType>
<xs:sequence minOccurs="0" maxOccurs="unbounded">
<xs:element name="entity">
<xs:complexType>
<xs:sequence>
```

```
<xs:element name="member" minOccurs="0" maxOccurs="unbounded"
type="xs:IDREFS" />
<xs:element name="custprop" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="NAME" type="xs:string" use="required" />
<xs:attribute name="VALUE" type="xs:string" use="required" />
<xs:attribute name="SECRET" type="xs:boolean" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="ENTITY_TYPE" type="EntityType" use="required" />
<xs:attribute name="UNIQUE_ID" type="xs:ID" use="required" />
<xs:attribute name="DCM_SERVER_RID" type="xs:IDREFS" />
<xs:attribute name="BMC_ADDRESS" type="xs:string" />
<xs:attribute name="NAME" type="xs:string" />
<xs:attribute name="DESCRIPTION" type="xs:string" />
<xs:attribute name="PDU_PWR_LIMIT" type="xs:int" />
<xs:attribute name="NAMEPLATE_PWR_UNMNGD_EQPMNT" type="xs:int" />
<xs:attribute name="DERATED_PWR" type="xs:int" />
<xs:attribute name="NAMEPLATE_PWR" type="xs:int" />
<xs:attribute name="DIMENSION" type="xs:float" />
<xs:attribute name="LOCATION" type="xs:float" />
<xs:attribute name="BMC_PASSWORD" type="xs:string" />
<xs:attribute name="BMC_KEY" type="xs:string" />
<xs:attribute name="BMC_USER" type="xs:string" />
<xs:attribute name="CONNECTOR_NAME" type="xs:string" />
<xs:attribute name="SNMP_ADDRESS" type="xs:string" />
<xs:attribute name="SNMP_PROTOCOL" type="xs:string" />
<xs:attribute name="SNMP_COMMUNITY_STRING" type="xs:string" />
<xs:attribute name="SNMP_USER" type="xs:string" />
<xs:attribute name="SNMP_AUTHENTICATION_PASSWORD" type="xs:string" />
<xs:attribute name="SNMP_ENCRYPTION_PASSWORD" type="xs:string" />
<xs:attribute name="SSH_ADDRESS" type="xs:string" />
<xs:attribute name="SSH_USER" type="xs:string" />
<xs:attribute name="SSH_PASSWORD" type="xs:string" />
<xs:attribute name="SSH_PORT" type="xs:string" />
<xs:attribute name="DCM_SERVER_ADDRESS" type="xs:string" />
<xs:attribute name="DCM_SERVER_PORT" type="xs:int" />
<xs:attribute name="EVENT_HANDLER_HOST_ADDRESS" type="xs:string" />
<xs:attribute name="EVENT_HANDLER_PORT" type="xs:int" />
<xs:attribute name="ENABLE_TLS" type="xs:string" />
<xs:attribute name="SERVER_POWER_SAMPLING_PERIOD" type="xs:int" />
<xs:attribute name="SERVER_THERMAL_SAMPLING_PERIOD" type="xs:int" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="ENC_KEY" type="xs:string" />
</xs:complexType>
</xs:element>
</xs:schema>
```

See Also

EntityProperty

importHierarchy

exportHierarchy

Hierarchy File Example

Hierarchy File Example

The Intel(R) DCM: Energy Director installation includes a sample hierarchy file. To use this file, add information about your nodes. By default, this file installs to the sample files directory: C:\Program Files\Intel\DataCenterManager\Samples

Follow these steps to create the encryption key and entity write-only properties for the hierarchy file:

1. Create an encryption key.
2. Encode the encryption key in base64.
3. Use submitSecretKey with the base64 encoded encryption key to create an encryption handle.
4. Enter the returned encryption handle as the `ENC_KEY` value in the hierarchy file.
5. For each entity, do the following:

   1) Create the properties which are imported as write-only properties, e.g., `BMC_PASSWORD`, or `SSH_PASSWORD`.

   2) Use the encryption key to encrypt the write-only properties.

   3) Encode the encrypted write-only properties in base64.

   4) Enter the encoded encrypted write-only properties as the `BMC_PASSWORD`, or `SSH_PASSWORD`.

The following example shows a file that sets the node hierarchy in Intel(R) Datacenter Manager: Energy Director:

NOTE

This example uses a sample encryption key and node BMC passwords. For the file to work, you must use your own encryption key and password.

```
<?xml version="1.0" encoding="utf-8"?>
<hierarchy ENC_KEY="52b852eb38f429e2c3e9bfb16deb3e13a5a9b909">
```

```
       <entity UNIQUE_ID="iDC" ENTITY_TYPE="DATACENTER" NAME="DC"
DESCRIPTION="Main datacenter">
            <member>iR1</member>
            <member>iR2</member>
      </entity>
      <entity UNIQUE_ID="iR1" ENTITY_TYPE="ROOM" NAME="Room 1"
DESCRIPTION="Server room">
            <member>n1</member>
            <member>n2</member>
            <member>n3</member>
      </entity>
            <entity UNIQUE_ID="iR2" ENTITY_TYPE="ROOM" NAME="Room 1"
DESCRIPTION="Server room">
            <member>n4</member>
            <member>n5</member>
            <member>n6</member>
      </entity>
      <entity UNIQUE_ID="n1" ENTITY_TYPE="NODE" NAME="node 1"
                  DESCRIPTION="first node with special app"
DERATED_PWR="400"
                  NAMEPLATE_PWR="600" BMC_ADDRESS="12.34.56.00"
BMC_USER="Admin"
                  BMC_PASSWORD="F4hm4+6XZLRTIAc1eUxX9Q==" LOCATION="3"
DIMENSION="2"/>
      <entity UNIQUE_ID="n2" ENTITY_TYPE="NODE" NAME="node 1"
                  DESCRIPTION="first node with special app"
DERATED_PWR="400"
                  NAMEPLATE_PWR="600" BMC_ADDRESS="12.34.56.01"
BMC_USER="Admin"
                  BMC_PASSWORD="F4hm4+6XZLRTIAc1eUxX9Q==" LOCATION="3"
DIMENSION="2"/>
      <entity UNIQUE_ID="n3" ENTITY_TYPE="NODE" NAME="node 1"
                  DESCRIPTION="first node with special app"
DERATED_PWR="400"
                  NAMEPLATE_PWR="600" BMC_ADDRESS="12.34.56.02"
BMC_USER="Admin"
                  BMC_PASSWORD="F4hm4+6XZLRTIAc1eUxX9Q==" LOCATION="3"
DIMENSION="2"/>
      <entity UNIQUE_ID="n4" ENTITY_TYPE="NODE" NAME="node 1"
                  DESCRIPTION="first node with special app"
DERATED_PWR="400"
                  NAMEPLATE_PWR="600" BMC_ADDRESS="12.34.56.03"
BMC_USER="Admin"
                  BMC_PASSWORD="1234" LOCATION="3" DIMENSION="2"/>
      <entity UNIQUE_ID="n5" ENTITY_TYPE="NODE" NAME="node 1"
                  DESCRIPTION="first node with special app"
DERATED_PWR="400"
                  NAMEPLATE_PWR="600" BMC_ADDRESS="12.34.56.04"
BMC_USER="Admin"
                  BMC_PASSWORD="1234" LOCATION="3" DIMENSION="2"/>
      <entity UNIQUE_ID="n6" ENTITY_TYPE="NODE" NAME="node 1"
                  DESCRIPTION="first node with special app"
DERATED_PWR="400"
                  NAMEPLATE_PWR="600" BMC_ADDRESS="12.34.56.05"
BMC_USER="Admin"
                  BMC_PASSWORD="1234" LOCATION="3" DIMENSION="2"/>
```

```
      <entity UNIQUE_ID="l3" ENTITY_TYPE="LOGICAL_GROUP"
DESCRIPTION="Application Servers" >
            <member>n1</member>
            <member>n2</member>
            <member>n6</member>
      </entity>
      <entity UNIQUE_ID="l4" ENTITY_TYPE="LOGICAL_GROUP"
DESCRIPTION="SMTP Servers" >
      </entity>
      <entity UNIQUE_ID="l5" ENTITY_TYPE="LOGICAL_GROUP"
DESCRIPTION="Company X servers" >
      </entity>
      <entity UNIQUE_ID="l6" ENTITY_TYPE="LOGICAL_GROUP"
DESCRIPTION="Company Y servers" >
      </entity>
      <entity UNIQUE_ID="l7" ENTITY_TYPE="LOGICAL_GROUP"
DESCRIPTION="Company Z servers" >
      </entity>
</hierarchy>
```

Monitoring the Datacenter

Monitoring the Datacenter: Overview

Intel(R) DCM: Energy Director monitors the power consumption and inlet temperature of the devices in data centers periodically. The historical data is stored in an internal database. Different types of data are monitored based on the device capabilities, including, e.g.,

- Average power consumption within a period
- Instantaneous power for a very short period
- Inlet temperature

The table below shows the available data types, and what devices they apply to.

Data Type	Device
Average/maximum/minimum power	Devices with average power monitoring capability
Instantaneous power	Devices with instantaneous power monitoring capability
Estimated power	Devices with the power estimation capability
IT equipment power/energy	IT equipment devices (not PDUs and UPSes)
PDU power and observed maximum PDU power	PDU devices with instantaneous power monitoring capability
Average/maximum/minimum inlet temperature	Devices with inlet temperature monitoring capability
Average outlet temperature and airflow	Devices with both outlet temperature monitoring capability and airflow monitoring capability
Average/maximum/minimum CPU power	Devices with CPU power monitoring capability
Average/maximum/minimum	Devices with memory power

memory power	monitoring capability
CPU utilization	Devices managed with in-band mechanism (e.g., WMI on Windows, SSH on Linux)

Intel(R) DCM: Energy Director also aggregates the device data to groups. The table below shows the commonly used data types applying to groups of devices.

Data Type	Data
IT equipment power/energy	Aggregated power/energy of IT equipment in a group
Total average power	Aggregated average power of IT devices with average power monitoring capability in a group
Instantaneous power	Aggregated instantaneous power of IT devices with instantaneous power monitoring capability in a group
Estimated power	Aggregated estimated power of IT devices with power estimation capability in a group
PDU power	Aggregated power of PDU devices in a group
Observed maximum PDU power	Maximum value of aggregated PDU power in a period
Maximum/minimum inlet temperature	Maximum/minimum inlet temperature of devices in a group (collected on devices with the corresponding capability only)
Average inlet temperature	Average inlet temperature of the devices with the corresponding capability in a group

Airflow	Total airflow of the devices with the corresponding capability in a group
Average outlet temperature	Average outlet temperature (weighted with airflow) of the devices with the corresponding capability in a group

NOTE

In Intel(R) DCM: Energy Director, power values are in Watts, temperature values are in Celsius, and airflow values are in cubic feet per minute (CFM).

To turn all monitoring on or off, use setCollectionState.

To set a property that applies across Intel(R) DCM: Energy Director, use setGlobalProperty. For a list of properties you can set, see Global Properties.

Intel(R) DCM: Energy Director supports different ways to retrieve the data, including, e.g., dumping the data in raw forms without internal aggregation, retrieving the latest readings, getting data aggregated.

NOTE

If a certain data point is not collected due to a failure (e.g., network connection broken), in the future when the data point is queried, Intel(R) DCM: Energy Director will return an entry of -1 to indicate that the corresponding data point is not available.

NOTE

When a Node Manager server enters any Sx state aside from S0, the server does not provide power or temperature measurements. When Intel(R) DCM: Energy Director does not receive power or temperature measurements, it calls the IPMI command `Get Chassis Status` to check if the node is off. If the node is off, Intel(R) DCM: Energy Director assumes that the server is consuming 30W.

See Also

Aggregating Data

Query/Metrics Interface: Overview

AggregationLevel

QueryType

MetricType

Sampling Frequency

The sampling frequency is the time interval between power and thermal measurements that Intel(R) DCM: Energy Director collects from managed nodes. You can set the sampling frequency in seconds through setGlobalProperty. The default value is 60 seconds.

Valid sampling frequency intervals are: 30, 60, 180, 300, 360, or 600 seconds.

You can set different sampling frequencies for power consumption, and temperature. For example, if you set NODE_THERMAL_SAMPLING_FREQUENCY to 180 seconds and NODE_POWER_SAMPLING_FREQUENCY power to 60 seconds, then Intel(R) DCM collects temperature data on each 180-second interval and collects power data on each 60-second interval.

When you change the sampling frequency, Intel(R) DCM: Energy Director implements the changes from the beginning of the next 30 or 60 minutes interval, depending on the change of measurement granularity.

NOTE

You need to make sure that the device supports the sampling frequency you set, otherwise the device is not monitored. For example, by default, IBM* enclosure/blade only supports monitoring every 600 seconds. In this case, if a mismatching frequency is configured, those devices are not monitored.

HP* Platforms with DCMI Interface

Intel(R) DCM supports managing HP* platforms with DCMI interface exposed. As different DCMI platforms may have different sets of configurable sampling frequencies, Intel(R) DCM: Energy Director monitors the platforms with the most appropriate configuration and processes the measurement data with aggregation. For example, typically the HP* platforms support monitoring frequency of 10 seconds, 5 minutes, or 24 hours. If Intel(R) DCM: Energy Director is configured to use the sampling frequency of 60 seconds, it will monitor the platform every 10 seconds and aggregate the 6 values retrieved in every 60 seconds.

IBM* Enclosures/Blades

By default, IBM* enclosure/blade only supports reporting the power consumption averaged over 600 seconds, and therefore Intel(R) DCM: Energy Director is only able to work with the devices with the sampling frequency of 600 seconds. Intel(R) DCM: Energy Director provides connector configuration files (ibmAMMPluginConfig.xml and ibmBladePluginConfig.xml) to disable the sampling cycle enforcement by changing the line of '<EnforceAveragePeriodCheck value='True' />' to '<EnforceAveragePeriodCheck value='False' />'. After that, Intel(R) DCM: Energy Director monitors the devices according to the sampling frequency specified with any value, treats the power averaged over a specific period of 600 seconds reported from the devices as the power averaged over the previous cycle.

See Also

setGlobalProperty

GlobalProperty

Measurement Granularity

The measurement granularity is the resolution of power/thermal data measurements that are stored in the Intel(R) DCM: Energy Director database for further query/metric usage. You can configure the measurement granularity in seconds through `setGlobalProperty`. The default value is 180 seconds.

Valid measurement data granularity includes 30, 60, 180, 300, 360, 600, 1800, and 3600 seconds, and it must be multiple of power sampling frequency.

You can set different measurement granularity for power consumption, and temperature. For example, if you set `NODE_THERMAL_MEASUREMENT_GRANULARITY` to 180 seconds and `NODE_POWER_MEASUREMENT_GRANULARITY` to 360 seconds, Intel(R) DCM: Energy Director stores temperature data for the granularity of 180 seconds and stores power data for every 360 seconds.

The recommended measurement data granularity is listed in the following table:

Number of Nodes	Recommended Granularity for Managed Nodes	Recommended Granularity for Unmanaged Nodes
No more than 1000	30 seconds	600 seconds
1001 to 5000	180 seconds	

5001 to 10000	360 seconds	

Old measurement data is compressed with a granularity of 1 hour.

See Also

setGlobalProperty

GlobalProperty

Real-time Monitoring

Intel(R) DCM: Energy Director supports monitoring the real-time data of PDUs and UPSes, for example, voltage, load, estimated time remaining on battery. The data is retrieved through real-time communication with the devices initiated by the API call.

See Also

getRealTimePduData

getRealTimeUpsData

Power Estimation

For nodes without power monitoring capabilities, Intel(R) DCM: Energy Director provides the mechanism for management software to attach power estimators to the nodes. With these estimators, management software is able to explicitly input the power estimation values, or implicitly provide the related information for a simple estimation on the server power consumption. The estimated power values are stored in Intel(R) DCM: Energy Director. Intel(R) DCM: Energy Director provides a unified data repository. Estimated power values can be queried through the query type `ESTIMATED_PWR`.

Intel(R) DCM: Energy Director provides the function of estimating power consumption dynamically based on the utilization data retrieved from its operating system. Currently the function applies to Microsoft* Windows Servers with WMI exposed, different Linux servers, and some virtual machine managers.

Intel(R) DCM: Energy Director also provides the function of estimating power consumption dynamically with advanced power model trained from the historical utilization and power data.

Leveraging the power estimation mechanism, Intel(R) DCM: Energy Director allows associating PDU outlets to a server (or an enclosure). In this case, the power monitored from the PDU outlets is regarded as the power estimated for the server (or the enclosure).

Leveraging the power estimation mechanism, for certain enclosures/chassis Intel(R) DCM: Energy Director supports estimating enclosure/chassis power by referring to the PSU power monitored from the nodes inside the enclosures/chassis.

NOTE

Power estimators can only be attached to devices without power monitoring capabilities. If a device is attached with a power estimator before its power monitoring capability is discovered, once the capability is discovered, the power estimator is detached.

See Also

getQueryData

getMetricData

Querying Aggregated Data

Intel(R) DCM: Energy Director enables you to aggregate data from multiple nodes or groups, or across different time periods. To aggregate this data, use the Query/Metrics Interface.

Aggregation Level

When you call getQueryData and getMetricData, you can set the level to which returned data is aggregated. You can aggregate data in one of three ways:

- No aggregation: The method call returns data for each node separately.
- Self: Data is aggregated to the level of the entity for which you are getting the data. For example, if you request data for a group, it aggregates all the data to the level of that group.
- Aggregation to level of physical group: Data is aggregated to the level of a physical group. The physical group can be a rack, row, or room. Note that the correct power data for enclosures are supposed to be provided directly from the enclosure devices. To aggregate the power data from the individual blade servers under the enclosure to the enclosure without power monitoring capability, one may need to specify the entity property `AGGREGATION_MULTIPLIER`. For logical group power data, the aggregation is

performed on the device/node level, depending on how the logical group is organized to contain the enclosures and blades.

Queries
You can use getQueryData to get the minimum, maximum or average temperature or power usage of nodes or groups. For complete descriptions of the different types of queries, see `QueryType`.

Metrics
You can use getMetricData to get a variety of metrics relating to temperature or power usage of nodes or groups. For complete descriptions of the different types of metrics, see MetricType. Values returned for metrics are based on historical averages.

NOTE

During data aggregation, if all the data to be aggregated is not available, Intel(R) DCM: Energy Director will give an aggregated result of -1. If some of the data is available while some is not, Intel(R) DCM: Energy Director will try to perform an aggregation with a certain approximation method for those unavailable data.

See Also

AggregationLevel

QueryType

MetricType

getQueryData

getMetricData

Data Statistics and Analysis

Intel(R) DCM: Energy Director provides data statistics and analysis capabilities.

For example, Intel(R) DCM: Energy Director supports retrieving the historical IT equipment power distribution of physical entities. Calling `startCollectingPowerDistribution` starts collecting historical power statistics on an entity. To retrieve the statistics, call `getPowerDistribution`. Call `stopCollectingPowerDistribution` to stop the collection process with the data wiped.

NOTE

For scalability considerations, it is recommended to keep the number of entities below 500 for simultaneous collection of historical power statistics.

One may retrieve the current inlet temperature distribution of a physical group by calling `getInletTemperatureDistribution`.

There are some common ways to get certain indicators on cooling of a server room. One may evaluate how well a group of servers are cooled against over-heating (with respect to a certain recommended temperature and a certain allowable temperature) by calling getCoolingIndicator, and evaluate how well a group servers are over-cooled (with respect to a certain recommended temperature and a certain allowable temperature) by calling `getOverCoolingIndicator`. One may also get the temperature rise across a group of servers with outlet temperature and airflow monitoring capability by calling `getTemperatureRise`, which can be combined with the temperature difference between the outlet and the inlet of the air conditioners to determine whether hot air recirculation or cold air by-passing exists.

Intel(R) DCM: Energy Director also provides the building blocks for advanced usages in data analysis, with heuristics and experts' knowledge.

In some datacenters, there are servers which are under-utilized for most of the time. Typically the workload on the server does not deserve a dedicated physical server and can be consolidated. One may evaluate whether a server is an under-utilized one by calling `evaluateLowUtilizationServer`.

With the visibility of server inlet temperature, datacenter cooling becomes an important area for optimization. One may evaluate the current cooling status of a room by calling `analyzeCoolingStatus`, in which the current status evaluation, suggestions for optimizations, and benefits after following the suggestions are given. One may also specify the upper limit of the inlet temperature for the room according to different cooling criteria given by American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (ASHRAE).

Intel(R) DCM: Energy Director 3.x provides the new function of building advanced power models taking utilization data as the input to predict power consumption. One may use Intel(R) DCM: Energy Director to dump both the utilization data and the power data for servers with power monitoring capability. After feeding the data dumped to built a power model, the model can be used to perform what-if power analysis, that is, to predict the power consumption based on the hypothesized values of resource utilization. Use the API `addPowerModel` to create a new power model, in which utilization vectors and power data measured are fed. Use the API `predictPower` to predict the power consumption taking a utilization vector (with the

same length as that of the vector in creating power model) as the input. The APIs, `enumeratePowerModels` and `removePowerModel`, are provided to manage the power models created.

Database Schema: Measurement Data

Measurement Data: Overview

The measurement data schema enables you to pull Intel(R) DCM: Energy Director measurement data to meet your system requirements, for example, draw a trending graph, or analyze data.

Intel(R) DCM: Energy Director measurement data schema stores the data of the monitored entity, and establishes the relationship between the tables of measurement data and the entities with *entity_id*.

For performance and database size consideration, Intel(R) DCM: Energy Director separates the monitoring data into two categories: Power and Thermal. Intel(R) DCM: Energy Director also compresses power and thermal monitoring data into one-hour and one-day compressed data.

In addition, Intel(R) DCM: Energy Director separates the physical entity and group entity measurement data for easy querying and better performance. If you need measurement data for a group, you can query the corresponding tables to get the data.

These are the measurement data tables described in this section.

Table Name	Description
T_Entity	Store managed entities, including group entities and physical entities.
T_Power_Raw, T_Power_1Hour, & T_Power_24Hour	Store raw power data, 1-hour compressed power data, and 24-hour compressed power data for devices with average power monitoring capability
T_Ins_Power_Raw, T_Ins_Power_1Hour, & T_Ins_Power_24Hour	Store raw power data, 1-hour compressed power data, and 24-hour compressed power data for devices with instantaneous power monitoring

	capability
T_Estimation_Raw, T_Estimation_1Hour, & T_Estimation_24Hour	Store raw power data, 1-hour compressed power data, and 24-hour compressed power data for devices with power estimation capability
T_IT_Equipment_Power_Raw, T_IT_Equipment_Power_1Hour, & T_IT_Equipment_Power_24Hour	Store raw IT equipment power data, 1-hour compressed IT equipment power data, and 24-hour compressed IT equipment power data for devices with different power monitoring capabilities
T_Group_Power_Raw & T_Group_Power_1Hour	Store group level average power data in the raw data granularity, 1-hour granularity, and 24-hour granularity
T_Group_Ins_Power_Raw, T_Group_Ins_Power_1Hour, & T_Group_Ins_Power_24Hour	Store group level instantaneous power data in the raw data granularity, 1-hour granularity, and 24-hour granularity
T_Group_Estimation_Raw, T_Group_Estimation_1Hour, & T_Group_Estimation_24Hour	Store group level power estimation data in the raw data granularity, 1-hour granularity, and 24-hour granularity
T_Group_IT_Equipment_Power_Raw, T_Group_IT_Equipment_Power_1Hour, & T_Group_IT_Equipment_Power_24Hour	Store group level IT equipment power data in the raw data granularity, 1-hour granularity, and 24-hour granularity
T_Thermal_Raw, T_Thermal_1Hour, & T_Thermal_24Hour	Store raw inlet temperature data, 1-hour compressed inlet temperature data, and 24-hour compressed inlet temperature data for devices with inlet temperature monitoring

	capability
T_Group_Thermal_Raw & T_Group_Thermal_1Hour	Store average inlet temperature data and 1-hour compressed average inlet temperature data for group entities.

Table: T_Entity

Table *T_Entity* stores all the entities including group and physical entities in Intel(R) DCM: Energy Director. This table provides the entity type and entity ID. This information is important for collecting the measurement data.

Table properties:

T_Entity		
PK	**entity_id**	**int identity**
	entity_type creation_date is_active	varchar(20) datetime char(1)

Column properties:

Column Name	Data Type	PK	Notes
entity_id	int(auto increment)	Y	Primary key of table *T_Entity*
entity_type	varchar(20)		Description for entity type. See EntityType.
creation_date	datetime		Entity created time
is_active	char(1)		*Y* or *N* value is acceptable for this column. For active entity (*Y*), it means the entity can be managed by Intel(R) DCM: Energy Director. For inactive entity (*N*), it means the entity is deleted. Intel(R) DCM: Energy Director will not manage it, but only keep the data

			in database for query purpose.

Table: T_Power_Raw, T_Power_1Hour, and T_Power_24Hour
The tables are used to store raw power data, one-hour compressed power data, and 24-hour compressed power data for devices with average power monitoring capability.

NOTE

- The power raw data granularity is the same as NODE_POWER_MEASUREMENT_GRANULARITY, but NODE_POWER_MEASUREMENT_GRANULARITY may change at Intel(R) DCM: Energy Director runtime, so this table stores current granularity into the column *measurement_freq*.
- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_Power_Raw:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of table *T_Power_Raw*
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.

measurement_date	datetime, not NULL		Measurement data created time.
avg_value	bigint, not Null		The average power consumption in the past time slot.
min_value	bigint		The minimum power consumption in the past time slot.
max_value	bigint		The maximum power consumption in the past time slot.
estimated_avg_value	bigint		A value for approximation in the past time slot when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Column Properties of Table T_Power_1Hour and Table T_Power_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of table *T_Power_1Hour/ T_Power_24Hour*
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not Null		Measurement data created time.
avg_value	bigint, not Null		The average power consumption in the past

			hour/day.
min_value	bigint		The minimum power consumption in the past hour/day.
max_value	bigint		The maximum power consumption in the past hour/day.
estimated_avg_value	bigint		A value for approximation in the past hour/day when the data is missing.

Table: T_Ins_Power_Raw, T_Ins_Power_1Hour, and T_Ins_Power_24Hour

The tables are used to store raw power data, one-hour compressed power data, and 24-hour compressed power data for devices with instantaneous power monitoring capability.

NOTE

- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_Ins_Power_Raw, T_Ins_Power_1Hour and T_Ins_Power_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.

measurement_date	datetime, not NULL		Measurement data created time.
ins_value	bigint, not Null		The instantaneous power consumption in the past slot/hour/day
estimated_ins_value	bigint		A value for approximation in the past slot/hour/day when the data is missing.
avg_ins_value	bigint		The average instantaneous power consumption in the past slot/hour/day which can be used for average IT equipment calculation
ins_pdu_value	bigint		The instantaneous power consumption of PDU devices in the past slot/hour/day
estimated_ins_pdu_value	bigint		A value for approximation in the past hour/day when the data is missing.
max_ins_pdu_value	bigint		The maximum instantaneous power consumption of PDU devices in the past slot/hour/day
estimated_max_ins_pdu_value	bigint		A value for approximation in the past hour/day when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Table: T_Estimation_Raw, T_Estimation_1Hour, and T_Estimation_24Hour

The tables are used to store raw power data, one-hour compressed power data, and 24-hour compressed power data for devices with power estimation capability.

NOTE

- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_Estimation_Raw, T_Estimation_1Hour and T_Estimation_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
avg_value	bigint, not Null		The estimated power consumption in the past slot/hour/day
estimated_avg_value	bigint		A value for approximation in the past slot/hour/day when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Table: T_IT_Equipment_Power_Raw, T_IT_Equipment_Power_1Hour, and T__IT_Equipment_Power_24Hour

The tables are used to store raw IT equipment power data, one-hour compressed IT equipment power data, and 24-hour compressed IT equipment power data for devices with different power monitoring capabilities.

NOTE

- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_IT_Equipment_Power_Raw, T_IT_Equipment_Power_1Hour and T_IT_Equipment_Power_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
it_equipment_power	bigint, not Null		The IT equipment power in the past slot/hour/day
estimated_it_equipment_power	bigint		A value for approximation in the past slot/hour/day when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Table: T_Group_Power_Raw and T_Group_Power_1Hour
The tables are used to store group level average power data in the raw data granularity and 1-hour granularity.

NOTE

- The group power raw data granularity is the same as NODE_POWER_MEASUREMENT_GRANULARITY, but NODE_POWER_MEASUREMENT_GRANULARITY may change at Intel(R) DCM: Energy Director runtime, so this table stores current granularity into the column *measurement_freq*.
- The power raw data is kept for the amount of time set in TIME_UNTIL_GROUP_DATA_COMPRESSION. Default value: 7 days. When this time passes, the power raw data is deleted from the disk.

Column Properties of Table T_Group_Power_Raw:

Column Name	Data Type	PK	Notes
measurement_id	int (auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director system maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
group_total_avg	bigint		TOTAL_AVG_PWR of group in the past time slot.
estimated_group_total_avg	bigint		A value for approximation in the past time slot when the data is missing.
group_total_min	bigint		TOTAL_MIN_PWR of group in the past time slot.

group_total_max	bigint		TOTAL_MAX_PWR of group in the past time slot.
group_min_avg	bigint		MIN_AVG_PWR of group in the past time slot.
group_max_avg	bigint		MAX_AVG_PWR of group in the past time slot.
estimated_group_max_avg	bigint		A value for approximation in the past time slot when the data is missing.
group_derated_power	bigint		The sum of de-rated power of unmanaged nodes in the group. Unmanaged nodes are nodes that are not managed by Intel(R) DCM: Energy Director
avg_value	bigint		The average power consumption of all the nodes/enclosures in the group in the past time slot.
min_value	bigint		The minimum power consumption of the nodes/enclosures in the group in the past time slot.
max_value	bigint		The maximum power consumption of the nodes/enclosures in the group in the past time slot.

group_total_avg_cap	bigint		TOTAL_AVG_PWR_CAP of the group in the past time slot.
estimated_group_total_avg_cap	bigint		A value for approximation in the past time slot when the data is missing.
group_total_max_cap	bigint		TOTAL_MAX_PWR_CAP of the group in the past time slot.
group_max_avg_cap	bigint		MAX_AVG_PWR_CAP of the group in the past time slot.
estimated_group_max_avg_cap	bigint		A value for approximation in the past time slot when the data is missing.

Column Properties of Table T_Group_Power_1Hour:

Column Name	Data Type	PK	Notes
measurement_id	int (auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director system maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
group_total_avg	bigint		TOTAL_AVG_PWR of group in the past hour.

estimated_group_total_avg	bigint		A value for approximation in the past hour when the data is missing.
group_total_min	bigint		TOTAL_MIN_PWR of group in the past hour.
group_total_max	bigint		TOTAL_MAX_PWR of group in the past hour.
group_min_avg	bigint		MIN_AVG_PWR of group in the past hour.
group_max_avg	bigint		MAX_AVG_PWR of group in the past hour.
estimated_group_max_avg	bigint		A value for approximation in the past hour when the data is missing.
group_derated_power	bigint		The sum of de-rated power of unmanaged nodes in the group. Unmanaged nodes are nodes that are not managed by Intel(R) DCM: Energy Director
avg_value	bigint		The average power consumption of all the nodes/enclosures in the group in the past hour.
min_value	bigint		The minimum power consumption of the nodes/enclosures in the group in the past hour.
max_value	bigint		The maximum power consumption of the

			nodes/enclosures in the group in the past hour.
measurement_freq	int		The past measurement time slot, in seconds.
group_total_avg_cap	bigint		TOTAL_AVG_PWR_CAP of the group in the past hour.
estimated_group_total_avg_cap	bigint		A value for approximation in the past hour when the data is missing.
group_total_max_cap	bigint		TOTAL_MAX_PWR_CAP of the group in the past hour.
group_max_avg_cap	bigint		MAX_AVG_PWR_CAP of the group in the past hour.
estimated_group_max_avg_cap	bigint		A value for approximation in the past hour when the data is missing.

Table: T_Group_Ins_Power_Raw, T_Group_Ins_Power_1Hour, and T_Group_Ins_Power_24Hour

The tables are used to store group level instantaneous power data in the raw data granularity, 1-hour granularity and 24-hour granularity.

NOTE

- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_Group_Ins_Power_Raw, T_Group_Ins_Power_Raw_1Hour, and T_Group_Ins_Power_Raw_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto	Y	Primary key of tables

	increment)		
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
ins_value	bigint, not Null		The instantaneous power consumption in the past slot/hour/day
estimated_ins_value	bigint		A value for approximation in the past slot/hour/day when the data is missing.
avg_ins_value	bigint		The average instantaneous power consumption in the past slot/hour/day which can be used for average IT equipment calculation
ins_pdu_value	bigint		The instantaneous power consumption of PDU devices of the group in the past slot/hour/day
estimated_ins_pdu_value	bigint		A value for approximation in the past slot/hour/day when the data is missing.
max_ins_pdu_value	bigint		The maximum instantaneous power consumption of PDU

			devices of the group in the past slot/hour/day
estimated_max_ins_pdu_value	bigint		A value for approximation in the past slot/hour/day when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Table: T_Group_Estimation_Raw, T_Group_Estimation_1Hour, and T_Group_Estimation_24Hour

The tables are used to store group level power estimation data in the raw data granularity, 1-hour granularity and 24-hour granularity.

NOTE

- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_Group_Estimation_Raw, T_Group_Estimation_1Hour, and T_Group_Estimation_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
avg_value	bigint, not Null		The estimated power consumption in the past

			slot/hour/day
estimated_avg_value	bigint		A value for approximation in the past slot/hour/day when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Table: T_Group_IT_Equipment_Power_Raw, T_Group_IT_Equipment_Power_1Hour, and T_Group_IT_Equipment_Power_24Hour

The tables are used to store group level IT equipment power data in the raw data granularity, 1-hour granularity and 24-hour granularity.

NOTE

- The power raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the raw data is deleted from the disk.

Column Properties of the Table T_Group_IT_Equipment_Power_Raw, T_Group_IT_Equipment_Power_1Hour, and T_Group_IT_Equipment_Power_24Hour:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of the table
entity_id	int, not NULL		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not NULL		Measurement data created time.
it_equipment_power	bigint, not		The IT equipment power

	Null		data in the past slot/hour/day
estimated_it_equipment_power	bigint		A value for approximation in the past slot/hour/day when the data is missing.
measurement_freq	int		The frequency of data measurement, in seconds.

Table: T_Thermal_Raw, T_Thermal_1Hour, T_Thermal_24Hour
The tables are used to store raw inlet temperature data, one-hour compressed inlet temperature data, and 24-hour compressed inlet temperature data for devices with inlet temperature monitoring capability

NOTE

- The thermal raw data granularity is the same as NODE_THERMAL_MEASUREMENT_GRANULARITY, but NODE_THERMAL_MEASUREMENT_GRANULARITY may change at Intel(R) DCM: Energy Director runtime, so this table stores current granularity into the column *measurement_freq*.
- The thermal raw data is kept for the amount of time set in TIME_UNTIL_NODE_DATA_COMPRESSION. Default value: 7 days. When this time passes, the thermal raw data is deleted from the disk.

Column Properties of the Table T_Thermal_Raw:

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of table *T_Thermal_Raw.*
entity_id	int, not null		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not		Measurement data created time.

	null		
avg_value	bigint, not null		The average inlet temperature in the past time slot.
min_value	bigint		The minimum inlet temperature in the past time slot.
max_value	bigint		The maximum inlet tmperature in the past time slot.
measurement_freq	int		The past measurement time slot, in seconds.

Column Properties of the Table T_Thermal_1Hour and T_Thermal_24Hour :

Column Name	Data Type	PK	Notes
measurement_id	int(auto increment)	Y	Primary key of the table.
entity_id	int, not null		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director maintains the data consistency.
measurement_date	datetime, not null		Measurement data created time.
avg_value	bigint, not null		The average inlet temperature in the past time slot.
min_value	bigint		The minimum inlet temperature in the past time slot.
max_value	bigint		The maximum inlet tmperature in the past time slot.

Table: T_Group_Thermal_Raw and T_Group_Thermal_1Hour
The tables are used to store average inlet temperature data and one-hour compressed average inlet temperature data for group entities.

NOTE

- The group thermal data granularity is the same as NODE_THERMAL_MEASUREMENT_GRANULARITY, but `NODE_THERMAL_MEASUREMENT_GRANULARITY` may change at Intel(R) DCM: Energy Director runtime, so this table stores current granularity into the column *measurement_freq*.
- The group thermal data is kept for the amount of time set in TIME_UNTIL_GROUP_DATA_COMPRESSION. Default value: 7 days. When this time passes, the group thermal data is deleted from the disk.

Column Properties of the Table T_Group_Thermal_Raw:

Column Name	Data Type	PK	Notes
measurement_id	int (auto increment)	Y	Primary key of table *T_Group_Thermal_Raw*.
entity_id	int, not null		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director system maintains the data consistency.
measurement_date	datetime, not null		Measurement data created time.
avg_value	bigint		The average inlet temperature of all nodes in the group in the past time slot.
min_value	bigint		The minimum inlet temperature of the nodes in the group in the past time slot.
max_value	bigint		The maximum inlet temperature of the nodes in the group in the past time slot.
measurement_freq	int		The past measurement time

			slot, in seconds.

Column Properties of the Table T_Group_Thermal_1Hour:

Column Name	Data Type	PK	Notes
measurement_id	int (auto increment)	Y	Primary key of table *T_Group_Thermal_1Hour*.
entity_id	int, not null		See table *T_Entity*. ***Note:*** No mandatory UPDATE CASCADE and DELETE CASCADE. Intel(R) DCM: Energy Director system maintains the data consistency.
measurement_date	datetime, not null		Measurement data created time.
avg_value	bigint		The average inlet temperature of all nodes in the group in the past hour.
min_value	bigint		The minimum inlet temperature of the nodes in the group in the past hour.
max_value	bigint		The maximum inlet temperature of the nodes in the group in the past time slot.

Using Control Policies

Control Policies: Overview

You can use control policies to limit the amount of power that a group or node consumes. Each policy applies to one entity, either a group or a node.

Intel(R) DCM: Energy Director provides several policy types:

- `CUSTOM_PWR_LIMIT` limits the total power consumption of an entity. When the policy applies to a group, Intel(R) DCM: Energy Director actively reallocates the power budgets to the individual servers within the group in each monitoring cycle. It attempts to minimize the gap between power demands of entities and power allocation, in order to minimize the performance impact of the group power capping. Intel(R) DCM: Energy Director monitors the power consumption data of the servers, estimates the power demand of the servers, and reallocates the power budgets with a sophisticated approach by applying a heuristic discriminative approach to solve a probabilistic model. In general, Intel(R) DCM: Energy Director reacts quickly by allocating more power budgets to servers to get new tasks running properly. If the total power demand of the group exceeds the group power constraint, Intel(R) DCM: Energy Director gives a balanced power allocation. The policies are commonly applied to increase the server density with respect to power or cooling capacity.
- `STATIC_PWR_LIMIT` forces a node to be capped with a fixed power limit. Use this policy to explicitly dictate the power budget allocation or the power throttling level when the appropriate power budgets of the nodes are known. The policy type is also recommended for static power budget allocation on HP enclosures for rack density increase even if the enclosures are included in rack/room policies with the type of `CUSTOM_PWR_LIMIT`.
- `PWR_EFFICIENT` forces the server to its most power efficient state, that is the P-state with the lowest performance and power without clock throttling. The effect may be overwritten by an extreme low power budget allocated from other policies or `MIN_PWR`.
- `MIN_PWR` throttles power consumption of an entity as much as possible. Use this policy to prolong business continuity in the case of an emergency.

- `MIN_PWR_ON_INLET_TEMP_TRIGGER` throttles the power consumption of the entity as much as possible when the average inlet temperature exceeds a threshold. Use this policy to prolong business continuity in the case of datacenter cooling system failure.
- `CPU_PWR_LIMIT` throttles the power consumption of the server's CPU sub-system with a certain limit. This policy type only applies to Intel(R) Node Manager 2.0 platforms and does not apply to group entities.
- `MEM_PWR_LIMIT` throttles the power consumption of the server's memory sub-system with a certain limit. This policy type only applies to Intel(R) Node Manager 2.0 platforms and does not apply to group entities.

Multiple policies can simultaneously act on different entities. Multiple policies of different types can simultaneously act on a single entity.

Intel(R) DCM: Energy Director attempts to maintain all active policies simultaneously. If Intel(R) DCM: Energy Director cannot enforce a policy for three consecutive monitoring cycles, it sends the event `CONTROL_POLICY_CANNOT_BE_MAINTAINED`. After receiving this event, you can mediate the problem. For example, you can release the power target or move workloads to another server group.

See Also

Policy Priority Levels

PolicyType

setPolicyEx

Control Policy Example

This example illustrates how to use Intel(R) DCM: Energy Director power policies to increase rack density and balance power across servers based on workload demand. This example used Urbanna systems with an Intel® Microarchitecture Codename Nehalem processor, running a Red Hat Enterprise Linux* 5.2 operating system running SPECpower_ssj2008 1.01.

In this example, a rack that can support up to 3000 W, hosts four servers, each with a nameplate power supply of 650 W. Those four servers are divided into two groups of two servers each, as follows:

- Group A runs a customer facing workload with high service-level agreement. The workload has 50% system utilization between 5PM-9AM and 100% utilization between 9AM-5PM.
- Group B runs an internal IT batch job, with an average of 50% utilization throughout the day.

The IT administrator wants to add more servers to this rack. Due to the high service level agreement on Group A, the IT administrator wants minimal impact on Group A performance, but is not very concerned with performance impact on Group B.

The following is an example of how an IT administrator can use Intel(R) DCM: Energy Director to add more servers to the rack:

1. Use the Trend view to track power consumption of the rack over several days. In this example, the rack consumes 1150W at peak hours (9AM-5PM) and 980W at other times (5PM-9AM).
2. Apply a rack-level power policy of 1100W, with Group A set to High priority and Group B set to Low priority. This policy is effective over the whole day.

Intel(R) DCM maintains the 1100W rack power policy. At peak times, Intel(R) DCM: Energy Director: Energy Director senses that Group A needs more power, and moves some power from Group B to Group A. In this example, power for Group A increased from 520W off-peak to 676W peak, and power for Group B decreased from 480W off-peak to 424W peak. There was no observed performance impact to the Group A workload.

The IT administrator can now provision the rack based on the assumption that these four servers will use 1100W. Previously, the IT administrator had assumed that the servers each used 650W, for 2600W total. The reduced power provisioning enables hosting more servers on the rack.

See Also

Control Policies Interface: Overview

Policy Priority Levels

Control Policies and Datacenter Hierarchy

A node can be included in more than one group. When you set a power policy on a group, the total power supplied to that group cannot be more than the amount defined in the policy. If a node is included in multiple groups with different policies,

the policies of each group may affect the other groups. If multiple policies apply to one node, Intel(R) DCM: Energy Director implements the most restrictive power policy for the node. The following is an example:

Group A includes Node1, Node2, and Node3. Group A has a `CUSTOM_PWR_LIMIT` of 225W.

Group B includes Node3 and Node4. Group B has a `CUSTOM_PWR_LIMIT` of 300W.

Each node in Group A receives 75W.

For Group B, the total power supplied is 300w. Since Group A limits Node3 to 75W, Node4 receives 225W.

If the power policy for Group A changes, then the implementation of the power policy for Group B could change also, even if the Group B policy itself does not change.

See Also

Control Policies Interface: Overview

Setting the Datacenter Hierarchy

Policies on Unavailable Nodes

When a policy applies to a node that is not connected to Intel(R) DCM: Energy Director, the policy assumes the maximum realistic power requirement for that node. It calculates the maximum realistic power requirement as follows:

1. Intel(R) DCM: Energy Director uses the latest power cap value, if available.
2. If the historical power limit is not available, Intel(R) DCM: Energy Director uses the de-rated power (DERATED_PWR) value for the node.

See Also

Control Policies Interface: Overview

Setting the Datacenter Hierarchy

Policy Priority Levels

When you set or update a policy, you can select different priority levels for each node. For example, you can set priority levels based on the service level agreements associated with workloads running on different nodes. If there is performance loss as a result of power throttling on a server with a higher priority, Intel(R) DCM: Energy Director regards it as more important than the same case on a server with a lower priority, and tends to reserve more budgets for servers with higher priorities.

For each node, you can choose one of the priority levels:

- Low
- Medium (Default)
- High
- Critical: Intel(R) DCM: Energy Director reserves power, defines as the de-rated power to the node.

Though priority lists are policy-specific and a node may have different priorities in different policies, during policy calculation, a higher priority of a node in one policy may override a lower priority set to the same node in another policy.

NOTE

Policy priority levels currently apply only to the `CUSTOM_PWR_LIMIT` policy type.

See Also

Control Policies Interface: Overview

Control Policies and Datacenter Hierarchy

PolicyType

setPolicyEx

updatePolicyEx

Policy Modes

Three separate factors determine the behavior of a policy:

Factor	Description
Enabled/Disabled	If enabled, the policy is available for use. Use setPolicyState to enable or disable the policy.
Scheduling	Use schedulePolicy to schedule when a policy applies. When a policy is both Enabled and scheduled to apply, the policy becomes active. An active policy can be triggered or not triggered.
Triggered	When the condition for an active policy is met, that policy is triggered. If a policy has no condition, it is always triggered. When a policy is triggered, it enforces power or thermal constraints. You set the policy condition through

	setPolicyEx or updatePolicyEx.

NOTE

To change any policy setting, the policy must be Disabled.

See Also

Control Policies Interface: Overview

schedulePolicy

setPolicyState

setPolicyEx

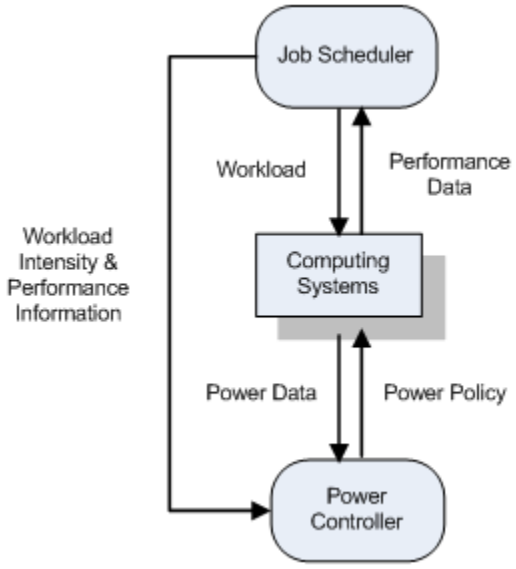
updatePolicyEx

Building Blocks for Performance-Aware Power Saving

Building Blocks for Performance-Aware Power Saving

Intel(R) DCM: Energy Director provides performance-aware power saving capability, application level performance indicators is measured by power managers/controllers to further exploit the room for power reduction.

The following figure describes the high-level system diagram of a typical setting of performance-aware power saving. The job scheduler is responsible for scheduling workloads to computing systems and acts as the performance manager to monitor the application level performance. The information is provided to the power controller which computes the intelligent power policy. The power policy is then executed to reduce the power consumption of the computing systems.



Intel(R) DCM: Energy Director provides the building block of performance-aware power saving as an experimental feature. For performance-aware power saving to work, you must have performance expectation of the workload clearly defined. Furthermore, an ordered set of the workload intensity levels exists: similar workload intensities imply similar power demands to complete the job with a certain performance constraint, and larger workload intensity implies a larger power demand with certain tolerable noises.

To use the feature, the job scheduler shall first call `startPowerSaving` to enable it on the server.

After that, it is expected to hint Intel(R) DCM: Energy Director with the current workload intensity by calling `updateWorkloadIntensity`. The workload intensity value shall range from 0.0 to 1.0, indicating the relative intensity of the workload. Based on the hint of the current workload intensity and the past experiences, Intel(R) DCM: Energy Director configures the optimal power limit (ranges from a sufficiently low power value to the de-rated power of the server) for the server.

Meanwhile, the job scheduler is expected to inform Intel(R) DCM: Energy Director whether your performance expectation is maintained or not by calling `updatePerformanceFeedback`. According to the performance feedback, Intel(R) DCM: Energy Director updates the past experiences of power saving, re-calculates the optimal power limit, and possibly re-adjusts the power limit on the server.

To disable the feature, call `stopPowerSaving`.

NOTE

Currently the feature is not compatible with other power policies. If both power saving and other power policies are enabled on the same entity, the behaviors are undefined.

Case Study: Power Saving on TPCC-UVa

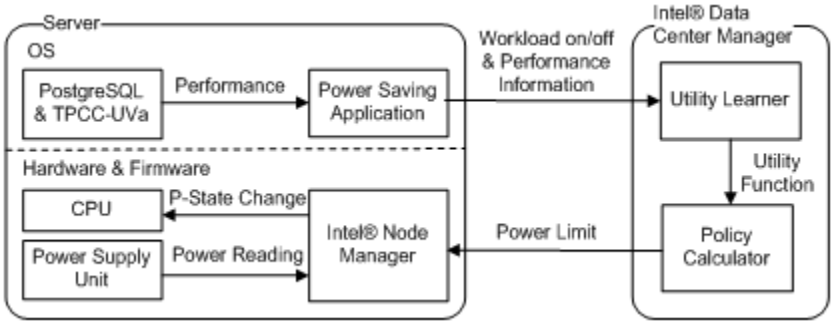
This case study shows how power saving works on TPCC-UVa.

TPC-C: An on-line transaction processing benchmark, involving multiple transaction types, a complex database and execution structure. Refer to: http://www.tpc.org/tpcc/.

tpmc: The execution of the benchmark produces a performance parameter based on throughput, it is called "TPC-C transactions per minute" (tpmC).

TPCC-UVa: An open source implementation of TPC-C written in C language using the PostgreSQL database system and a simple transaction monitor. Refer to: http://www.infor.uva.es/~diego/tpcc-uva.html.

In this case study, performance-aware power saving is adapted to the benchmark of TPCC-UVa. The following figure shows the system diagram of performance-aware power saving adapted to the benchmark.



There are several components in the system:

- **TPCC-UVa:** It is instrumented by Intel(R) Energy Checker (EC), an SDK to measure both software productivity and hardware platform energy consumption to facilitate reporting energy efficiency metrics. For more details of Intel(R) EC, refer to: http://software.intel.com/en-us/articles/intel-energy-checker-sdk/.
- **Power Saving Application:** It performs as the centralized coordinator. It reads the performance information from the benchmark through Intel(R) EC's instrumentation, provides the information whether the workload is running, and whether the performance impacts Intel(R) DCM: Energy Director.
- **Intel(R) DCM: Energy Director**: It is a power management middleware which provides the building block for performance-aware power saving.
- **Utility Learner:** It is located inside Intel(R) DCM. It collects the performance feedback, builds the utility functions on how the power saving task is performed, considering both the power reduction and the performance impact.
- **Policy Calculator:** It is located inside Intel(R) DCM. It uses utility functions to generate the best power policy which is a power limit on the server. The power limit is enforced to Intel(R) Node Manager
- **Intel(R) Node Manager:** A firmware component, it responsible for capping the server's power below the power limit, with a control loop of retrieving

power readings from the power supply unit and regulating the CPU P-state. The process is repeated over time.

Both TPCC-UVa and PostgreSQL are instrumented by Intel(R) EC. The following figures compare the differences of the overall architecture with/without instrumentation. Intel(R) EC collects the front-end performance information from the remote terminal emulator, as well as the back-end performance information from the transaction monitor of the database. The performance information is logged to a predefined location.

Figure: PostgreSQL and TPCC-UVa Overall Architecture without Instrumentation:

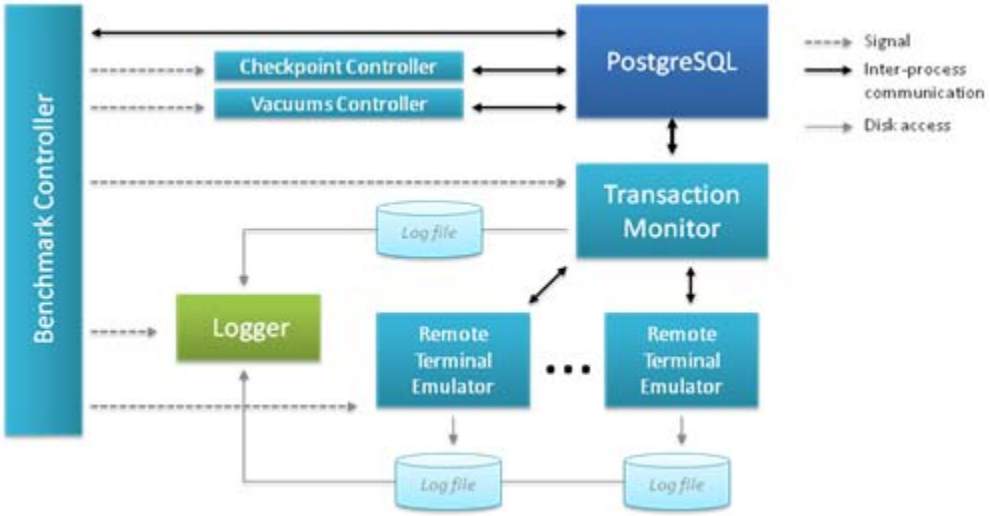
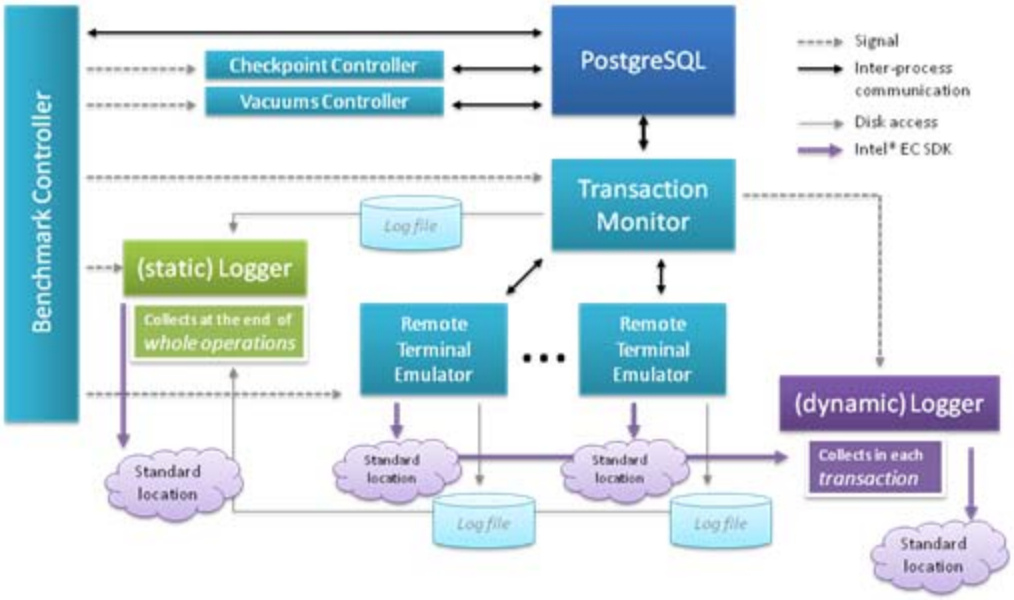


Figure: PostgreSQL and TPCC-UVa Overall Architecture with Instrumentation:



The power saving application acts as a coordinator bridging TPCC-UVa benchmark with the power saving building block in Intel(R) DCM. It provides the 0-1 workload intensity information to Intel(R) DCM: Energy Director indicating whether the benchmark is running. The application also reads the performance counters output by Intel(R) EC in TPCC-UVa and judges whether the performance gets impacted. The performance feedback is provided to Intel(R) DCM: Energy Director for updating the utility of power capping decisions.

To judge whether the performance gets impacted, the application first checks the front-end counters of each client terminals periodically. The number of transactions processed and the number of well-done transactions for every five transaction types within the period are collected. The data is aggregated across all the client terminals and the percentage that the transactions are processed as well-done ones is calculated. By comparing the percentage values of each of the five transaction types with a predefined threshold, the application judges whether the client side experiences get impacted.

The application also monitors the performance counter of the back-end database. The number of transactions completed in the recent five seconds is sampled periodically. Before starting power saving, a preliminary run of the benchmark without power saving kicked in is performed, collecting a set of samples. During the run with power saving, the application collects a few samples within an interval and compares the running samples with the original sample set. Welch's t-test is performed to test whether the difference of the mean number of transactions completed from the two sample sets are statistically significant. Specifically, it computes the t value as:

$$\frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

where $\overline{X}_i$, s_i^2, and N_i are the i^{th} sample mean, sample variance, and sample size. Welch-Satterthwaite Equation is used to approximate the degrees of freedom associated with the variance as:

$$\upsilon \approx \frac{(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2})^2}{\frac{s_1^4}{N_1^2(N_1 - 1)} + \frac{s_2^4}{N_2^2(N_2 - 1)}}$$

Once t and υ are computed, the values are used with the t-distribution to test whether the difference of the means is statistically significant in terms of comparing the p-value with a predefined significance level.

For the power saving application to provide the feedback that the performance is not impacted, it requires both the front-end performance and the back-end performance to pass the check.

Experiments were conducted for a comparative study on TPCC-UVa benchmark, investigating the effects of performance-aware power saving versus power saving with the power management module in operating system.

The system running the benchmark in the experiment consists of two Intel(R) Xeon® X5570 CPUs, 12GB DDR3 1066MHz memory, and a hard disk of 320GB with 7200RPM. Each of the CPU has four cores and hyper-threading is enabled on them. The operating system of the server is Red Hat Enterprise 5.2. Throughout the experiment, the on-demand governor in the operating system is enabled, it provides the functionality of P-state regulation based on CPU utilization. This configuration is regarded as the baseline.

TPCC-UVa benchmark is set up on the server. Before running the experiments, Different configurations of the benchmark are tried as a preliminary step to maximize the throughput, reaching the settings of 24 warehouses and 10 terminals per warehouse. In the experiments, the ramp up period of TPCC-UVa is 10 minutes and the measurement period is 4 hours.

In performance-aware power saving with a coordinated approach, the power saving application and the TPCC-UVa benchmark run on the same server. The middleware, Intel(R) DCM: Energy Director, however, is set up on a different server simulating the deployment scenario that the power management middleware locates on a centralized node managing the others in the entire data center. Intel(R) Node Manager 1.5 is enabled on the server running the benchmark and is managed by the instance of Intel(R) DCM: Energy Director.

The power saving application judges whether the performance gets impacted every minute. To check whether the front-end performance gets impacted, 99% is used as

the threshold for the values of percentage of well-done transactions for each of the five transaction types in the minute. To assess the back-end performance impact, samples without power saving is first collected. The benchmark for 1 hour is run and one data sample every 15 seconds is collected. For each data sample, the performance counter of number of transactions completed in the past 5 seconds is collected, resulting in a static sample set of 240 samples without power saving. During performance-aware power saving, in every minute the power saving application samples the performance counter 4 times, resulting in a running sample set with 4 samples. Welch t-test is then applied to the two data sets with a significance level of 0.005 to judge whether the back-end performance gets impacted.

TPCC-UVa benchmark is running with power saving of on-demand governor in Red Hat Enterprise 5.2 and with performance-aware power saving respectively. The following table shows the experimental results of performance reported by TPCC-UVa and average power during the running of the benchmark for both the baseline and performance-aware power saving.

Power Saving Method	Performance	Average Power
Baseline	300 tpmC	183W
Performance-Aware	300 tpmC	164W

From the experimental results, you can see that comparing with power saving of the power management module in the operating system, more than 10% power reduction is achieved with performance-aware power saving, while the performance of the benchmark gets little impact.

It is notable that the system used in this experiment is equipped with two relatively strong CPUs, but is with a relatively weak hard disk. It indicates that the system has relatively strong computational capabilities and is relatively weak in disk I/O performance. As a typical online transaction processing workload, the performance of TPCC-UVa is likely to be bounded by the performance of the disk on the system. In such a scenario, it is not difficult to conjecture that additional power reduction could be exploited by throttling the CPUs, if the workload performance is monitored and is observed to get maintained.

This experiment demonstrates that comparing with power saving of the power management module in operating systems which refers to the resource utilization counters, additional energy is saved with little performance impact by incorporating

application level performance feedback into power management software through a coordination of multiple components.

Creating and Handling Events

Events: Overview

Intel(R) DCM: Energy Director provides these event types:

- Predefined
- Custom

Predefined Events

Predefined events are events defined internally by Intel(R) DCM: Energy Director or alerts routed by Intel(R) DCM: Energy Director from devices. To get predefined events, subscribe an event handler. By default, all predefined events are subscribed to an event handler.

The table below lists the predefined events from the devices supported by Intel(R) DCM: Energy Director:

Supported Devices	Events
Intel(R) Intelligent Power Node Manager 1.5 servers	IPMI_PWR_UNIT, IPMI_PWR_SUPPLY, IPMI_PROCESSOR_THERMAL_TRIP, IPMI_FAN
DCMI 1.0 servers, for example, HP* iLO2/iLO3 servers/blades with DCMI interface exposed	IPMI_PWR_UNIT, IPMI_PWR_SUPPLY (not available for blade servers), IPMI_PROCESSOR_THERMAL_TRIP, IPMI_FAN (not available for blade servers)
Dell* iDRAC6 servers	IPMI_PWR_UNIT, IPMI_PWR_SUPPLY (not available for blade servers), IPMI_PROCESSOR_THERMAL_TRIP, IPMI_FAN (not available for blade servers)
Generic IPMI 2.0 servers, for example, common IBM* rack servers with IPMI interface exposed	IPMI_PWR_UNIT, IPMI_PWR_SUPPLY (not available for blade servers), IPMI_PROCESSOR_THERMAL_TRIP, IPMI_FAN (not available for blade servers)
APC* PDUs	PDU_LOW_LOAD, PDU_HIGH_LOAD, PDU_OVERLOAD

Dell* PDUs	PDU_LOW_LOAD, PDU_HIGH_LOAD, PDU_OVERLOAD, PDU_OUTLET_ON, PDU_OUTLET_OFF, PDU_OUTLET_LOW_LOAD, PDU_OUTLET_HIGH_LOAD, PDU_OUTLET_OVERLOAD (outlet level events are only available for a limited number of models, for example, Dell* Managed Rack PDU Dell 6605)
Emerson* PDUs	PDU_LOW_LOAD, PDU_HIGH_LOAD, PDU_OVERLOAD
ServerTech* PDUs	PDU_HIGH_LOAD, PDU_OUTLET_ON, PDU_OUTLET_OFF
Raritan* PDUs	PDU_LOW_LOAD, PDU_HIGH_LOAD, PDU_OVERLOAD, PDU_OUTLET_ON, PDU_OUTLET_OFF (some certain events are only available for a limited number of models)
APC* UPSes	UPS_LOW_BATTERY, UPS_SHUTDOWN, UPS_ON_BYPASS
Dell* UPSes	UPS_LOW_BATTERY, UPS_BAD_INPUT
Eaton* UPSes	UPS_LOW_BATTERY, UPS_BAD_INPUT, UPS_BAD_BATTERY
Emerson* UPSes	UPS_LOW_BATTERY

Custom Events and Notifications

Custom events are defined based on power consumption or temperature. A custom event evaluates the condition periodically and is triggered each time the custom condition is satisfied. To define a custom event, use defineCustomEvent.

Custom notifications are triggered when the custom condition passes a certain threshold. To define a custom notification, use defineNotification.

See Also

defineCustomEvent

defineNotification

subscribePredefinedEvent

PredefinedEventType

CustomEventType

Handling Events

Handling Events

The Intel(R) DCM: Energy Director event structure is based on WS-eventing. The event handler must be able to parse WS-eventing.

To register an event handler:

1. Create an event handler. The event handler must be an HTTP or TCP server listening to a port that you define.
2. Supply the event handler URL.

See Also

Event and Notification Message Format

SeverityLevel

CustomEventType

PredefinedEventType

Event and Notification Message Format

Each Intel(R) DCM: Energy Director event or notification contains the following fields:

Field	Description
`<dcm:PredefinedEventType>`	The type of predefined event. For custom events, the value is always *Null*. See PredefinedEventType.
`<dcm:CustomEventId>`	The ID of the custom event that triggered the call. ID is provided by `defineCustomEvent`.
`<dcm:NotificationId>`	The ID of the custom event that triggered the call. ID is provided by `defineNotification`.
`<dcm:NotificationType>`	The field indicating whether the evaluation condition starts getting triggered or stop being triggered.
`<dcm:PolicyId>`	The ID of the control policy that triggered the

	call. ID is provided by setPolicy or `setPolicyEx`. For custom events, the value is always *Null*.
`<dcm:EntityID>`	The ID of the entity that the event applies.
`<dcm:EventTime>`	The time that the event occurs.
`<dcm:NotificationTime>`	The time that the notification occurs.
`<dcm:SeverityLevel>`	The severity level of the event. For custom events, the value is always *CUSTOM*. See SeverityLevel.
`<dcm:Signature>`	A base64 encoded signature that can be used to verify the event integrity.
`<dcm:Info>`	Additional information for the event. For example, for predefined events, it includes a node IP address. For custom events, it includes the event value. Example: If the event type is MAX_PWR, event value is the number of the maximum power, in Watts.
`<dcm:Data>`	A BLOB data translated to base64 format that comes directly from the client system. For example, IPMI data. Intel(R) DCM does not change the BLOB data. The value may be *Null*.

See Also

Predefined Event Message Example

Custom Event Message Example

Custom Notification Message Example

Code Sample: HTTP Event Handler

Predefined Event Message Example

The following is an example of a predefined event message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:dcm="http://wsdl.intf.dcm.intel.com/events">
```

```
      <s12:Header>
            <wsa:Action>http://wsdl.intf.dcm.intel.com/event</wsa:Actio
n>
            <wsa:MessageID>uuid:8fd69fe1-29c7-4a26-ab8b-
d628b9c2ec39</wsa:MessageID>
      </s12:Header>
      <s12:Body>
            <dcm:Event>
                  <dcm:PredefinedEventType>CONFIGURATION_CHANGED</dcm:P
redefinedEventType>
                  <dcm:PolicyId>-1</dcm:PolicyId>
                  <dcm:CustomEventId>0</dcm:CustomEventId>
                  <dcm:EntityID>-1</dcm:EntityID>
                  <dcm:EventTime>2010-02-25T3:49:29Z</dcm:EventTime>
                  <dcm:SeverityLevel>INFORMATIVE</dcm:SeverityLevel>
                  <dcm:Info>The value of property:
NODE_THERMAL_SAMPLING_FREQUENCY has changed from: 60 to: 30</dcm:Info>
                  <dcm:Data></dcm:Data>
            </dcm:Event>
            <dcm:Signature>yA7pqfBp+/bqaIXInX1HESsGIMrSAz/1NdSyHE9srwg=
</dcm:Signature>
      </s12:Body>
</s12:Envelope>
```

See Also

Handling Events

Events Interface: Overview

Custom Event Message Example

The following is an example of a custom event message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:dcm="http://wsdl.intf.dcm.intel.com/events">
      <s12:Header>
            <wsa:Action>http://wsdl.intf.dcm.intel.com/event</wsa:Actio
n>
            <wsa:MessageID>uuid:0585cf43-dda6-4f07-ac6f-
ad16f4d001c7</wsa:MessageID>
      </s12:Header>
      <s12:Body>
            <dcm:Event>
                  <dcm:CustomEventId>2</dcm:CustomEventId>
                  <dcm:EntityID>2</dcm:EntityID>
                  <dcm:EventTime>2010-02-25T3:54:00Z</dcm:EventTime>
                  <dcm:SeverityLevel>CUSTOM</dcm:SeverityLevel>
                  <dcm:Info>Custom Event: 2, Entity: 2, Threshold: 10,
Type: TOTAL_AVG_PWR, Condition: GREATER_THAN, Evaluation Period: 30,
Description: event on node1, Time: Thu Feb 25 11:54:00 CST 2010,
Severity: CUSTOM, Value: 149;</dcm:Info>
```

```
                  <dcm:Data></dcm:Data>
            </dcm:Event>
            <dcm:Signature>Qs5VxEC8dylkskdCTe/NysT03eTcVZe8FvK8Ebf2JW8=
</dcm:Signature>
      </s12:Body>
</s12:Envelope>
```

See Also

Event and Notification Message Format

Predefined Event Message Example

Code Sample: HTTP Event Handler

SeverityLevel

CustomEventType

defineCustomEvent

Handling Events

Custom Notification Message Example

The following is an example of a custom notification message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:dcm="http://wsdl.intf.dcm.intel.com/events">
      <s12:Header>
    <wsa:Action>http://wsdl.intf.dcm.intel.com/event</wsa:Action>
    <wsa:MessageID>uuid:85785334-cac7-4e6f-
bc47bb6f672f8387</wsa:MessageID>
      </s12:Header>
      <s12:Body>
            <dcm:Notification>
          <dcm:NotificationId>392</dcm:NotificationId>
          <dcm:NotificationType>Start</dcm:NotificationType>
          <dcm:EntityID>780</dcm:EntityID>
          <dcm:NotificationTime>2010-11-
26T7:58:00Z</dcm:NotificationTime>
          <dcm:SeverityLevel>INFORMATIVE</dcm:SeverityLevel>
          <dcm:Info>Notification: 392, Entity: 780, Threshold: 60, Type:
TOTAL_AVG_PWR, Condition: GREATER_THAN, Evaluation Period: 40,
Description: condition is met in this evaluation period: true, in last
evaluation period: false, Time: Fri Nov 26 15:58:00 CST 2010, Severity:
INFORMATIVE, Value: 317;</dcm:Info>
          <dcm:Data></dcm:Data>
    </dcm:Notification>
        <dcm:Signature>h4Ud6w86MSMaphi9AM6937hsHOFS27F6Zw5oKpCk0o8=
</dcm:Signature>
      </s12:Body>
</s12:Envelope>
```

See Also

Event and Notification Message Format

SeverityLevel

defineNotification

API Reference

What's New

This section outlines new API changes implemented since Intel(R) DCM: Energy Director 3.0.

To view a complete version of API reference, go to `<install_dir>\external\apache-tomcat\webapps\DataCenterManager\DevGuide\API_Reference\index.html`.

The following tables list the API changes by components since Intel(R) DCM: Energy Director 3.0.

Datacenter Modeling Interface

Item	Description	Notes
`enum EntityProperty { ..., INBAND_XXXXs, GROUP_LIMIT_ON_ENCLOSURE, MAX_PWR_BUDGET, MIN_PWR_BUDGET, PDU_PWR_AS_IT_EQPMNT_PWR, MGMT_PROCESSOR_TYPE, FIRMWARE_VERSION, MGMT_CONSOLE_URL, TEMPERATURE_UPPER_LIMIT, SERIAL_NUMBER, REPORT_INLET_TEMPERATURE }`	`INBAND_XXXXs`: Properties for managing a server through the OS in-band protocols, e.g., WMI (new in Intel(R) DCM: Energy Director 3.1) `GROUP_LIMIT_ON_ENCLOSURE`: The property indicates whether the enclosure is treated as the device endpoint in group power control (new in Intel(R) DCM: Energy Director 3.1) `MAX_PWR_BUDGET & MIN_PWR_BUDGETL`: Optional properties dictating the upper bound and the lower bound of device level power budget allocation. (new in Intel(R) DCM: Energy Director 3.2) `PDU_PWR_AS_IT_EQPMNT_PWR`: Optional property specifying whether to use PDU power to calculate physical group IT equipment power (new in Intel(R) DCM: Energy	New feature

	Director 3.3) `MGMT_PROCESSOR_TYPE, FIRMWARE_VERSION & MGMT_CONSOLE_URL`: The properties provide the management processor information (new in Intel(R) DCM: Energy Director 3.4) `TEMPERATURE_UPPER_LIMIT`: The property specifies the upper limit of the inlet temperature of a room as the criterion for cooling analysis (new in Intel(R) DCM 3.4) `SERIAL_NUMBER`: The property is used to identify the corresponding enclosure of HP SL systems managed by SL Advanced Power Manager (new in Intel(R) DCM: Energy Director 3.4) `REPORT_INLET_TEMPERATURE`: The property is used to monitoring inlet temperature through PDU temperature sensors. It applies to PDUs which are able to provide temperature readings in real-time monitoring (new in Intel(R) DCM: Energy Director 3.7)	
`enum ProtocolType { ..., INBAND_PROTOCOL }`	`INBAND_PROTOCOL`: new protocol type in identifying/discovering devices managed with in-band mechanism (new in Intel(R) DCM: Energy Director 3.3)	New feature
`String getEntityProperty (int entityId, EntityProperty propertyName)`	Retrieve the value of a single property of an entity (new in Intel(R) DCM: Energy Director 3.4)	API usability improvement

Query/Metrics Interface

Item	Description	Notes
`EnumerationData getBatchQueryData(int[]`	Query/metric API on a	Usability improvements

entityIds, QueryType queryType, Date startTime, Date endTime, int aggPeriod) EnumerationData getBatchMetricData (int[] entityIds, MetricType metricType, Date startTime, Date endTime, int aggPeriod)	batch of entities (new in Intel(R) DCM: Energy Director 3.2)	on query/metric API
EnumerationRawData dumpMetricData (int entityId, MetricType metricType, Date startTime, Date endTime)	Dump IT equipment power data without aggregation (new in Intel(R) DCM: Energy Director 3.2)	Usability improvements on data access API
startCollectingPowerDistribution (int entityId) PowerDistributionInfo getPowerDistribution (int entityId) stopCollectingPowerDistribution (int entityId)	APIs for retrieving statistics of historical IT equipment power on physical entities (new in Intel(R) DCM: Energy Director 3.2)	New feature
MeasurementDistributionInfo getInletTemperatureDistribution (int entityId)	Retrieve the current inlet temperature distribution of a physical group (new	New feature

	in Intel(R) DCM 3.2)	
`int getCoolingIndicator (int groupId, int allowedTemperature, int recommendedTemperature)`	Calculate the indicator on how well a group of servers are cooled against over-heating (new in Intel(R) DCM: Energy Director 3.1)	New feature
`int getOverCoolingIndicator (int groupId, int allowedTemperature, int recommendedTemperature)`	Calculate the indicator on how well a group of servers are over-cooled (new in Intel(R) DCM: Energy Director 3.6)	New feature
`int getTemperatureRise (int groupId)`	Calculate the air temperature rise across a group of servers with outlet temperature and airflow monitoring capability	New feature

	(new in Intel(R) DCM: Energy Director 3.6)	

enum QueryType {
...,
CPU_UTIL,
DISK_IO
}
	CPU_UTIL: CPU utilization of servers managed with in-band mechanism (new in Intel(R) DCM: Energy Director 3.3)
	DISK_IO: Disk I/O data of servers managed with in-band mechanism (new in Intel(R) DCM: Energy Director 3.5)
		New feature

Control Policy Interface

Item	Description	Notes
enum PolicyType { ..., PWR_EFFICIENT, MEM_PWR_LIMIT	PWR_EFFICIENT: Policies driving the servers to the P-state with the lowest power and	New feature

}	performance without clock throttling, which brings the best power efficiency (new in Intel(R) DCM: Energy Director 3.2) MEM_PWR_LIMIT: Power limit on the server's memory sub-system (new in Intel(R) DCM: Energy Director 3.1)	

Events Interface

Item	Description	Notes
enum CustomEventType { ..., IT_EQPMNT_PWR }	New custom event type corresponding to the metric type of IT_EQPMNT_PWR (new in Intel(R) DCM: Energy Director 3.2)	New feature

Expert System Interface

Item	Description	Notes
ServerPowerProfiles[] enumrateServerPowerProfiles ()	Provide the server power profiles for different server models and configurations	New feature

	(new in Intel(R) DCM: Energy Director 3.1)	

`int addServerPowerProfile (ServerPowerProfile serverPowerProfile)`
`updateServerPowerProfile (ServerPowerProfile serverPowerProfile)`
`removeServerPowerProfile (int profileId)`
	Manage the server power profiles (new in Intel(R) DCM: Energy Director 3.4)	New feature

`NetworkDevicePowerProfile[] enumerateNetworkDevicePowerProfiles()`	Provide the network device power profiles for different models and configurations (new in Intel(R) DCM: Energy Director 3.4)	New feature
`StorageDevicePowerProfile[] enumerateStorageDevicePowerProfiles()`	Provide the storage device power profiles for different models and configurations (new in Intel(R) DCM: Energy	New feature

	Director 3.5)	

`int addNetworkDevicePowerProfile (NetworkDevicePowerProfile networkDevicePowerProfile)`
`updateNetworkDevicePowerProfile (NetworkDevicePowerProfile networkDevicePowerProfile)`
`removeNetworkDevicePowerProfile (int profileId)`
	Manage the network device power profiles (new in Intel(R) DCM: Energy Director 3.4)	New feature

`int addStorageDevicePowerProfile (StorageDevicePowerProfile storageDevicePowerProfile)`
`updateStorageDevicePowerProfile (StorageDevicePowerProfile storageDevicePowerProfile)`
`removeStorageDevicePowerProfile (int profileId)`
	Manage the network device power profiles (new in Intel(R) DCM: Energy Director 3.5)	New feature

`CoolingStatusInfo analyzeCoolingStatus (int roomId)`	Evaluate the current cooling status of a room and providing suggestions for cooling optimization (new in Intel(R) DCM: Energy Director 3.3)	New feature

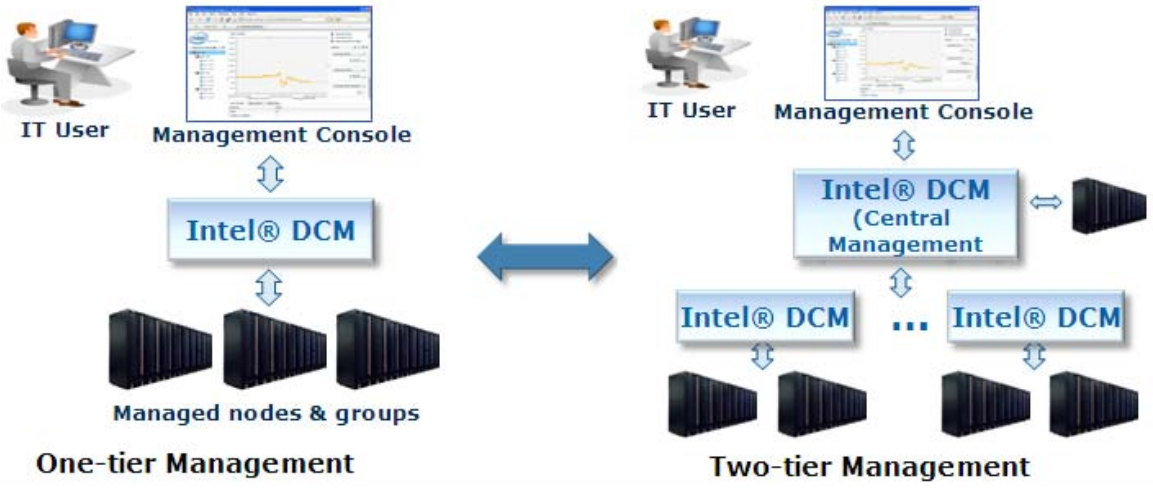
`EvaluationResult evaluateLowUtilizationServer (int entityId)`	Evaluate whether a server is an under-utilized one (new in Intel(R) DCM: Energy Director 3.3)	New feature

`int addPowerModel (String discription, PowerUtilizationData[] powerUtilizationData)`
`ModelInfo[] enumeratePowerModels()`
`void removePowerModel (int modelId)`
`double predictPower (int modelId, double[] utilizationData)`
	Build and use advanced power models taking utilization as input to predict power (new in Intel(R) DCM: Energy Director 3.6)	New feature

Deploying the Servers

Deployment: Overview

Intel(R) DCM: Energy Director can be deployed in one-tier or two-tier management environment as shown in the following figure:



In one-tier management environment, Intel(R) DCM: Energy Director server manages nodes (NM, DCMI, PDU, etc.) using a southbound node connector service through the interface exposed by the node. Your management console controls the nodes directly. Intel(R) DCM: Energy Director supports up to 5000 managed nodes in one-tier management environment.

In two-tier management environment, Intel(R) DCM: Energy Director server can be categorized as tier-one server and tier-two server according to the management structure position in the tiers management hierarchy. Intel(R) DCM: Energy Director tier-one and tier-two servers are binary identical. The runtime configuration enables Intel(R) DCM: Energy Director server instance to perform as tier-one or tiers. Intel(R) DCM: Energy Director system supports single tier-one server but multiple tier-two servers.

Tier-one Server

Tier-one server performs the central management server role in the distributed tiers management mode, including:

- Provides the single management entry point to your management console through Intel(R) DCM: Energy Director WSDL API.
- Oversees the entire Intel(R) DCM: Energy Director hierarchy and maintains global hierarchy information.

- Manages tier-two servers (for example, manages hierarchy, policy, event, measurements, etc.) through WDSL interface and http based event channel.
- Performs global policy intelligence:
  - Analyzes policy domain intersection with tier-two servers
  - Dispatches policy to associated tier-two server, if the policy does not span multiple tier-two servers
  - Divides the policy to sub policies to remove tier-two server intersection, if the policy spans multiple tier-two servers
  - Passes the sub policies to the corresponding tier-two servers
- Statically allocates the power budget for tier-two servers.
- Maintains group level measurement (raw and compressed), and optionally stores node level compressed data (configurable).
- Manages tier-two servers by Intel(R) DCM: Energy Director connector through standard Intel(R) DCM: Energy Director API calls.
- Manages nodes directly by extensible node connectors though the exposed management interfaces of the nodes.

Tier-two Server

The tier-two server manages subsets of nodes, it is controlled by central management server, its role including:

- Managed by tier-one server through existing Intel(R) DCM: Energy Director API (WSDL API and http based event channel).
- Maintains the hierarchy information within its management zone level.
- Performs full policy intelligence for the policy within its management zone.
- Provides group level measurement data to the tier-one server.
- Stores node level measurements (raw and compressed) in tier-two data storage.
- Acts as footprint with option to configure. Can be deployed as virtual appliance.

Changing Server Role

Server Roles Introduction

These are possible roles of Intel(R) DCM: Energy Director server:

- standalone server
- central management server
- tier-two server.

Intel(R) DCM tier-one and tier-two servers are binary identical. The runtime configuration enables Intel(R) DCM: Energy Director server instance to perform as tier-one or tiers. Intel(R) DCM: Energy Director server performs one of the following roles at one time:

Standalone Server:

- Works as one tier
- Managed by the management console through Intel(R) DCM: Energy Director API
- Manages node entities

Central Management Server:

- Is part of a two-tier configuration
- Managed by the management console through Intel(R) DCM: Energy Director API
- Manages Intel(R) DCM: Energy Director server(s)
- Manages node entities (Optional)

Tier-two Server:

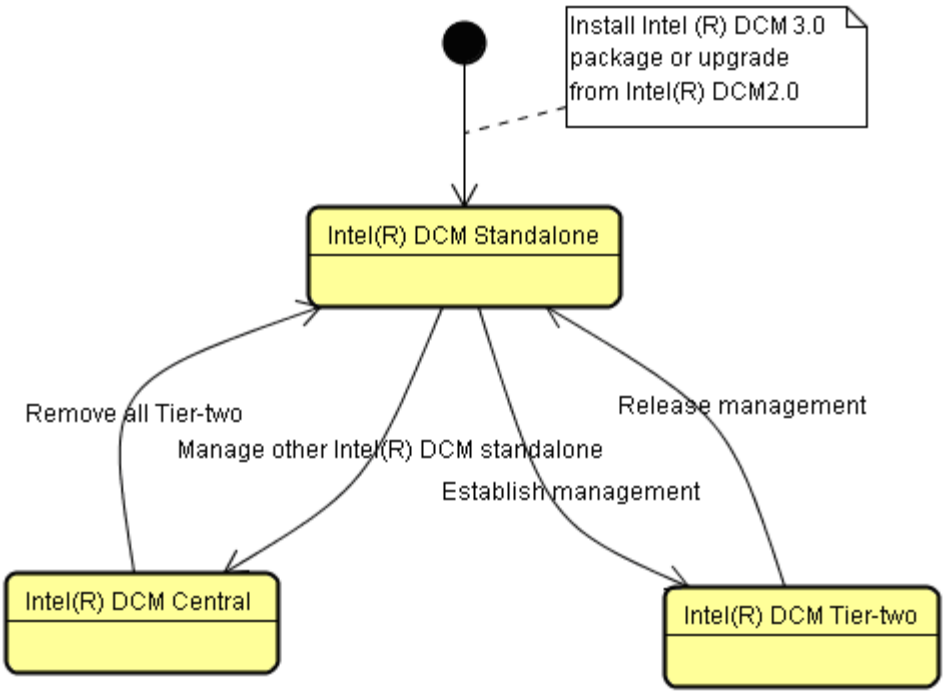
- Is part of a two- tier configurations
- Managed by the central management server
- Manages node entities

In Intel(R) DCM: Energy Director, Server Role flag is always stored in the database, and runtime changes according to the system hierarchy.

Changing Server Role

You can change the role of an Intel(R) DCM: Energy Director server, according to the server configuration.

The following figure shows how the server role changes with the addition or removal of other Intel(R) DCM: Energy Director servers.



Initial State: Standalone Server

After you upgrade or install Intel(R) DCM: Energy Director, the initial state is set as standalone server.

Changing Standalone Server to Central Management Server

You can change Standalone Server to Central Management Server by adding another standalone server at runtime using API addEntity. This API also changes the managed standalone server to a tier-two Server.

When changing a standalone server to tier-two server, critical data in the tier-two server is automatically uploaded to the central server through the following transactions:

When changing standalone server to tier-two server in the two-tier environment, critical data existing in the tier-two server will be automatically uploaded to the central server through a series of transaction, including:

- Generate a new GUID signature for Intel(R) DCM: Energy Director to have version control
- Lock Intel(R) DCM: Energy Director modeling/policy/event change
- Pull critical data (hierarchy/policy/event) from the tier-two server to the central server and establish the critical data mapping relationship between the two servers

- Establish the measurement data mapping
- Commit data to complete the server role change

NOTE

- Adding a standalone server may take long time, in order to keep the data consistency, the following behaviors happens during the transaction:
  - Central server denies all modification operation for modeling, policy and event
  - Tier-two server rejects web service call from other software
- If the role change process fails or is interrupted, the Intel(R) DCM: Energy Director central server will remove the uploaded dirty data at the next boot.

CAUTION

Interrupting the role change process may disrupt data consistency.

See Also

Changing Central Management Server to Standalone Server

Changing Standalone Server to Tier-two Server

Standalone server changes to tier-two server when receiving the call for API addEntity. The critical data is uploaded to the Central Server, including:

- The tier-two server hierarchy
- Existing policies on the tier-two server. These policies keep the current status
- Existing custom events on the tier-two server
- Mapping relationship of measurement data and entities. Measurement data for performance consideration is not kept.
- The stand alone server changes role to tier-two server upon successful upload. If the process fails, it rolls back to standalone server.

See Also

Changing Tier-two Server to Standalone Server

Changing Central Management Server to Standalone Server

You can change the Central Management Server to Standalone server by removing all tier-two servers using the API `removeEntity`, at runtime. The process is the same as changing standalone server to tier-two server. However, uploading data behavior is changed to removing redundant data at the central server.

NOTE

- If the tier-two server manages many entities at the same time, removing a tier-two server can take a long time.
- If you remove a tier-two server (set `delCollectedData` parameter as *FALSE* when calling API `removeEntity`), Intel(R) DCM: Energy Director keeps the data (hierarchy, policy, event, etc.) on the tier-two server. If you set `delCollectedData` parameter as *TRUE*, all the tier-two server related data in the tier-two database is deleted. In both cases, the tier-two related data on the central server is removed.

See Also

Changing Standalone Server to Central Management Server

Changing Tier-two Server to Standalone Server
Upon successfully removing a tier-two server from Intel(R) DCM: Energy Director, the tier-two server automatically changes to standalone server.

See Also

Changing Standalone Server to Tier-two Server

Controlling Communication Timeout between Central Server and Tier-two Server

In two-tier management environment, tier-two server is managed through web service interface, you can configure the value of the web service invocation timeout.

There are different timeout values for different services, you can configure the base timeout value directly, other values will be changed automatically.

- The base timeout value applies to the heartbeat service between central server and tier-two server, the default value is 60s. This value can be configured in *user.config.xml* by adding the following entry

  ```
  <entry key="SERVER_REQUEST_TIMEOUT_SECOND">60</entry>
  ```

  The range of the value is 0 to 1800, in seconds. *0* denotes no timeout.

- Most of other services consume much more time than the heartbeat service. Intel(R) DCM applies different timeout values (multiple of base timeout) to these services.

In most occasions, the default timeout value is recommended, you don’t need to configure. However, in the following occasions, you need to configure the value to control timeout:

- Network latency is extremely high and heartbeat service timeout happens occasionally.
- Tier-two server has too many nodes (more than 5000) and time-consuming service timeout happens.
- You want to disable timeout mechanism (use value 0) for diagnostic purpose.

NOTE

The new configuration only takes effect after *Intel(R) Data Center Manager Server* service is restarted.

Consistency Model

Data Categories

The following lists the persistent data categories of Intel(R) DCM: Energy Director:

- Critical data
- Configuration data
- Miscellaneous data

Critical data

Includes datacenter hierarchy, user defined policies and user defined custom events. In two-tier management mode, some critical data, for example, self hierarchy under the tier-two server, belongs to the tier-two server and exists in both tier-two server and central server.

Configuration data

Includes all global configuration properties and user subscribed predefined events. All configuration data applies throughout Intel(R) DCM: Energy Director system level. It exists in the central server and all the tier-two servers.

Miscellaneous data

Includes all measurement data (both raw and compressed data), policy history data and action log.

Consistency Model

In two-tier management environment, both central server and tier-two servers provide persistent storage to keep the data persistent in a distributed system. Intel(R) DCM: Energy Director consistency model operates with the data exists in central server and tier-two server.

There are three persistent data categories, however, only critical data and configuration data are included in the consistency model.

In general, Intel(R) DCM: Energy Director uses eventual consistency model. From a user's perspective, the data in distributed system is synchronized eventually.

For critical data and configuration data, different mechanisms are used:

Critical data consistency is maintained through consistency protocol. Modification operation starts from tier-two server, ends at central server.

- If failure happens in tier-two server, the external operation fails with a consistent condition.
- If failure happens in central server, the external operation fails with an inconsistent condition, the modification in tier-two server is rolled back eventually. This is triggered by either the next modification operation, or background activity which happens every three minutes.

Configuration data consistency is maintained through automatic synchronization. Modification operation starts from central server, ends at tier-two server.

- If failure happens in central server, the external operation fails with a consistent condition.
- If failure happens in tier-two server, the external operation succeeds with an internal inconsistent condition. Intel(R) DCM: Energy Director will synchronize the configuration data from central server to tier-two server through the background activity every three minutes.

See Also

Data Categories

Inconsistent Situation

Inconsistent situation only happens in some special cases, for example:

- Unexpected failure in central server or tier-two server while Intel(R) DCM: Energy Director is handling external modification operation.
- Tier-two server replacement, see Failover Solutions.

When inconsistent situation happens, the following issues may happen under a temporary inconsistent condition:

- Measurement data is invalid:

  Some measurement data, which are monitored or aggregated in tier-two server, is invalid due to the incorrect hierarchy data on tier-two server.

- Operation fails:

  Some operations, which are conducted on tier-two server, for example, to power on/off one group in tier-two server or to rediscover one node in tier-two server may fail. In this case, critical data and configuration data access operation are not impacted, including retrieving and modification.

Critical data inconsistent situation is detected through background activity every three minutes. Critical data synchronization is triggered automatically and related events is issued.

You need to be aware of the situation when the issues happen. You can wait three minutes for a Intel(R) DCM: Energy Director internal consistency check, or modify critical data in tier-two server manually and check the related events. After the synchronization finishes, you can redo your operation.

See Also

Failover Solutions

CRITICAL_DATA_SYNCHRONIZATION_STARTING

CRITICAL_DATA_SYNCHRONIZATION_ENDED

Migrating from Previous Versions

Migrating from Version 3.0 to 3.x

About Migrating from Version 3.0

If you have developed Intel(R) DCM: Energy Director 3.0 based applications, you can migrate your applications so as to use Intel(R) DCM: Energy Director 3.x new features.

This section provides information on how to migrate your applications from Intel(R) DCM: Energy Director 3.0 based applications to Intel(R) DCM: Energy Director 3.x based applications.

Before migration, do the following:

1. Read the Intel(R) DCM: Energy Director Developer's Guide to understand Intel(R) DCM: Energy Director 3.x new features, and new APIs.
2. Consider what components in your applications you want to migrate to use Intel(R) DCM: Energy Director 3.x new or redefined features.
3. Refer to Intel(R) DCM: Energy Director Installation Guide on how to install or upgrade to Intel(R) DCM: Energy Director 3.x.
4. Read the instructions of each component before modifying your application.
5. Find the difference between Intel(R) DCM: Energy Director 3.0 and 3.x to better understand the new or redefined interfaces.
6. Refer to the steps in each topic and code samples of each component to implement to your applications.

See Also

Data Center Modeling

- Managing Cisco Switches with Cisco EnergyWise Technology Enabled
- Managing Microsoft Windows Servers through WMI

Monitoring Data Center

- Utilization-based Power Estimation
- Calculating Cooling Indicator

Control Policies

- Power Limit on HP Enclosures

- Memory Power Limit on Node Manager 2.0 Platforms

Others

- Knowledge Base of Server Power Profiles

Data Center Modeling

Managing Cisco Switches with Cisco EnergyWise Technology Enabled

Intel(R) DCM: Energy Director 3.x supports managing Cisco* switches with Cisco* EnergyWise Technology enabled as network devices by specifying the connector `com.intel.dcm.plugin.ciscoSwitchPlugin` when adding entities. Intel(R) DCM: Energy Director monitors the instantaneous power consumption of the switches.

Managing Cisco UCS Chassis/Enclosures

Intel(R) DCM: Energy Director 3.x supports managing Cisco* Unified Computing System (UCS) chassis/enclosures by communicating with Cisco* UCS Managers as an experimental feature. To add Cisco* UCS chassis/enclosures, you need to specify the connector `com.intel.dcm.plugin.ciscoUCSChassisPlugin`, and provide the corresponding addresses and credentials for communication with the Cisco* UCS Manager. The chassis/enclosure managed by the Cisco* UCS Manager are identified with their distinguished names. The capability of instantaneous power monitoring and power control on the enclosure level are provided on the platforms. Under standalone deployment of Intel(R) DCM: Energy Director, it is also supported to automatically build and rebuild the hierarchy under the chassis/enclosures by calling rediscoverEntity on the entities.

NOTE

The feature is validated on Cisco* UCS simulators.

Managing HP iLO Platforms with Power Regulator Capability

Intel(R) DCM: Energy Director 3.x supports managing HP* iLO platforms through the combination of DCMI interface and SSH interface.

The entities are added by specifying the connector `com.intel.dcm.plugin.hpiLOPlugin`. In addition to the generic power monitoring

and capping capabilities provided in the generic DCMI connector (`com.intel.dcm.plugin.Dcmi10Plugin`), one may further enforce power efficient policies on HP* iLO platforms. The policies are applied by configuring HP* Power Regulator in Static Low Power mode.

See Also

Power Efficient Policies

Managing Fujitsu Platforms with DCMI Interface Exposed

Intel(R) DCM: Energy Director 3.x supports managing Fujitsu* platforms with DCMI interface exposed. The servers are managed by specifying the generic DCMI connector, `com.intel.dcm.plugin.Dcmi10Plugin`, in adding the entities. The features of power monitoring and power control (depending on the capability of the specific system) are provided.

Managing Cisco Rack Servers with DCMI Interface Exposed

Intel(R) DCM: Energy Director 3.x supports managing Cisco* rack servers with DCMI interface exposed. The servers are managed by specifying the generic DCMI connector, `com.intel.dcm.plugin.Dcmi10Plugin`, in adding the entities. The features of power monitoring and power control (depending on the capability of the specific system) are provided.

Managing Dell Enclosures and Blades through SSH or HTTPS/WS-MAN

Intel(R) DCM: Energy Director 3.x supports managing Dell* enclosures through SSH opened for connection on CMC. You need to specify the connector `com.intel.dcm.plugin.dellCMCSSHPlugin` when adding the entities. The capability of enclosure level power consumption monitoring and limiting is supported. The API getEnclosureAndBladeInfo is also supported on the entities to collect the blade information under the enclosures.

Intel(R) DCM: Energy Director 3.x also supports managing individual Dell* blade servers inside Dell* enclosures through SSH opened for connection on CMC. When adding the entities, You need to specify the connector `com.intel.dcm.plugin.dellBladePlugin` and specify the service tags to identify the blade servers to be managed.

In standalone deployment of Intel(R) DCM: Energy Director, you can call rediscoverEntity on Dell* enclosure entities so that the hierarchy under the enclosures are built or rebuilt automatically with new blades added, obsolete blades disassociated, and other blades moved in hierarchy.

When the connector `com.intel.dcm.plugin.dellCMCPlugin` is used to manage Dell* enclosures with the firmware version no earlier than 4.40 over HTTPS/WS-MAN, Energy Director 3.x also supports monitoring and limiting the enclosure level power of the enclosures.

Managing Microsoft Windows Servers through WMI

Intel(R) DCM: Energy Director 3.x supports managing Microsoft* Windows Servers with WMI interface exposed.

To add the servers, Intel(R) DCM: Energy Director should be deployed on a Windows server with Microsoft* Visual C++ 2010 Redistributable installed. You need to specify the connector `com.intel.dcm.plugin.WMIWindowsPlugin` when adding the entities. Windows full computer names are used as the platform identifiers to identify and discriminate different servers.

Applying the utilization-based power estimator to the entities provides dynamic estimation of the power consumption for the entities.

NOTE

To secure the remote WMI connection in managing Microsoft* Windows servers through WMI, please refer to http://msdn.microsoft.com/library/aa393266.aspx.

See Also

Power Estimation

Utilization-based Power Estimation

Managing Linux Servers through SSH

Intel(R) DCM: Energy Director 3.x supports managing Red Hat Enterprise Linux Servers* and Novell SUSE Linux Enterprise Servers* with SSH opened for connection.

You need to specify the connector `com.intel.dcm.plugin.SSHLinuxPlugin` when adding the entities. Linux FQDNs are used as the platform identifiers to identify and discriminate different servers.

Applying the utilization-based power estimator to the entities provides dynamic estimation of the power consumption for the entities.

See Also

Power Estimation

Utilization-based Power Estimation

Managing Xen Servers through SSH

Intel(R) DCM: Energy Director 3.x supports managing Xen servers with SSH opened for connection.

You need to specify the connector `com.intel.dcm.plugin.SSHXenPlugin` when adding the entities. FQDNs are used as the platform identifiers to identify and discriminate different servers.

Applying the utilization-based power estimator to the entities provides dynamic estimation of the power consumption for the entities.

See Also

Power Estimation

Utilization-based Power Estimation

Managing VMWare ESX/ESXi Servers through SSH

Intel(R) DCM: Energy Director 3.x supports managing VMWare* servers with SSH opened for connection.

You need to specify the connector `com.intel.dcm.plugin.SSHESXPlugin` when adding the entities. ESX/ESXi server names are used as the platform identifiers to identify and discriminate different servers.

If server power reading is provided in ESX/ESXi on the server, one may enable Intel(R) DCM: Energy Director to retrieve the power reading by following the configuration below:

1. Run esxtop in command prompts
2. Press 'p' to switch to the power view
3. Press 'w' to write the file with the namd of '.dcmpower'.

If ESX/ESXi does not provide the funtion of server power reading, one may apply the utilization-based power estimator to the entities for dynamic estimation of the power consumption for the entities.

See Also

Monitoring the Datacenter: Overview

Power Estimation

Utilization-based Power Estimation

Managing Raritan PDUs

Intel(R) DCM: Energy Director 3.x supports managing Raritan* PDUs for instantaneous power monitoring and real-time monitoring. It also provides outlet level power monitoring, outlet power status monitoring and control depending on the capability of the specific PDU.

See Also

Monitoring the Datacenter: Overview

Real-time Monitoring

Managing Enclosures of HP SL Systems

Intel(R) DCM: Energy Director 3.x supports managing enclosures of HP* ProLiant SL Scalable systems (e.g., HP ProLiant s6500 chassis) by communicating with HP* SL Advanced Power Managers (APM). To add enclosures of HP* SL systems, you need to specify the connector `com.intel.dcm.plugin.hpSLAPMPlugin`, and provide the corresponding addresses and credentials for communication with the HP* SL APM. The enclosures managed by the HP* SL APM are identified with their serial numbers. The capability of instantaneous power monitoring and power control on the enclosure level are provided on the platforms.

Managing Chatsworth PDUs

Intel(R) DCM: Energy Director 3.x supports managing Chatsworth* PDUs for instantaneous power monitoring and real-time monitoring. It also provides outlet level power monitoring, outlet power status monitoring and control depending on the capability of the specific PDU.

Intel(R) DCM: Energy Director 3.x supports reporting temperature values from the temperature sensors of the PDUs as well.

See Also

Monitoring the Datacenter: Overview

Real-time Monitoring

Managing Fujitsu Enclosures and Switches

Intel(R) DCM: Energy Director 3.x supports managing Fujitsu* enclosures and switches with SNMP interface exposed. To add Fujitsu* enclosures or switches, you need to specify the generic SNMP connector `com.intel.dcm.plugin.snmpPlugin` and provide the corresponding addresses and credentials for SNMP communication. The functions of instantaneous power monitoring and inlet temperature monitoring are provided on the platforms.

Establishing the Basis of Managing More SNMP Devices

Intel(R) DCM: Energy Director 3.x establishes the basis of managing more SNMP devices through the generic SNMP connector `com.intel.dcm.plugin.snmpPlugin`.

A configuration file provides the rules and details on how devices are identified and managed. It is expected that the configuration file will be updated in the future so that more SNMP devices can be managed without the need of upgrading Intel(R) DCM: Energy Director.

Retrieving Management Processor Information

Intel(R) DCM: Energy Director 3.x provides the capability of retrieving management processor information for HP* iLO 2/3/4 devices, HP* OA devices, Dell* iDRAC6/7 devices, IBM* IMM2 devices, and IBM* AMM devices. The processor information includes the management processor type, firmware version, and management console URL, which are retrieved through the entity property `MGMT_PROCESSOR_TYPE`, `FIRMWARE_VERSION`, and `MGMT_CONSOLE_URL`.

Retrieving In-Band Data for Out-of-Band IPMI Devices

Intel(R) DCM: Energy Director 3.x supports retrieving in-band data (e.g., CPU utilization data) for out-of-band IPMI devices.

By specifying the entity properties, `INBAND_OS_TYPE`, `INBAND_ADDRESS`, `INBAND_USERNAME`, and `INBAND_PASSWORD` for out-of-band IPMI entities, in-band communication channels are attached to the out-of-band entities. The entities then obtain the corresponding capability of in-band utilization monitoring with utilization data retrieved.

See Also

Managing Microsoft Windows Servers through WMI

Managing Linux Servers through SSH

Managing Xen Servers through SSH

Managing VMWare ESX/ESXi Servers through SSH

Monitoring CPU Utilization for Devices Managed with In-band Mechanism

Establishing the Basis of Managing More SSH Devices

Intel(R) DCM: Energy Director 3.x establishes the basis of managing more SSH devices through the generic SSH connector `com.intel.dcm.plugin.sshPlugin`.

A configuration file provides the rules and details on how devices are identified and managed. It is expected that the configuration file will be updated in the future so that more SSH devices can be managed without the need of upgrading Intel(R) DCM: Energy Director.

Managing Supermicro Multi-Node Systems

Intel(R) DCM: Energy Director 3.x supports managing Supermicro multi-node systems with node manager enabled using the connector `com.intel.dcm.plugin.smcNMPlugin`. When using the connector to manage the servers, some servers may possess the capability of real-time PSU monitoring, which can be used with the power estimation mechanism to monitor the enclosure/chassis level power.

See Also

Power Estimation

Monitoring Data Center

Utilization-based Power Estimation

Intel(R) DCM: Energy Director 3.x provides the function of estimating power consumption dynamically based on the utilization data retrieved from its operating system. Currently the function applies to Microsoft* Windows Servers with WMI exposed, different Linux servers, and some virtual machine managers.

Intel(R) DCM: Energy Director 3.x also provides the function of estimating power consumption dynamically with advanced power model trained from the historical utilization and power data.

See Also

Power Estimation

Managing Microsoft Windows Server through WMI

Managing Linux Servers through SSH

Managing Xen Servers through SSH

Managing VMWare ESX/ESXi Servers through SSH

Advanced Power Modeling

Power and Temperature Statistics

Intel(R) DCM: Energy Director 3.x provides the feature of retrieving historical IT equipment power statistics of physical entities. To start collecting historical power statistics on an entity, call the API `startCollectingPowerDistribution`. To retrieve the statistics, call the API `getPowerDistribution`. Call the API `stopCollectingPowerDistribution` to stop the collection process with the data wiped. For scalability considerations, it is recommended to keep the number of entities below 500 for simultaneous collection of historical power statistics.

Intel(R) DCM: Energy Director 3.x provides the feature of retrieving the current inlet temperature statistics of a physical group by calling the API getInletTemperatureDistribution.

Calculating Cooling Indicator

There are some common ways to get certain indicators on cooling of a server room. Intel(R) DCM: Energy Director 3.x provides the function of calculating how well a

group of servers are cooled against over-heating through the API getCoolingIndicator and calculating how well a group of servers are over-cooled through the API `getOverCoolingIndicator`. Intel(R) DCM: Energy Director 3.x also provides the function of calculating the temperature rise across a group of servers with outlet temperature and airflow monitoring capability through the API `getTemperatureRise`, which can be combined with the temperature difference between the outlet and the inlet of the air conditioners to determine whether hot air recirculation or cold air by-passing exists.

Querying with a Batch of Entities

Intel(R) DCM: Energy Director 3.x provide 2 new APIs, `getBatchQueryData` and `getBatchMetricData`, for querying with a batch of entities to improve the usability of API calls.

Dumping IT Equipment Power Data

Intel(R) DCM: Energy Director 3.x supports dumping IT equipment power data without internal aggregation with the API `dumpMetricData`. The API is similar to `dumpMeasurementData`.

Monitoring Physical Group Power through PDUs

Intel(R) DCM: Energy Director 3.x supports monitoring and reporting IT power consumption of physical groups through PDUs by configuring the entity property `PDU_PWR_AS_IT_EQPMNT_PWR`. After configuring the value of the property as true, Intel(R) DCM: Energy Director takes the value of `PDU_PWR` as IT_EQPMNT_PWR for the physical group, instead of aggregating the values from its direct children.

Monitoring CPU Utilization and Disk I/O for Devices Managed with In-band Mechanism

Intel(R) DCM: Energy Director 3.x provides the mechanism of monitoring CPU utilizations and disk I/O for devices managed with in-band mechanism, e.g., WMI of Microsoft* Windows servers, SSH of different Linux systems. The new query types, `CPU_UTIL` and `DISK_IO`, are used to retrieve the data.

The capability also applies to out-of-band IPMI devices with in-band communication channel attached.

See Also

Managing Microsoft Windows Servers through WMI

Managing Linux Servers through SSH

Managing Xen Servers through SSH

Managing VMWare ESX/ESXi Servers through SSH

Retrieving In-Band Data for Out-of-Band IPMI Devices

New Sampling Frequency & Data Granularity

Intel(R) DCM: Energy Director 3.x provides a new sampling frequency value and a new data granularity value, 300s (5 minutes).

See Also

Sampling Frequency

Measurement Granularity

Reporting inlet temperature with PDU temperature sensors

Intel(R) DCM: Energy Director supports reporting inlet temperature with PDU temperature sensors. By specifying the entity property `REPORT_INLET_TEMP` for PDUs which are able to provide temperature readings in real-time monitoring, the data of average inlet temperature, maximum inlet temperature, and minimum inlet temperature are summarized from the readings of the sensors. The inlet temperature data will be used in aggregation at the group level as well.

See Also

Monitoring the Datacenter: Overview

Real-time Monitoring

Monitoring Enclosure/Chassis Power Based on PSU Power Readings

Leveraging the power estimation mechanism, for certain enclosures/chassis Intel(R) DCM: Energy Director 3.x supports estimating enclosure/chassis power by referring to the PSU power monitored from the nodes inside the enclosures/chassis. Examples are Supermicro multi-node systems with node manager enabled and selected models of Dell microserver chassis hosting Dell C5220 nodes.

To monitor the enclosure/chasis power based on PSU power readings, create a dummy enclosure, specify its entity property `PWR_ESTIMATOR` in the format of

'PSUEstimator:*entity_id*[;*entity_id*[...]]' in which each of the entity IDs refers to a node with the capability of reading power from the corresponding PSU. Conforming with the other cases of power estimation. the enclosure/chassis power can be retrieved through the query type of ESTIMATED_PWR.

See Also

Power Estimation

Control Policies

Power Limit on HP Enclosures, Dell Enclosures Cisco UCS Chassis/Enclosures, and IBM Blades

Intel(R) DCM: Energy Director 3.x provides the function of power limit on HP* enclosures, Dell* enclosures, and Cisco* UCS chassis/enclosures.

It is recommended to configure the static power limit on HP* enclosures, Dell* enclosures, and Cisco* UCS chassis/enclosures with the policy type STATIC_PWR_LIMIT.

For power policies with the types of CUSTOM_PWR_LIMIT, MIN_PWR, and MIN_PWR_ON_INLET_TEMP_TRIGGER, the new entity property GROUP_LIMIT_ON_ENCLOSURE applying to HP* enclosures, Dell enclosures, and Cisco* UCS chassis/enclosures decides whether the power limits are enforced at the enclosure level, or to the blades under the enclosures with the power control capability (that is, for HP devices, the HP* blades managed with the connector com.intel.dcm.plugin.Dcmi10Plugin or com.intel.dcm.plugin.hpiLOPlugin).

By configuring the entity property GROUP_LIMIT_ON_ENCLOSURE with the value of *True*, the enclosures can be included in power policies with the type of CUSTOM_PWR_LIMIT for rack density increase. However, even in that case we recommend to dictate the static power allocation on HP* enclosures within the group power limit by applying additional static power policies.

Intel(R) DCM: Energy Director 3.x also provides the function of power limit on individual IBM* blades.

Power Efficient Policies

Intel(R) DCM: Energy Director 3.x provides a new policy type PWR_EFFICIENT applying to HP* iLO platforms. With the policy applied, the servers are driven to the

P-state with the lowest power and performance without clock throttling, which brings the best power efficiency. It is implemented through configuring HP* Power Regulator in Static Low Power mode, resulting in the best power efficiency below a certain performance level and the best overall power efficiency when the full performance of the processor is not required (cf., *Power Regulator for ProLiant Servers: Technology Brief, 5th Edition*).

NOTE

To enable the capability of configuring power efficient policies on HP* iLO platforms, HP* Power Regulator should not be configured in OS Control Mode. If HP* Power Regulator is configured in OS Control Mode, changing the setting, rebooting the platform, and rediscovering the node in Intel(R) DCM: Energy Director are required.

Bounds of Device Level Power Budget Allocation

Intel(R) DCM: Energy Director 3.x provides optional control knobs on device level power budget allocation in policy calculation for custom power policies. 2 optional entity properties, `MAX_PWR_BUDGET` and `MIN_PWR_BUDGET`, are employed to dictate the upper bound and the lower bound of device level power budget allocation. The power budget allocated to the device will never exceed or drop below the bounds.

Memory Power Limit on Node Manager 2.0 Platforms

Intel(R) DCM: Energy Director 3.x provides a new policy type `MEM_PWR_LIMIT` applies to node entities with the memory power control capability. The policy is used to enforce the power limit on the memory sub-system of a server. Currently the power limit only applies to some of the Node Manager 2.0 platforms.

Others

Knowledge Base of Device Power Profiles

Intel(R) DCM: Energy Director 3.x provides the function of retrieving the knowledge on device power profiles for different device models and configurations through the API `enumerateServerPowerProfiles`, `enumerateNetworkDevicePowerProfiles`, and `enumerateStorageDeviceProfiles`.

Server power profiles contain the information on the typical peak power and active idle power for the servers. Network device and storage device power profiles contain

the information on the typical power of the devices (if available) and the other power information.

Intel(R) DCM: Energy Director 3.x also provides the capability of managing the power profiles so that new profiles can be added and existing profiles can be updated.

NOTE

The original server power profile data comes from the benchmark result page of SPECpower_ssj(R) (http://www.spec.org/power_ssj2008/results/power_ssj2008.html) dated Oct. 23rd, 2012. SPEC(R) and the benchmark name SPECPower_ssj(R) are registered trademarks of the Standard Performance Evaluation Corporation. For more information about SPECPower_ssj, see http://www.spec.org/power_ssj2008/.

The original network device power profile data comes from *Cisco Switch Guide: Scalable, intelligent LAN switching for campus, branch, and data center networks of all sizes* (http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/prod_brochure0900aecd80357ff4.html) dated Nov. 9th, 2012.

The original storage device power profile data comes from the materials below, dated Mar. 31st, 2013:
http://www.emc.com/collateral/specification-sheet/h11340-datadomain-ss.pdf
http://www.emc.com/collateral/hardware/data-sheet/h6802-datadomain-expansion-shelves-ss.pdf
http://www.emc.com/collateral/software/data-sheet/h3454-avamar-data-store-ss.pdf
http://www.emc.com/collateral/hardware/data-sheet/h8939-dlm1000-ds.pdf
http://www.emc.com/collateral/software/specification-sheet/h5937-emc-disk-lbry-mf-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h10690-ss-isilon-s-series.pdf
http://www.emc.com/collateral/software/specification-sheet/h10639-isilon-x-series-ss.pdf
http://www.emc.com/collateral/software/specification-sheet/h10640-isilon-nl-series-ss.pdf

http://www.emc.com/collateral/data-sheet/h11231-ss-isilon-perf-acc.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h10791-isilon-backup-accelerator-ss.pdf
http://www.emc.com/collateral/hardware/data-sheet/h4097-clariion-ax4-ds.pdf
http://www.emc.com/collateral/hardware/specification-sheet/c1147-clariion-cx3-40-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/c1145-clariion-cx3-80-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h5509-emc-clariion-cx4-120-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h5508-emc-clariion-cx4-240-ss.pdf
http://www.epoka.dk/cgi-files/external/pdf/emc_datasheet/clariion%20cx/cx4%20480%20-%20specification%20sheet.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h5506-emc-clariion-cx4-960-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/c1079-clariion-cx500-networkedstorage-systems.pdf
http://www.spectra.com/pdfs/clariiondl.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h8716-symmetrix-vmax-10k-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h6176-symmetrix-vmax-20k-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h10989-symmetrix-vmax-40k-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h7078-vplex-metro-ss.pdf
http://www.emc.com/collateral/software/specification-sheet/h8514-vnx-series-ss.pdf
http://www.emc.com/collateral/hardware/specification-sheet/h8515-vnxe-ss.pdf
http://www.emc.com/collateral/software/data-sheet/h11263-atmos-g3-dense-480-ss.pdf
http://www.emc.com/collateral/software/specification-sheet/h5853-atmos-stor-hrdw-ss.pdf

http://www.netapp.com/us/products/storage-systems/fas2200/fas2200-tech-specs.aspx
http://www.netapp.com/us/products/storage-systems/fas3100/fas3100-tech-specs.aspx
http://www.netapp.com/us/products/storage-systems/fas3200/fas3200-tech-specs.aspx
http://www.netapp.com/us/products/storage-systems/fas6000/fas6000-tech-specs.aspx
http://www.netapp.com/us/products/storage-systems/fas6200/fas6200-tech-specs.aspx
http://www.netapp.com/us/products/storage-systems/e5400/e5400-tech-specs.aspx
http://www.netapp.com/us/products/storage-systems/e2600/e2600-tech-specs.aspx

New Event Type

A new custom event type, IT_EQPMNT_PWR, is defined based on the corresponding metric type.

Identifying Low Utilization Servers

Intel(R) DCM: Energy Director 3.x provides the building block of identifying low utilization servers for power optimization.

In some data centers, there are servers which are under-utilized for most of the time. Typically the workload on the server does not deserve a dedicated physical server and can be consolidated.

With certain heuristics, Intel(R) DCM: Energy Director evaluates whether a server is under-utilized. An API, `evaluateLowUtilizationServer`, is provided to score the server.

Analyzing Current Cooling Status

Intel(R) DCM: Energy Director 3.x provides the building blocks to analyze the current cooling status of a room.

With the visibility of server inlet temperature, datacenter cooling becomes an important area for optimization. One may evaluate the current cooling status of a

room by calling `analyzeCoolingStatus`, in which the current status evaluation, suggestions for optimizations, and benefits after following the suggestions are given.

Advanced Power Modeling

Intel(R) DCM: Energy Director 3.x provides the new function of building advanced power models. The power model for a certain server takes its utilization data as the input to predict its power consumption.

One may use Intel(R) DCM: Energy Director to dump both the utilization data and the power data for servers with power monitoring capability. After feeding the data dumped to build a power model, the model can be used to perform what-if power analysis, that is, to predict the power consumption based on the hypothesized values of resource utilization.

Use the API `addPowerModel` to create a new power model, in which utilization vectors and power data measured are fed. Use the API `predictPower` to predict the power consumption taking a utilization vector (with the same length as that of the vector in creating power model) as the input. The APIs, `enumeratePowerModels` and `removePowerModel`, are provided to manage the power models created.

The models created can be used for dynamic power estimation, if utilization data could be retrieved from the servers automatically.

See Also

Power Estimation

Migrating from Version 2.0 to 3.0

About Migrating from Version 2.0

If you have developed Intel(R) DCM: Energy Director 2.0 based applications, you can migrate your applications so as to use Intel(R) DCM: Energy Director 3.0 new features.

This section provides information on how to migrate your applications from Intel(R) DCM: Energy Director 2.0 based applications to Intel(R) DCM: Energy Director 3.0 based applications.

Before migration, do the following:

1. Read the Intel(R) DCM Developer's Guide to understand Intel(R) DCM: Energy Director 3.0 new features, and new APIs.

2. Consider what components in your applications you want to migrate to use Intel(R) DCM: Energy Director 3.0 new or redefined features.
3. Refer to Intel(R) DCM Installation Guide on how to install or upgrade to Intel(R) DCM: Energy Director 3.0.
4. Read the instructions of each component before modifying your application.
5. Find the difference between Intel(R) DCM: Energy Director 2.0 and 3.0 to better understand the new or redefined interfaces.
6. Refer to the steps in each topic and code samples of each component to implement to your applications.

See Also

Datacenter Hierarchy

- Managing Node Manager 2.0 Platforms
- Managing HP Platforms with DCMI Interface Exposed
- Managing HP/IBM Blades and Enclosures
- Managing IBM Rack Servers
- Managing Dell iDRAC7 Platforms
- Managing Dell iDRAC6 Platforms
- Managing Dell Enclosures
- Managing HP Platforms with LO100 Interface Exposed
- Managing Cisco UCS Devices
- Device Discovery
- Identifying Platforms with Specific Identifiers
- Modeling Unsupported
- New Entity Properties
- Improving API Usability in Modeling Data Center

Monitoring Datacenter

- Monitoring and Controlling PDU Outlets
- Monitoring Instantaneous Power

- Estimating Power
- Monitoring Server Power through PDU Outlet
- Monitoring Airflow and Outlet Temperature
- Aggregating PDU Power on Groups
- Monitoring Real-time PDU/UPS Data
- Estimating Active Idle Power
- Estimated IT Equipment Power
- Improving API Usability in Querying Data

Control Policies

- Adding New Policy Type
- Configuring Number of CPU Cores
- Saving Power with Aid of Job Scheduler and Performance Manager

Managing Events

- Using Notifications as New Event Mechanism
- Adding New Event Type

Data Center Modeling

Managing Node Manager 2.0 Platforms

Intel(R) DCM: Energy Director 3.0 supports managing platforms with Intel(R) Intelligent Power Node Manager 2.0 enabled. To add the platforms, you need to specify the connector `com.intel.dcm.plugin.Nm20Plugin`.

Platforms with Intel(R) Node Manager 2.0 enabled have better power limiting capabilities, for example, larger ranges of effective power limiting, shorter correction time for power bursts, etc.

On some of the platforms, Intel(R) DCM: Energy Director is also able to monitor the power of CPU and memory sub-systems, limit CPU sub-system power, and configure the number of CPU cores to be de-activated after the next reboot.

Managing HP Platforms with DCMI Interface Exposed

Intel(R) DCM: Energy Director 3.0 supports managing HP* platforms with DCMI interface exposed. After updating the firmware of HP* servers with Integrated Lights-Out (iLO) 2 to the latest version, the platforms can be managed by Intel(R) DCM: Energy Director 3.0 as DCMI platforms. HP* servers with iLO3 can be directly managed.

HP* servers may require iLO Advanced Pack Licenses to enable the power capping functionality. Before the licenses are activated on the servers, they are managed by Intel(R) DCM: Energy Director 3.0 as nodes without power capping capability. After the activation, the nodes need to be rediscovered to work with Intel(R) DCM: Energy Director 3.0 on power capping.

Managing HP/IBM Blades and Enclosures

Intel(R) DCM: Energy Director 3.0 supports managing HP* and IBM* blades and enclosures. Enclosures are modeled with a specific type of physical group and are added into Intel(R) DCM: Energy Director as device entities. Individual blade servers are modeled as nodes.

You can use the connectors listed in the table below to add the HP* and IBM* enclosures into Intel(R) DCM: Energy Director.

Connector	Protocol	Device
com.intel.dcm.plugin.ibmAMMPlugin	SSH	IBM* blade enclosures with AMM interface exposed
com.intel.dcm.plugin.ibmBladePlugin	SSH	IBM* blade servers inside an IBM* enclosure
com.intel.dcm.plugin.hpBladeOAPlugin	SSH	HP* blade enclosures with OA interface exposed
com.intel.dcm.plugin.hpBladeServerPlugin	SSH	HP* blade servers inside an HP* enclosure
com.intel.dcm.plugin.Dcmi10Plugin	IPMI	HP* iLO2/iLO3 blades with DCMI interface

		exposed

Management software can use the API getEnclosureAndBladeInfo to retrieve the information of the physical blades under an enclosure to create or update the hierarchy through the regular datacenter modeling APIs. In standalone deployment of Intel(R) DCM: Energy Director, you can call rediscoverEntity on IBM* or HP* enclosure entities so that the hierarchy under the enclosures are built or rebuilt automatically with new blades added, obsolete blades disassociated, and other blades moved in hierarchy.

IBM* enclosures and blades managed through Advanced Management Module (AMM) interface only support average power monitoring with a 10-minute sampling period, they do not support inlet temperature monitoring. Note that due to the platform power monitoring capability, the average periods of power consumption may differ from enclosure to enclosure, therefore they are not aligned with monitoring cycle of Intel(R) DCM: Energy Director.

HP* enclosures managed through Onboard Administrator (OA) interface only support instantaneous power monitoring. When creating enclosure hierarchy automatically, HP* blade entities are added through the OA interface, but not the DCMI interface which enables the capability of average power monitoring. There is no monitoring capability on HP* blades managed through OA interface.

Managing IBM Rack Servers

Intel(R) DCM: Energy Director 3.0 supports managing IBM* rack servers with IPMI 2.0 interface exposed by specifying the connector `com.intel.dcm.plugin.Ipmi20Plugin` when adding entities. Intel(R) DCM: Energy Director monitors the instantaneous power consumption and the inlet temperature of the servers with a certain delay of several seconds on the platform level.

Managing Dell iDRAC7 Platforms

Intel(R) DCM: Energy Director 3.0 supports managing Dell* servers with Dell Remote Access Controller (iDRAC) 7 by specifying the new connector `com.intel.dcm.plugin.dellIdrac7Plugin`. Intel(R) DCM: Energy Director can monitor and limit the power of these platforms.

Managing Dell iDRAC6 Platforms

Intel(R) DCM: Energy Director 3.0 supports managing Dell* servers with Dell Remote Access Controller (iDRAC) 6 by specifying the new connector `com.intel.dcm.plugin.dellIdrac6Plugin` when adding entities. The platforms are managed by Intel(R) DCM: Energy Director 3.0 as nodes with the power monitoring capability and optionally with the power control capability when the servers are rack servers (not blades).

Managing Dell Enclosures

Intel(R) DCM: Energy Director 3.0 supports managing Dell* enclosures through Chassis Management Controller (CMC) with HTTPS/WS-MAN interfaces exposed by specifying the connector `com.intel.dcm.plugin.dellCMCPlugin` when adding entities. However, the entities do not provide functionalities of enclosure power and temperature monitoring, they only act as proxies to collect the blade information under the enclosures through the API getEnclosureAndBladeInfo. The entities do not support automatic creation of hierarchy.

Managing HP Platforms with LO100 Interface Exposed

Intel(R) DCM: Energy Director 3.0 supports managing HP* platforms with Lights-Out 100 (LO100) interface exposed by specifying the connector `com.intel.dcm.plugin.Ipmi20Plugin` when adding the entities. The capability of instantaneous power monitoring is supported on the platforms.

Managing Cisco UCS Devices

Intel(R) DCM: Energy Director 3.0 supports managing Cisco* Unified Computing System (UCS) devices by communicating with Cisco* UCS Managers as an experimental feature. To add Cisco* UCS devices, you need to specify the connector `com.intel.dcm.plugin.ciscoUCSPlugin`, and provide the corresponding addresses and credentials for communication with the Cisco* UCS Manager. The devices managed by the Cisco* UCS Manager are identified with their distinguished name. The capability of instantaneous power monitoring is provided on the platforms.

NOTE

The feature is validated on Cisco* UCS simulators.

Device Discovery

Intel(R) DCM: Energy Director 3.0 provides a set of APIs for management software to discover devices from the network.

API identifyEntity - Use this API to check whether Intel(R) DCM: Energy Director is able to communicate with a device on the network with a specific connector and specific credentials.

API identifyEntityByProtocol - Use this API to specify the protocol and the credentials and determine the appropriate connector for Intel(R) DCM: Energy Director to communicate with the devices.

API discoverEntities - Use this API to specify the IP range, protocol, and credentials to discover the devices supported by Intel(R) DCM: Energy Director.

Identifying Platforms with Specific Identifiers

Intel(R) DCM: Energy Director 3.0 provides new method in identifying the platforms.

When a platform is added into Intel(R) DCM: Energy Director for management, a platform specific identifier is retrieved and stored in database. Intel(R) DCM: Energy Director uses the platform identifier to check whether the entity on the network is the one that Intel(R) DCM: Energy Director expects to manage. The identifier is also used to check whether a duplicate entity is added.

The identifier can be retrieved with the entity property `PLATFORM_ID`.

If a platform is added with the *force* option specified, the identifier remains empty and will not take effect until the communication with the platform is established and its platform identifier gets updated. In this case, it might be possible that the identifiers of the nodes in the instance of Intel(R) DCM: Energy Director conflict. Intel(R) DCM: Energy Director is detects the conflicts periodically. Predefined events with the type `ENTITY_WITH_DUPLICATED_PLATFORMID` are sent out when a conflict is observed.

Modeling Unsupported Devices

Intel(R) DCM: Energy Director 3.0 supports modeling unsupported devices by specifying the connector `com.intel.dcm.plugin.DummyPlugin` when adding entities. The unsupported devices do not have any Intel(R) DCM: Energy Director capabilities. Their de-rated power values are counted when calculating the metrics of IT equipment power (`IT_EQPMNT_PWR`).

New Entity Properties

Intel(R) DCM: Energy Director 3.0 provides several new entity properties:

NODE_PWR_LIMIT reflects the current power limit value of the power policy applied on the node.

CUSTOMIZED_INFO provides the storage of any information associated with the entity from management software (For example, an SOAP message). Intel(R) DCM: Energy Director 3.0 does not use or interpret the value of the property.

DEVICE_TYPE indicates whether the device is a server, a blade, an enclosure or a PDU.

DEVICE_MODEL provides a human readable name for the device defined by Intel(R) DCM: Energy Director.

ASSET_TAG provides a unique identifier of the device defined by the manufacturer. This property requires a tag that is available for retrieval by Intel(R) DCM: Energy Director.

IPMI_CIPHER_SUITE allows management software to specify the required cipher suite used for IPMI communication for enhanced security instead of letting Intel(R) DCM: Energy Director to figure it out automatically.

PWR_ESTIMATOR provides the way to input an estimation of the power consumption for a server without power monitoring capabilities.

Improving API Usability in Modeling Data Center

Intel® DCM: Energy Director 3.0 provides the new APIs for data center modeling with usability improvements.

getPhysicalParent tracks the parent of an entity in the physical hierarchy.

getImmediateLogicalPredecessors tracks the direct predecessors of an entity in the logical view.

enumerateEntitiesWithDetails and findEntitiesWithDetails return the entities along with their properties.

Monitoring Data Center

Monitoring and Controlling PDU Outlets

Intel(R) DCM: Energy Director 3.0 supports monitoring the real-time power status of the outlets of Avocent* and APC* PDUs by calling API `getOutletState`. It also supports controlling power on/off the outlets of the PDUs by calling API `powerOnOutlet` and `powerOffOutlet`.

Monitoring Instantaneous Power

Intel(R) DCM: Energy Director 3.0 an monitor the instantaneous power consumption of devices, including:

- HP* enclosures with Onboard Administrator interface exposed
- IBM* rack servers
- Several PDU models

The power data is exposed through the query type INS_PWR.

Estimating Power

Intel(R) DCM: Energy Director 3.0 enables management software to input the estimated power values for nodes without power monitoring capabilities, therefore, all the power data is stored in a unified data repository. You can query the estimated power values through the query type `ESTIMATED_PWR`.

Monitoring Server Power through PDU Outlet

Intel(R) DCM: Energy Director 3.0 enables monitoring the power of servers without power monitoring capabilities by associating the server with the outlets of the PDUs with outlet power monitoring capability. This feature leverages the power estimation mechanism by inputting PDU outlet estimator in entity property PWR_ESTIMATOR.

Monitoring Airflow and Outlet Temperature

Intel(R) DCM: Energy Director 3.0 supports monitoring the airflow and the outlet temperature on platforms with the corresponding sensors. Group level aggregation is also performed across the platforms with such capabilities.

Aggregating PDU Power on Groups

Intel(R) DCM: Energy Director 3.0 enables aggregating PDU power on groups with two new queries: `PDU_PWR` and `OBSV_MAX_PDU_PWR`. This feature provides historical

power data observed from PDUs associated with the group, and can be used as a reference in power or cooling capacity planning.

Monitoring Real-time PDU/UPS Data

Intel(R) DCM: Energy Director 3.0 enables monitoring the real-time data of PDUs and UPSes, for example, voltage, load, estimated time remaining on the battery, through the new API `getRealTimePduData` and `getRealTimeUpsData`. You can retrieve the data through real-time communication with the device, initiated by the API call.

Monitoring CPU and Memory Power

Intel(R) DCM: Energy Director 3.0 supports monitoring the power of CPU and memory sub-systems on some of the platforms with Intel(R) Node Manager 2.0 enabled. A set of query types are added to expose the data at the node level.

Estimating Active Idle Power and Observing Maximum Power

Intel(R) DCM: Energy Director 3.0 provides estimation of active idle power of a server based on its power history. To retrieve the estimation, call `getActiveIdlePowerEstimation`. To reset the estimation, call `resetActiveIdlePowerEstimation`.

Intel(R) DCM: Energy Director 3.0 also records the peak power reading observed in the power history. To retrieve the peak power value, call `getObservedMaxPower`. To reset the peak power value, call `resetObservedMaxPower`.

Estimating IT Equipment Power

Intel(R) DCM: Energy Director 3.0 provides estimation of the IT equipment power for a group by combining all the information available, including:

- Average power consumption
- Instantaneous power data
- Estimated power of servers without power monitoring capabilities
- De-rated power of servers without power monitoring or power estimation capabilities
- De-rated power of unmanaged equipment

The metric type IT_EQPMNT_PWR along with the other metric types derived from it and DERATED_PWR are redefined in Intel(R) DCM: Energy Director 3.0.

Improving API Usability in Querying Data

Intel(R) DCM: Energy Director 3.0 provides these new APIs to improve API usability in querying data:

`getLatestQueryData` retrieves the most recent data collected.

`dumpMeasurementData` dumps the data available in DCM database without internal aggregation.

The following APIs were enhanced to simplify measurement granularity smaller than 360 seconds:

- `getQueryData`
- `getMetricData`
- `getQueryAggregationPeriodList`
- `getMetricAggregationPeriodList`

These APIs enable alignment with the aggregation period of the query, instead of the previous requirement to align the start time with the 360s boundary. The APIs also produces fine-grain alignments according to the change above.

Control Policies

Adding New Policy Types

Intel(R) DCM: Energy Director 3.0 provides the following new policy types:

- `STATIC_PWR_LIMIT` applies to node entities. By configuring a policy with the new type on a node, the power limit enforced on the node is fixed and will not change even if the node is impacted by other policies with the type of `CUSTOM_PWR_LIMIT`. The feature is useful when power budgets are expected to reserve for nodes with history information on the power demand.
- `CPU_PWR_LIMIT` applies to node entities with the CPU power control capability. The policy is used to enforce the power limit on the CPU sub-system of a server.

Configuring Number of CPU Cores

Intel(R) DCM: Energy Director 3.0 enables configuring the number of CPU cores on some of the Intel(R) Node Manager 2.0 platforms. To activate only the specified number of CPU cores during the next reboot, call `disableCore`.

Saving Power with Aid of Job Scheduler and Performance Manager

Intel(R) DCM: Energy Director 3.0 provides a building block for performance-aware power saving as an experimental feature to co-work with job scheduler and performance manager for power reduction with limited performance impact.

The following APIs enable job scheduler and performance manager to provide the workload intensity and performance information of a certain server to Intel(R) DCM: Energy Director:

- `startPowerSaving`
- `updateWorkloadIntensity`
- `updatePerformanceFeedback`
- `stopPowerSaving`

Intel(R) DCM uses the information to optimize the power limit enforced on the server, reducing power while maintaining the performance level.

NOTE

When you use these APIs, you need to remove any previous power policies set for the entities. If both power saving and other power policies are enabled on the same entities, the behaviors are undefined.

Managing Events

Using Notifications as New Event Mechanism

Intel(R) DCM: Energy Director 3.0 provides a new event mechanism called custom notifications.

Custom notifications are triggered when the evaluation condition passes a certain start, or stop condition. No matter how long the condition is satisfied, only two notifications are sent, one notifies when the start condition passed and the other one notifies when the stop condition passed.

The following APIs are provided for defining and managing custom notifications:

- `defineNotification`

- `getNotificationData`
- `enumerateNotifications`
- `getNotificationState`
- `setNotificationState`
- `removeNotification`

Adding New Event Types

Predefined events with the type ENTITY_CAPABILITIES_CHANGED are sent out when the capability of device changes. For example, when HP* servers with DCMI interface exposed get their iLO Advanced Pack Licenses enabled or disabled, the power control capabilities are changed accordingly.

Predefined events with the type `ENTITY_WITH_DUPLICATED_PLATFORMID` are sent out when a conflict of platform specific identifiers is observed on different entities. For example, some devices are forcibly added into Intel(R) DCM: Energy Director during the network communication failures, the duplicated specific identifiers are observed. For more details on platform specific identifiers, see Identifying Platforms with Specific Identifiers.

A new custom event type TOTAL_AVG_PWR_CAP is defined based on the corresponding query type.

Several new event types are defined for alerts from PDUs and UPSes, for example, PDU_HIGH_LOAD, UPS_LOW_BATTERY.

Failover Solutions

The following lists the three major failovers in Intel(R) DCM: Energy Director:

Application failure

If there are system conflicts, or a power outage, the Intel(R) DCM: Energy Director application on the central server and tier-two server may fail. In this case, all the data in the memory of Intel(R) DCM: Energy Director is lost.

Solution:

You can use a mechanism to detect whether Intel(R) DCM: Energy Director central server or tier-two server is active. If the status is crashed, you can restart the server.

System crash with data available

The Operating System (OS) or Database Management System (DBMS) have crashed, but the related data is still intact, including database file, configuration data and authentication credentials. In this case, there is no data loss, or data replication was conducted.

Solution:

For a central server failure, do the following:

1. Import the database file into a new database.
2. Set up a new central server.
3. Set the database of the central server to point to the database from step 1.
4. Import the configurations and credentials to central server.

For a tier-two server failure, do the following:

1. Import the database file into a new database.
2. Set up a new tier-two server.
3. Set the database of the tier-two server to point to the database from step 1.
4. Import the configurations and credentials to tier-two server.

System crash including data crash

The data is lost when system crashes.

Solution:

For a central server failure:

1. Set up a new central server.

2. Call addEntity to add the existing tier-two servers one by one.

NOTE

- To avoid data loss, you need to back up the database and configuration files.
- To back up and restore the database files, refer to: http://www.postgresql.org/docs/8.3/interactive/backup.html.

For a tier-two server failure:

Option 1. Replace the server

1. Call `setEntityProperties` to update the entity IP of the crashed tier-two server to the IP of the backup server.
2. After you connect the backup server to the central server, Intel(R) DCM: Energy Director automatically synchronizes the backup server with the data in the central server.

Option 2. Move the entity

1. Call `removeEntity` to remove the entity managed by the crashed tier-two server.
2. Call addEntity for each of the entities managed by the crashed to an existing tier-two server.

See Also

Manually Setting Hierarchy

Glossary

A

Action Log: The action log tracks user activities and system events in Intel(R) Data Center Manager.

Application Log: The application log tracks notifications on occurences in Intel(R) Data Center Manager.

B

Baseboard Management Controller (BMC): The BMC is a microcontroller on the node platform. The BMC monitors the system and uses the network to communicate.

C

Control Policy: Specifies power limitations to apply under certain conditions.

Custom Event: An event that you create, based on power consumption or temperature.

D

Derated Power: The nameplate power, reduced by a certain percentage. Designed to be a more realistic estimate of power usage than is the nameplate power.

E

Entity: Either a group or node.

Event: A notification that something has occurred.

Event, Custom: An event that you create, based on power consumption or temperature.

Event, Predefined: An event created by Intel(R) Data Center Manager. Based on system or node occurences, rather than power or thermal activity.

G

Global Property: A configuration property set for all of Intel(R) Data Center Manager.

Group: A collection of nodes or groups.

Group, Logical: A group of nodes and/or groups, defined by any non-physical parameters. For example, a logical group could include all web servers.

Group, Physical: A group of nodes or groups defined by physical characteristics. For example, a physical group could be a rack of nodes or a row of racks.

I

Intelligent Platform Management Interface (IPMI): An out of band interface for platform management. Intel(R) Data Center Manager communicates with nodes through IPMI.

L

Logical Group: A group of nodes and/or groups, defined by any non-physical parameters. For example, a logical group could include all web servers.

M

Managed Node: A node with Intel(R) Intelligent Power Node Manager, managed by Intel(R) Data Center Manager.

N

Nameplate Power: The total maximum power of all platform components. Usually printed on the system.

Node: A single server.

P

Physical Group: A group of nodes or groups defined by physical characteristics. For example, a physical group could be a rack of nodes or a row of racks.

Power Distribution Unit (PDU): A device that distributes electric power. The amount of power a PDU can supply is physically limited.

Predefined Event: An event created by Intel(R) Data Center Manager. Based on system or node occurences, rather than power or thermal activity.

U

Unmanaged Equipment: Power-consuming equipment in the datacenter without power management capabilities.

Unmanaged Node: Nodes without Intel® Intelligent Power Node Manager. Intel(R) Data Center Manager assumes that unmanaged nodes always use the derated power.

Index