System z

Application Programming Interfaces

SB10-7030-16

System z

Application Programming Interfaces

SB10-7030-16

Note:
Before using this information and the product it supports, read the information in "Safety" on page v, Appendix G, "Notices," on page 235, and *IBM Systems Environmental Notices and User Guide*, Z125–5823.

This edition, SB10-7030-16, applies to the IBM System z servers. This edition replaces SB10-7030-15.

There might be a newer version of this document in a **PDF** file available on **Resource Link**. Go to *http://www.ibm.com/servers/resourcelink* and click **Library** on the navigation bar. A newer version is indicated by a lowercase, alphabetic letter following the form number suffix (for example: 00a, 00b, 01a, 01b).

Contents

Safety

Safety notices

Safety notices may be printed throughout this guide. **DANGER** notices warn you of conditions or procedures that can result in death or severe personal injury. **CAUTION** notices warn you of conditions or procedures that can cause personal injury that is neither lethal nor extremely hazardous. **Attention** notices warn you of conditions or procedures that can cause damage to machines, equipment, or programs.

There are no **DANGER** notices in this guide.

World trade safety information

Several countries require the safety information contained in product publications to be presented in their translation. If this requirement applies to your country, a safety information booklet is included in the publications package shipped with the product. The booklet contains the translated safety information with references to the US English source. Before using a US English publication to install, operate, or service this IBM® product, you must first become familiar with the related safety information in the *Systems Safety Notices*, G229-9054. You should also refer to the booklet any time you do not clearly understand any safety information in the US English publications.

Laser safety information

All System z® models can use I/O cards such as FICON®, Open Systems Adapter (OSA), InterSystem Channel-3 (ISC-3), or other I/O features which are fiber optic based and utilize lasers (short wavelength or long wavelength lasers).

Laser compliance

All lasers are certified in the US to conform to the requirements of DHHS 21 CFR Subchapter J for Class 1 or Class 1M laser products. Outside the US, they are certified to be in compliance with IEC 60825 as a Class 1 or Class 1M laser product. Consult the label on each part for laser certification numbers and approval information.

CAUTION: Data processing environments can contain equipment transmitting on system links with laser modules that operate at greater than Class 1 power levels. For this reason, never look into the end of an optical fiber cable or open receptacle. (C027)

CAUTION: This product contains a Class 1M laser. Do not view directly with optical instruments. (C028)

About this publication

This document is intended to assist system management independent software vendors, customers, and system programmers in developing system management applications that provide integrated hardware and software system management solutions using the Console programming interfaces. A knowledge of the console and the C and/or Rexx language is recommended.

Note: Throughout this book, the term "Console" refers to the Hardware Management Console or the Support Element.

The Console is a direct-manipulation object-oriented graphical user interface that provides single point of control and single-system image for hardware elements. The Console provides the customer grouping support, aggregated and individual real-time system status by colors, consolidated hardware messages support, consolidated operating system messages support, consolidated service support, and hardware commands targeted at a single system, multiple systems, or a customer group of systems. Also, the Console is exception based through customizable acceptable statuses per object. The objects the Console currently manages are:

- Central Processing Complexes (CPCs)
- Central Processing Complex Processor Resource/Systems Manager™ (PR/SM™) partitions and/or native mode images (CPC Images)
- Central Processing Complex Coupling Facilities (Coupling Facility CPC Images)
- Customer defined groups of Central Processing Complexes, PR/SM partitions, native mode images, and/or Coupling Facilities.

In addition to providing an end user with the ability to view and manipulate managed objects, the Console also provides **management application programming interfaces (APIs)**. The management APIs provide the ability to get/set the attributes of a Console managed object, issue commands to be performed on a managed object from a local or remote application, receive asynchronous notifications, and generate Simple Network Management Protocol enterprise-specific traps.

In the following pages, the Console programming interfaces are detailed. The four areas to be covered are:

- Console APIs objectives
- Overview of the Console APIs architecture
- Console APIs definition, data structures, and usage
- Console managed object definitions and identifications.

Figures included in this document illustrate concepts and are not necessarily accurate in content, appearance, or specific behavior.

Message event notification

"HwmcaWaitEvent" on page 13 describes the capabilities available for the receipt of asynchronous message event notifications. While message event notifications are provided by all levels of Consoles, not all Consoles provide the capabilities for:

- Registering for only hardware or operating system message event notifications,
- Registering for only nonrefresh message event notifications

These capabilities are available in Consoles for:

- 9674 Coupling Facility EC D98085 or later, and

- 9672 Parallel Enterprise Server EC E12867 or later.

Load command support

"Commands API" on page 21 describes how to use the Commands API to perform a Load. The **HWMCA_LOAD_COMMAND** is available in Hardware Management Consoles with EC level E45976 or later and available on all standalone Support Elements that support APIs.

Hardware message refresh command support

"Commands API" on page 21 describes how to use the Commands API to request refresh events for existing hardware messages to be sent to registered applications. This command is available on all Consoles version 1.4.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Hardware message event data

"HwmcaWaitEvent" on page 13 describes the data provided in a hardware **HWMCA_EVENT_MESSAGES** event. While this event is available from all levels of Consoles, only Consoles version 1.4.0 or later include the following data in these types of events. (To locate the version level installed on your console, look at the title bar on the workplace window.)

- Time stamp of the hardware message,
- List of CPC Images associated with the hardware object generating the hardware message.

Activation profile support

"Reset activation profile object" on page 116 "Image activation profile object" on page 118, and "Load activation profile object" on page 135 describe the *Reset Activation Profile*, *Image Activation Profile*, and *Load Activation Profile* managed objects. The support for these managed objects is available only on Consoles version 1.4.4 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Hardware message delete command support

"Commands API" on page 21 describes how to use the Commands API to request the deletion of existing hardware messages. This command is available on all Consoles version 1.5.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Reset clear command support

"Commands API" on page 21 describes how to use the Commands API to perform a Reset clear of a CPC Image object. This command is available on all Consoles version 1.5.0 or later. To locate the version level installed on your console, look at the title bar on the workplace window.)

Security log event support

"HWMCA_EVENT_SECURITY_EVENT" on page 18 describes the data provided in a HWMCA_EVENT_SECURITY_EVENT event. This event is issued only from Hardware Management Consoles at Version 1.8.2 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Processing weight support

Support for the processing weight value and processing weight capped attributes was added to the CPC Image, Coupling Facility and Image Activation Profile objects on all Consoles version 1.5.1 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Activate CBU command support

"Commands API" on page 21 describes how to use the Command API to perform a real or test Capacity Backup Upgrade (CBU) activation. This command is available on all Consoles version 1.6.2 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.) For additional information about the Activate CBU command, see *Capacity on Demand User's Guide* (available only on the **Resource Link**® web site).

Import/Export profiles support

"Commands API" on page 21 describes how to use the Commands API to import or export profiles. This command is available on all Consoles version 1.6.2 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

External interrupt command support

"Commands API" on page 21 describes how to use the Commands API to perform an external interrupt for a CPC Image object. This command is available on all Consoles Version 1.7.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Reserve command support

"Commands API" on page 21 describes how to use the Commands API to reserve exclusive control of a CPC object. This command is available only on Support Element Consoles at 1.7.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Alert event support

Support for issuing the HWMCA_EVENT_ALERT has been removed. The Support Element Console no longer issues this event.

Object name added to event data

"HwmcaWaitEvent" on page 13 describes the data provided in the various events generated by the Console. While these events have been available for quite some time, additional information is now provided in all events except for the HWMCA_EVENT_NAME_CHANGE event from Consoles version 1.7.3 or later. This new event data consists of the name of the object the event pertains to.

Degrade indicator enhancements

The Degrade Indicator attribute of the Defined CPC object has been enhanced to have some additional values, which are used to identify additional degraded conditions. These additional values could be returned for this attribute from Consoles version 1.8.0 or later.

Partition identifier

Support for the partition identifier attributes was added to the CPC Image and Coupling Facility objects on all Support Element Consoles version 1.8.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

SCSI load/dump support

"Commands API" on page 21 describes how to use the Commands API to perform a SCSI (Small Computer System Interface) Load and SCSI Dump for a CPC Image object. This command is available on all Consoles Version 1.8.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Event qualification

"HwmcaWaitEvent" on page 13 describes the capabilities available for the receipt of asynchronous message event notifications. While message event notifications are provided by all levels of Consoles, not all Consoles provide the capabilities for providing additional qualification information when registering to receive events. These capabilities are available in Consoles Version 1.8.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Shutdown/Restart command support

"Commands API" on page 21 describes how to use the Commands API to shutdown/restart the Console. This command is available only on Consoles at Version 2.9.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

On/Off Capacity on Demand (On/Off CoD) support

Consoles at Version 2.9.1 or later provide the ability to activate, undo, or query information about a On/Off CoD record for a Defined CPC. (To locate the version level installed on your console, look at the title bar on the workplace window.) "Commands API" on page 21 describes how to use the Commands API to perform an Activation or Undo of an On/Off CoD record for a Defined CPC, while "Defined CPC" on page 81 describes the On/Off CoD related attributes for the Defined CPC object.

Important planning information for On/Off CoD API activation can be found in *Capacity on Demand User's Guide* (available only on the **Resource Link** web site).

Integrated Facility for Applications and Integrated Information Processors weight support

Support for the processing weight value and processing weight capped attributes for Integrated Facility for Applications (IFA) processors was added to the CPC Image and Image Activation Profile objects on all Consoles version 2.9.0 or later. Support for the processing weight value and processing weight capped attributes for IBM System z9® Integrated Information Processors (zIIP) was added to the CPC Image and Image Activation Profile objects on all Consoles version 2.9.1 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Processor running time support

Support for the processor running attributes was added to the Defined CPC and Reset Activation Profile objects on all Consoles version 2.9.1 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Group profile support

Group Profile Object, in Chapter 4, "Console application managed objects," on page 75, describes the new support for the Group Profile managed object. An additional attribute used to determine the list of Group Profile objects has also been added to the Defined CPC object as well. This support is available only on Consoles version 2.9.2 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Additional image activation profile attributes

Support for the following attributes was added to the Image Activation Profile objects on all Consoles version 2.9.2 or later:

- Load at activation
- Central storage
- Reserved central storage
- Expanded storage
- Reserved expanded storage
- Number of dedicated general-purpose processors
- Number of reserved dedicated general-purpose processors
- Number of dedicated Integrated Facility for Applications (IFA) processors
- Number of reserved dedicated Integrated Facility for Applications (IFA) processors
- Number of dedicated Integrated Facility for Linux (IFL) processors
- Number of reserved dedicated Integrated Facility for Linux (IFL) processors
- Number of dedicated Internal Coupling Facility (ICF) processors
- Number of reserved dedicated Internal Coupling Facility (ICF) processors
- Number of dedicated Integrated Information Processors (zIIP) processors
- Number of reserved dedicated Integrated Information Processors (zIIP) processors
- Number of shared general-purpose processors
- Number of reserved shared general-purpose processors
- Number of shared Integrated Facility for Applications (IFA) processors
- Number of reserved shared Integrated Facility for Applications (IFA) processors
- Number of shared Integrated Facility for Linux (IFL) processors
- Number of reserved shared Integrated Facility for Linux (IFL) processors
- Number of shared Internal Coupling Facility (ICF) processors
- Number of reserved shared Internal Coupling Facility (ICF) processors
- Number of shared Integrated Information Processors (zIIP) processors
- Number of reserved shared Integrated Information Processors (zIIP) processors

HwmcaGetBulk API

"HwmcaGetBulk" on page 11 describes the new HwmcaGetBulk application programming interface. This new API allows the application program to use the SNMP GetBulk request, which provides a mechanism for getting multiple attributes with a single request. While this API is being introduced with version 2.9.2, most earlier versions of Consoles already support this new request. (To locate the version level installed on your console, look at the title bar on the workplace window.)

SNMP over TCP support

Prior to version 2.9.2, the Data Exchange APIs exclusively used the User Datagram Protocol (UDP) of TCP/IP for the sending of SNMP requests and the receiving of SNMP responses. Consoles version 2.9.2 or later now have support for flowing SNMP requests/responses using the Transmission Control Protocol (TCP) of TCP/IP. Since TCP guarantees reliable delivery, the Data Exchange APIs will automatically attempt to use the TCP protocol first and then fall back to UDP if it is unavailable. Support for using TCP for SNMP is also being made available for earlier Console versions as well. Contact your IBM support representative for details on what microcode levels are needed for this support. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Version support

Support for a new version attribute has been added to the Defined CPC and Console Application objects on all Consoles version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Engineering Change (EC)/Microcode Level (MCL) support

Support for a new attribute that describes the Engineering Change and Microcode levels has been added to the Defined CPC and Console Application objects on all Consoles version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Internet Protocol (IP) addresses support

Support for a new attribute that describes all of the internal protocol (IP) addresses being used has been added to the Defined CPC and Console Application objects on all Consoles version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

z/VM IML/partition activation mode

The IML/Partition Activation mode attribute for CPC Image object supports a new value for when a CPC Image is activated is this newly supported mode. This support is available only on all Consoles version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Disabled wait event support

"HWMCA_EVENT_DISABLED_WAIT" on page 19 describes the data provided in the newly supported HWMCA_EVENT_DISABLED_WAIT event. This event is issued only on Consoles at Version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

No command response event support

"HwmcaWaitEvent" on page 13 describes the capabilities available for the receipt of asynchronous event notifications. While command response event notifications are provided by all levels of Consoles, not all Consoles provide support for the new event mask, HWMCA_EVENT_NO_COMMAND_RESPONSE, which is used to indicate the registering application does not want to receive HWMCA_EVENT_COMMAND_RESPONSE events. This new capability is available in Consoles Version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Temporary capacity support

New support in the form of a new object, new attributes, and new events has been added for temporary capacity support for Defined CPC objects. This support is available only on Consoles version 2.10.0 or later. (To locate the version level installed on your console, look at the title bar on the workplace window.)

Capacity Record Object, in Chapter 4, "Console application managed objects," on page 75 describes the new Capacity Record object and the object's associated attributes. Two new commands, HWMCA_ADD_CAPACITY_COMMAND and HWMCA_REMOVE_CAPACITY_COMMAND are also provided to allow for the addition and removal of temporary capacity for Defined CPC objects. Lastly, two new events are defined, HWMCA_EVENT_CAPACITY_CHANGE and

HWMCA_EVENT_CAPACITY_RECORD_CHANGE, to allow for registered applications to be notified about temporary capacity changes for Defined CPC objects, as well as changes in Capacity Record objects.

IPv6 support

Consoles version 2.10.0 or later fully support Internet Protocol Version 6 (IPv6). To take advantage of this new support, new versions of the build and run-time files are available for platforms that also support IPv6.

Additional data added to HWMCA_EVENT_DATA event

"HWMCA_EVENT_ENDED" on page 17 describes the data provided in this event. Additional information is now provided in this event on Console version 2.10.0 or later. This new event data consists of:

- the reason the console was ended,
- the name of the Console application component that caused the Console to end, and
- the type of shutdown that caused the Console to end.

Integrated Facility for Applications (IFA) are Application Assist Processor (AAP) in newer consoles

On Consoles version 2.10.0 or later, Integrated Facility for Applications (IFA) processors are called Application Assist Processor (AAP) processors.

Additional image activation profile attributes

Support for the following CPU counter and CPU sampling related attributes were added to the Image Activation Profile objects on all Consoles version 2.10.1 or later:

- Basic CPU counter authorization control
- Problem state CPU counter authorization control
- Crypto activity CPU counter authorization control
- Extended CPU counter authorization control
- Coprocessor group CPU counter authorization control
- Basic CPU sampling authorization control

IPL Token attribute for CPC Image object

Support for the IPL token attribute was added to the CPC Image object on all Consoles version 2.10.1 or later.

Server Time Protocol (STP) configuration support

Support for a new attribute that describes the STP configuration has been added to the Defined CPC object on all Consoles version 2.10.1 or later. Also, the following STP commands were added to the Defined CPC object:

- Swap Current Time Server
- Set STP Configuration
- Change STP-only CTN
- Join STP-only CTN
- Leave STP-only CTN

Additional temporary capacity support

Prior to version 2.10.1, only the total number of processors pending activation could be queried via the Data Exchange APIs. Starting in version 2.10.1, support has been added to be able to query the number of processors pending activation by type as well.

Additional image activation profile attributes

Support for the following crypto related attributes were added to the Image Activation Profile objects on all Consoles version 2.10.2 or later:

- Permit DEA key import functions
- Permit AES key import functions

Group Profile capacity support

Support for a new attribute that provides the current capacity value for a group profile has been added to the Image object on all Consoles version 2.11.0 or later.

Alternate subchannel IPL

Specifying an alternate subchannel IPL address to the Load command is supported on consoles version 2.11.1 or later.

Absolute capping

Absolute capping is supported on consoles version 2.12.1 or later.

Revisions

A technical change to the text is indicated by a vertical line to the left of the change.

Accessibility

This publication is in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties using this PDF file you can request a web-based format of this publication. Go to Resource Link at http://www.ibm.com/servers/resourcelink and click **Feedback** from the navigation bar on the left. In the **Comments** input area, state your request, the publication title and number, choose **General comment** as the category and click **Submit**. You can also send an email to reslink@us.ibm.com providing the same information.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. Send your comments by using Resource Link at http://www.ibm.com/servers/resourcelink. Click **Feedback** on the navigation bar on the left. You can also send an email to reslink@us.ibm.com. Be sure to include the name of the book, the form number of the book, the version of the book, if applicable, and the specific location of the text you are commenting on (for example, a page number, table number, or a heading).

Chapter 1. APIs objectives

The purpose of the Console application programming interfaces is to provide an open set of interfaces and a workstation platform for system management application providers. The interfaces provide the capability to use object-based industry-standard programming interfaces instead of building home-grown release specific programs for collecting the hardware information needed to provide an integrated hardware and software system management solution. Figure 1 illustrates the integration of system management applications using the Console application open programming interfaces to provide a single-system image (SSI) and a single point of control (SPOC).

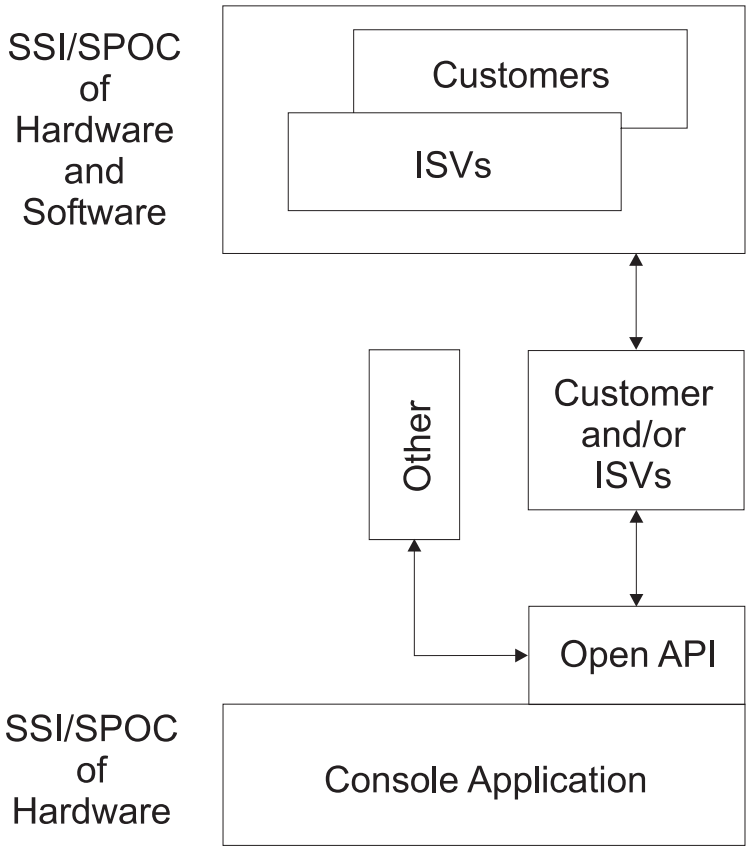

Figure 1. Console APIs Objectives

Chapter 2. Overview

This chapter contains a high-level diagram that illustrates how the Console application accomplishes the purpose of the application programming interfaces, shown in Figure 1 on page 1.

Figure 2 shows a high-level architecture and flow of information for the Console application management programming interfaces. The Console application APIs are implemented using the Simple Network Management Protocol (SNMP) agent. The objects managed by the Console application described in Chapter 4, "Console application managed objects," on page 75 are stored in the Simple Network Management Protocol management information base (MIB). For more information about using the management application programming interfaces, see "Management APIs" on page 5.

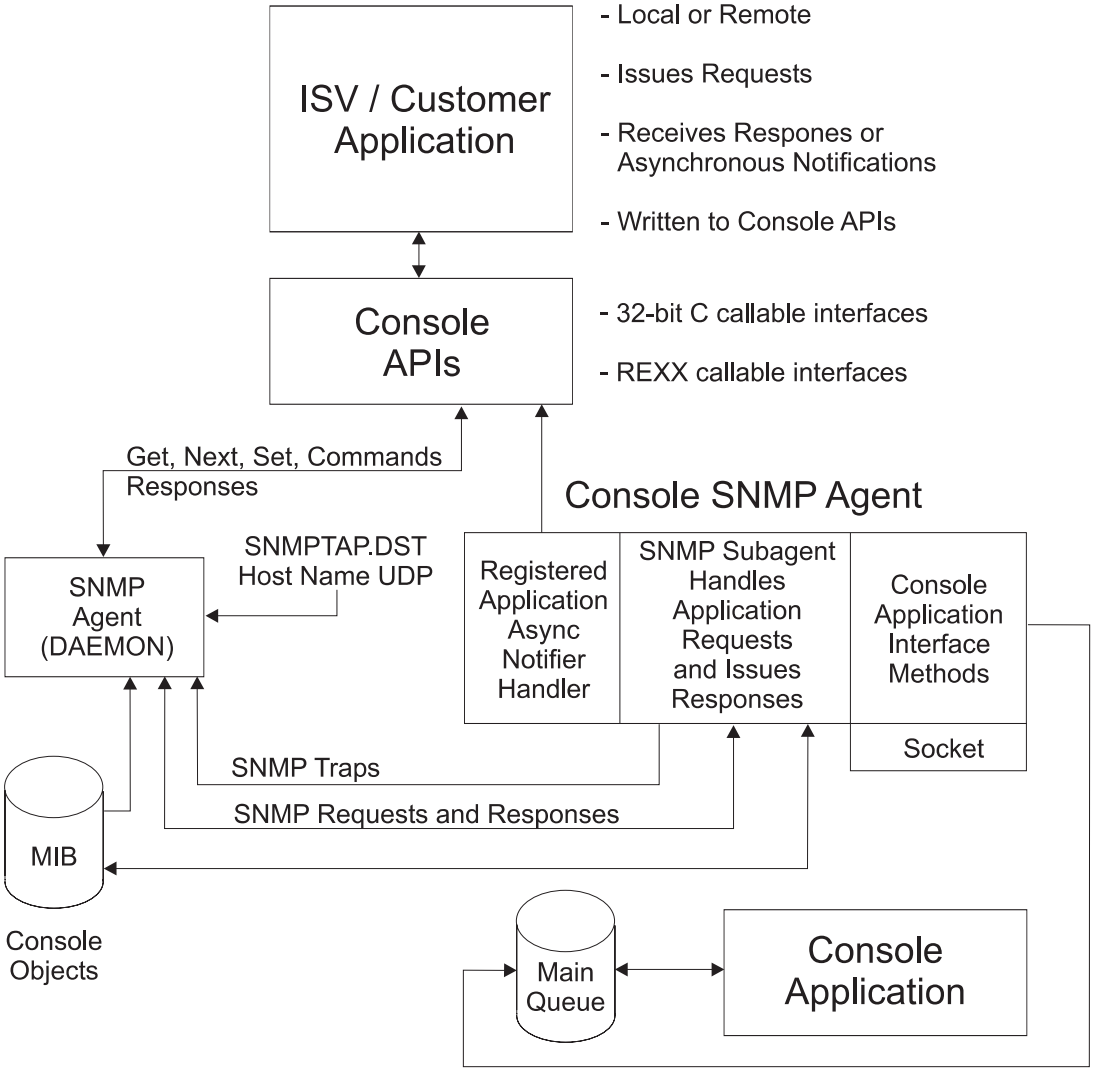

Figure 2. Console Application Data Exchange and Commands APIs

Chapter 3. Console application APIs

Management APIs

Data exchange APIs

The purpose of the Data Exchange APIs is to allow other applications, local or remote, the ability to exchange data related to the objects that the Console application manages. Specifically, this support allows other applications to request the Console application to:

- Query (Get/Get-Next) the attributes of objects,
- Change (Set) certain attributes of objects,
- Receive notification of significant events occurring to objects, and
- Generate enterprise-specific Simple Network Management Protocol traps for significant events occurring to objects.

The Data Exchange APIs use the Simple Network Management Protocol (SNMP) as the transport mechanism. The attributes of objects can be queried/changed through the underlying SNMP Set, Get, Get-Next requests, while event notification is accomplished through the use of the enterprise-specific SNMP Trap message.

Prior to version 2.9.2, the Data Exchange APIs exclusively used the User Datagram Protocol (UDP) of TCP/IP for the sending of SNMP requests and the receiving of SNMP responses. Consoles version 2.9.2 or later now have support for flowing SNMP requests/responses using the Transmission Control Protocol (TCP) of TCP/IP. Since TCP guarantees reliable delivery, the Data Exchange APIs automatically attempt to use the TCP protocol first and then fall back to UDP if it is unavailable.

The underlying SNMP protocol is encapsulated in several APIs in order to reduce the complexities for the application programmer. Specifically, the set of Data Exchange APIs consists of:

Hwmcalnitialize
: Used to perform some initialization tasks necessary for the remainder of the Data Exchange APIs set and the Commands API.

HwmcaGet
: Used to perform a query or Get request for a specified object or object attribute.

HwmcaGetNext
: Used to perform a query-next or Get-next request for an object or object attributes that occurs next in the lexical sequence of objects managed by the Console application.

HwmcaGetBulk
: Used to minimize the number of requests required to retrieve large amounts of object or object attribute data in a manner similar to what could be obtained with a series of HwmcaGetNext calls.

HwmcaSet
: Used to perform a change or Set request for a specified object or object attribute.

HwmcaWaitEvent
: Used to wait for a specified period (or forever) for an event notification from the Console application.

HwmcaTerminate
: Used to perform any cleanup tasks required by any of the other APIs in the set.

HwmcaBuildId
A convenience routine that can be used to construct an object identifier for any object supported by the Console application.

HwmcaBuildAttributeId
A convenience routine that can be used to construct an attribute object identifier for any object supported by the Console applications, based on the object identifier of the object itself.

Note: It is possible that some of these APIs might encounter problems if the Console that they are targeting has been configured to use the *Lockup/Screen saver mode* capability. It is recommended that Consoles used as targets for these APIs not have this feature of OS/2 enabled.

The following pages describe each of these APIs in greater detail.

HwmcaInitialize

Use this API to perform any initialization tasks required in order for the remainder of the API set to function correctly. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to an **HWMCA_INITIALIZE_T** structure. This structure defines all the information that is required for the Console application to perform the initialization request. The fields of the **HWMCA_INITIALIZE_T** structure meaningful are:

pTarget
A pointer to data specifying the target Console application for the initialization request.

This is a pointer to an **HWMCA_SNMP_TARGET_T** structure. The fields of this structure are:

pHost A pointer to a null terminated string specifying the host name or internet address for the target Console application.

szCommunity
A null terminated string specifying the community name that is to be used for the SNMP request made to the target Console application. (Refer to Chapter 6, "Configuring for the data exchange APIs," on page 191 for more information regarding the community name used in SNMP requests.)

ulSecurityVersion
Used to specify the desired authentication method. Use the value **HWMCA_SECURITY_VERSION2** for community name based SNMPv2c authentication. Use the value **HWMCA_SECURITY_VERSION3** for username and password based SNMPv3 authentication.

szUsername
Username to be used for SNMPv3 authentication.

szPassword
Password to be used for SNMPv3 authentication.

ulEventMask
Used to specify the types of event notifications that the application program would like to be registered for. Any combination of the **HWMCA_EVENT_*** constants logically ORed together can be specified. This event mask is used for all events emitted by Console applications managed objects, such as:
- HWMCA_EVENT_COMMAND_RESPONSE
- HWMCA_EVENT_MESSAGE
- HWMCA_EVENT_STATUS_CHANGE
- HWMCA_EVENT_NAME_CHANGE
- HWMCA_EVENT_ACTIVATE_PROF_CHANGE

- HWMCA_EVENT_CREATED
- HWMCA_EVENT_DESTROYED
- HWMCA_EVENT_EXCEPTION_STATE
- HWMCA_EVENT_ENDED
- HWMCA_EVENT_HARDWARE_MESSAGE
- HWMCA_EVENT_OPSYS_MESSAGE
- HWMCA_EVENT_NO_REFRESH_MESSAGE
- HWMCA_EVENT_STARTED
- HWMCA_EVENT_HARDWARE_MESSAGE_ DELETE
- HWMCA_EVENT_SECURITY_EVENT
- HWMCA_EVENT_CAPACITY_CHANGE
- HWMCA_EVENT_CAPACITY_RECORD_ CHANGE
- HWMCA_EVENT_DISABLED_WAIT

These event notifications are sent to all registered applications, independent of whether an application originated the request.

In addition to specifying the types of events that the application program wants to be registered for, this field can also be used to specify some additional options for the Data Exchange APIs. These additional options are:

- HWMCA_DIRECT_INITIALIZE

  By default, the Data Exchange APIs and the Commands API use SNMP when performing the HwmcaInitialize. This flag can be specified to instruct the HwmcaInitialize call to use a proprietary TCP/IP sockets level protocol to perform the HwmcaInitialize, rather than using the SNMP protocol. When this flag is specified it is possible for the HwmcaInitialize to be successful when using a community name that has read only address. When this flag is not used it is required that the community name used for the HwmcaInitialize call has read/write access.

  Note: Specifying this flag is highly recommended when a firewall exists between the Console and the API application. This is because the socket used for the HwmcaInitialize call is also used to send event to the API application. Since this socket connection targets a specific port on the Console (port 3161), it is very straight forward to define a rule in the firewall that allows connections to this port on the Console. If this flag is not specified, the Console attempts to establish a socket connection to a socket created when the API application called the HwmcaInitialize routine. Since the port number for this socket is not fixed, it is very difficult to define a firewall rule to allow this connection from the Console back to the API application.
- HWMCA_FORCE_CLIENT_PATH

  When using the Data Exchange APIs to target a Console with multiple LAN interfaces (for example, a token ring and ethernet interface), this flag can be used to instruct the Console to ensure that all Data Exchange APIs and the Commands API use the targeted internet address when sending and receiving data.
- HWMCA_SNMP_VERSION_2

  By default, the Data Exchange APIs and the Commands API use SNMP version 1. By specifying this flag, the Data Exchange APIs are instructed to used SNMP version 2 as the underlying protocol. The major reason a Data Exchange APIs application would specify this, is so that it can receive more detailed error return codes that are provided by SNMP version 2.
- HWMCA_TOLERATE_LOST_EVENTS

  By default, the HwmcaWaitEvent call terminates the connection to the target console if the API application is unable to process events as fast or faster than the target console is able to send them. By specifying this event mask flag, the connection will not be terminated in this case. Instead, events will not be sent to the API application while it is unable to receive them.

- HWMCA_QUALIFIER_SPECIFIED

  By default event notifications from all Console application managed objects that match the event masks specified in this field will be sent to the API application. By specifying this event mask flag, additional qualification information can be provided to further limit the event notifications that will be sent to the API application. When this event mask flag is specified, the calling API application should also provide additional qualification information in the *ulReserved* field. Refer to the description of the *ulReserved* field for details on how this additional qualification information is specified.
- HWMCA_EVENT_NO_COMMAND_RESPONSE

  By default, all HWMCA_EVENT_COMMAND_RESPONSE events are sent to each registered application. This event mask flag can be used to indicate that the registering application does not want to receive these events.

Note: Care should be used when trying to use the same **HWMCA_INITIALIZE_T** structure for HwmcaWaitEvent calls in addition to the rest of the APIs in the set. Events associated with a **HWMCA_INITIALIZE_T** structure will be queued until retrieved with the HwmcaWaitEvent or until another API, such as HwmcaGet, is called. Therefore, making calls, such as HwmcaGet, will cause any queued events to be discarded and lost.

When both HwmcaWaitEvent and other calls need to be made, an application should perform two HwmcaInitialize calls using two distinct **HWMCA_INITIALIZE_T** structures. The application can then use one of the **HWMCA_INITIALIZE_T** structures for only HwmcaWaitEvent calls and the other **HWMCA_INITIALIZE_T** structure for the other API calls.

ulReserved

This is a reserved field and must be set to zero for the Data Exchange APIs if the **HWMCA_QUALIFIER_SPECIFIED** event mask flag is not specified. If the **HWMCA_QUALIFIER_SPECIFIED** event mask flag is specified, then this field should contain a pointer to an **HWMCA_EVENT_QUALIFIER_T** structure, which is the first of a linked list of additional event qualification information. The fields of the **HWMCA_EVENT_QUALIFIER_T** structures in the list are:

ulEventMask

This field should be set to the event mask flag that is being qualified. Only one event mask flag should be specified in this field. For example, **HWMCA_EVENT_OPSYS_MESSAGE** should be specified when qualifying operating system message event notifications.

ulType

This field is used to indicate the type of event qualification information being provided. The following event qualification types are currently supported.

HWMCA_QUALIFIER_TYPE_NAME

This value is used to indicate that the event qualification data is the null terminated name of the managed object, which is specified in the *type.szName* field of this structure. An **HWMCA_EVENT_QUALIFIER_T** structure that specifies this event qualification type can be used to limit event notifications for the specified event mask to those associated with a managed object with the specified name.

pNext A pointer to the next **HWMCA_EVENT_QUALIFIER_T** structure. A **NULL** is used to indicate that there are no more structures in the linked list.

Once the **HWMCA_INITIALIZE_T** is used on a successful *HwmcaInitialize*, this field should not be altered in any way.

The remainder of the **HWMCA_INITIALIZE_T** structure should be left alone and will be filled in by the *HwmcaInitialize* API. It is important that this structure be left intact and accessible, since it must be passed as a parameter on each of the remaining Data Exchange APIs and Commands API.

In addition to using the **HWMCA_INITIALIZE_T** for any subsequent Data Exchange APIs, it can also be reused on another *HwmcaInitialize* call. The only field that can be changed when doing this is the *ulEventMask* field. By changing this value, an application can change the events notifications that it is registered to receive.

Refer to "Data exchange APIs initialize structure (HWMCA_INITIALIZE_T)" on page 58 for the C declaration of this structure.

ulTimeOut
Used to specify the amount of time that the calling application wants to wait for the *HwmcaInitialize* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *HwmcaInitialize* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the initialization request was successfully delivered and processed by the Hardware Management Console Application. A value of **HWMCA_DE_NO_ERROR** indicates successful completion.

Note: Upon successful completion of the HwmcaInitialize call, the ulEventMask field of the HWMCA_INITIALIZE_T can be checked for the HWMCA_SNMP_USING_TCP flag to determine if the initialized session is using UDP or TCP for the flow of SNMP data.

HwmcaRegister

Use this API to alter the event mask and/or event qualifiers used on a previous HwmcaInitialize call. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to the **HWMCA_INITIALIZE_T** structure that was used on the *HwmcaInitialize* API.

ulEventMask
Used to specify the new types of event notifications that the application program would like to be registered for. Any combination of the **HWMCA_EVENT_*** constants logically ORed together can be specified.

pQualifiers
If the **HWMCA_QUALIFIER_SPECIFIED** event mask flag is specified, then this field should contain a pointer to an **HWMCA_EVENT_QUALIFIER_T** structure, which is the first of a linked list of additional event qualification information.

ulTimeout
Used to specify the amount of time that the calling application wants to wait for the *HwmcaRegister* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *HwmcaRegister* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the register request was successfully delivered and processed by the Hardware Management Console Application. A value of **HWMCA_DE_NO_ERROR** indicates successful completion.

Note: The event mask and event qualifiers specified on the *HwmcaRegister* call will completely replace those in effect from the previous *HwmcaRegister* call.

HwmcaGet

Used to retrieve or Get the data associated with a specific object attribute. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to the **HWMCA_INITIALIZE_T** structure that was used on the *HwmcaInitialize* API.

pszObjectID
A pointer to a null terminated object identifier string for which the data is to be retrieved. Refer to Chapter 4, "Console application managed objects," on page 75 for more information about the object identifiers that the Console application manages.

pOutput
A pointer to an output buffer for the data of the returned object.

ulLength
The size of the output buffer specified by the *pOutput* argument.

pulBytesNeeded
A pointer to an unsigned long integer where the number of total bytes needed for this Get request is returned. If the returned value is greater than that specified in the *ulLength* argument, then the call should be made again, with a larger buffer in order to Get all the object data. If the buffer specified by *pOutput* is too small, then the retrieved object data should not be used, since it is incomplete.

ulTimeout
Used to specify the amount of time that the calling application wants to wait for the *HwmcaGet* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *HwmcaGet* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the retrieve/Get request was successfully delivered and processed by the Console application. A value of **HWMCA_DE_NO_ERROR** indicates successful completion.

Upon successful completion of the *HwmcaGet* API, the output buffer specified by *pOutput* is populated with a series of one or more **HWMCA_DATATYPE_T** structures along with their associated data. The fields of the **HWMCA_DATATYPE_T** structure are:

ucType
Defines the type of data represented by this **HWMCA_DATATYPE_T** structure. Possible values are:

HWMCA_TYPE_INTEGER
Represents a signed number value in host byte order.

HWMCA_TYPE_OCTETSTRING
Represents a null terminated string value.

HWMCA_TYPE_NULL
Used to denote that no value is present.

HWMCA_TYPE_IPADDRESS
Represents a 32- bit internet address in host byte order.

ulLength
Used to specify the length of the data represented by this **HWMCA_DATATYPE_T** structure.

pData A pointer to the actual data that this **HWMCA_DATATYPE_T** structure represents.

pNext A pointer to the next **HWMCA_DATATYPE_T** structure. A **NULL** is used to indicate that there are no more structures in the linked list.

Note: The value stored in the *pulBytesNeeded* field represents the total amount of data returned, while the *ulLength* field of each **HWMCA_DATATYPE_T** structure represents the length of each individual data element in the series.

HwmcaGetNext

Used to retrieve or Get the data associated with the object attribute that occurs next in the lexical sequence of objects, based on a specified object identifier. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.)

The arguments specified for this API are identical to those specified for the HwmcaGet API with two subtle differences.

1. The meaning of the *pszObjectID* argument is used as the base for the Get-Next operation, as opposed to having its object data retrieved.
2. Two **HWMCA_DATATYPE_T** structures and their associated data are returned. The first is the object identifier string for the object whose data is being returned and the second is for the data itself.

HwmcaGetBulk

Used to retrieve or Get data associated with a series of object attributes with a single request. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) This call can be viewed as performing a series of HwmcaGetNext calls with a single request. For additional details about the underlying SNMP GetBulkRequest used by this function refer to Request for Comments (RFC) 3416.

The arguments specified for this API are:

pInitialize
A pointer to the HWMCA_INITIALIZE_T structure that was used on the HwmcaInitialize API.

pszObjectIDs
A pointer to a linked list of HWMCA_DATATYPE_T structures used to specify the object identifiers to use for the GetBulk request. Refer to Chapter 4, "Console application managed objects," on page 75 for more information about the object identifiers that the Console application manages.

nonRepeaters
The number of object identifiers specified in the pszObjectIds argument that are to produce only one HWMCA_DATATYPE_T structure in the output buffer.

maxRepititions
The maximum number of HWMCA_DATATYPE_T fields to be placed in the output buffer for the remaining object identifiers specified in the pszObjectIDs argument.

pOutput
A pointer to an output buffer for the data of the returned object.

uILength
The size of the output buffer specified by the pOutput argument.

pulBytesNeeded
A pointer to an unsigned long integer where the number of total bytes needed for this GetBulk request is returned. If the returned value is greater than that specified in the ulLength argument, then the call should be made again, with a larger buffer in order to get the complete set of object data. If the buffer specified by pOutput is too small, then the retrieved object data should not be used, since it is incomplete.

ulTimeout
Used to specify the amount of time that the calling application wants to wait for the HwmcaGetBulk to complete. This value is specified in milliseconds and the value of HWMCA_INFINITE_WAIT can be used to cause the application to wait forever.

The HwmcaGetBulk API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the request was successfully delivered and processed by the Console application. A value of HWMCA_DE_NO_ERROR indicates successful completion. Upon successful completion of the HwmcaGetBulk API, the output buffer specified by pOutput is populated with a series of one or more HWMCA_DATATYPE_T structures along with their associated data. The fields of the HWMCA_DATATYPE_T structure are:

ucType
Defines the type of data represented by this HWMCA_DATATYPE_T structure. Possible values are:

HWMCA_TYPE_INTEGER
Represents a signed number value in host byte order.

HWMCA_TYPE_OCTETSTRING
Represents a null terminated string value.

HWMCA_TYPE_NULL
Used to denote that no value is present.

HWMCA_TYPE_IPADDRESS
Represents a 32- bit internet address in host byte order.

ulLength
Used to specify the length of the data represented by this HWMCA_DATATYPE_T structure.

pData
A pointer to the actual data that this HWMCA_DATATYPE_T structure represents.

pNext
A pointer to the next HWMCA_DATATYPE_T structure. A NULL is used to indicate that there are no more structures in the linked list.

Note: The value stored in the pulBytesNeeded field represents the total amount of data returned, while the ulLength field of each HWMCA_DATATYPE_T structure represents the length of each individual data element in the series.

HwmcaSet

Used to change or Set the data associated with a specific object attribute. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to the **HWMCA_INITIALIZE_T** structure that was used on the *HwmcaInitialize* API.

pszObjectID
A pointer to a null terminated object identifier string for which the data is to be changed or Set. Refer to Chapter 4, "Console application managed objects," on page 75 for more information about the object identifiers that the Console application manages.

pDataType
A pointer to an **HWMCA_DATATYPE_T** structure that specifies the data to be used for the Set request. The fields of the **HWMCA_DATATYPE_T** structure are:

ucType
Defines the type of data represented by this **HWMCA_DATATYPE_T** structure. Possible values are:

HWMCA_TYPE_INTEGER
Represents a signed number value in host byte order.

Note: The Data Exchange APIs currently only support lengths of 2 bytes or 4 bytes for the HWMCA_TYPE_INTEGER data type when using the HwmcaSet.

HWMCA_TYPE_OCTETSTRING
Represents a null terminated string value.

ulLength
Used to specify the length of the data represented by this **HWMCA_DATATYPE_T** structure.

pData A pointer to the actual data that this **HWMCA_DATATYPE_T** structure represents.

pNext This should be set to NULL for the *HwmcaSet* API and is ignored.

Refer to Chapter 4, "Console application managed objects," on page 75 for a description of the data types, data lengths, and valid data values of the data associated with each type of object managed by the Console application.

ulTimeOut
Used to specify the amount of time that the calling application wants to wait for the *HwmcaSet* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *HwmcaSet* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the change/Set request was successfully delivered and processed by the Console application. A value of **HWMCA_DE_NO_ERROR** indicates successful completion.

HwmcaWaitEvent

Used to wait for event notifications for objects managed by the Console application. The application specifies the types of events that it wants to receive through the use of the *ulEventMask* field of the **HWMCA_INITIALIZE_T** structure that is used on the *HwmcaInitialize* API. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to the **HWMCA_INITIALIZE_T** structure that was used on the *HwmcaInitialize* API.

pOutput
A pointer to an output buffer for the returned event notification data.

ulLength
The size of the output buffer specified by the *pOutput* argument.

pulBytesNeeded
A pointer to an unsigned long integer where the number of total bytes needed for this event notification is returned. If the returned value is greater than that specified in the *ulLength* argument, then the event notification data should not be used, since it is incomplete.

ulTimeOut
Used to specify the amount of time that the calling application wants to wait for an event notification. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *HwmcaWaitEvent* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if any errors occurred while waiting for the event notification. A value of **HWMCA_DE_NO_ERROR** indicates successful completion. A value of **HWMCA_DE_TIMEOUT** indicates that no event notifications were present in the specified timeout period.

Upon successful completion of the *HwmcaWaitEvent* API, the output buffer specified by *pOutput* is populated with a series of one or more **HWMCA_DATATYPE_T** structures along with their associated data. The fields of the **HWMCA_DATATYPE_T** structure are:

ucType
Defines the type of data represented by this **HWMCA_DATATYPE_T** structure. Possible values are:

HWMCA_TYPE_INTEGER
Represents a signed number value in host byte order.

HWMCA_TYPE_OCTETSTRING
Represents a null terminated string value.

HWMCA_TYPE_OBJECTID
Represents a null terminated object identifier string.

ulLength
Used to specify the length of the data represented by this **HWMCA_DATATYPE_T** structure.

pData A pointer to the actual data that this **HWMCA_DATATYPE_T** structure represents.

pNext A pointer to the next **HWMCA_DATATYPE_T** structure. A **NULL** is used to indicate that there are no more structures in the linked list.

Note: The value stored in the *pulBytesNeeded* field represents the total amount of data returned, while the *ulLength* field of each **HWMCA_DATATYPE_T** structure represents the length of each individual data element in the series.

The series of **HWMCA_DATATYPE_T** structures returned from the *HwmcaWaitEvent* API are used to specify:

- An HWMCA_TYPE_OBJECTID that specifies the object identifier of the object that the event notification pertains to
- An HWMCA_TYPE_INTEGER that specifies the event notification type for this event
- Any additional data for the event notification type, as specified below.

The additional data for each of the event notification types are:

HWMCA_EVENT_COMMAND_RESPONSE: Used to notify the application of completion information for a command that has been initiated through the use of the Commands API.

The additional data for this event consists of three object identifier/value pairs that describe the following:

1. An HWMCA_TYPE_OBJECTID that specifies the object identifier of the command for which this command response event has been generated.
2. An HWMCA_TYPE_INTEGER that specifies the return code value to be used to determine the success or failure of the command request that is associated with this command response event.

   Note: Refer to Appendix B, "HWMCA_EVENT_COMMAND_RESPONSE return codes," on page 199 for a list of possible values that can be returned.
3. An HWMCA_TYPE_INTEGER that specifies whether this is the last **HWMCA_EVENT_COMMAND_RESPONSE** event that will be issued for this command. A value of HWMCA_TRUE indicates this event as the last, while a value of HWMCA_FALSE indicates that more **HWMCA_EVENT_COMMAND_RESPONSE** events will be forthcoming.
4. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.
5. An HWMCA_TYPE_OCTETSTRING that specifies the command correlator.

   Note: This field will only be present if the command was invoked using the HwmcaCorrelatedCommand API call.

HWMCA_EVENT_MESSAGE: Used to notify the application that an object managed by the Console application or the Console application itself has a new or refreshed message. This event is generated only for the base objects and not for copies of objects within user-defined groups.

This event is returned to the application when any combination of the following values is used in the *ulEventMask* field of the **HWMCA_INITIALIZE_T** structure:
- HWMCA_EVENT_MESSAGE
- HWMCA_EVENT_HARDWARE_MESSAGE
- HWMCA_EVENT_OPSYS_MESSAGE

If the **HWMCA_EVENT_MESSAGE** value is specified in the *ulEventMask* field of the **HWMCA_INITIALIZE_T** structure, then the application will be notified of both hardware and operating system message events.

If only the **HWMCA_EVENT_HARDWARE_MESSAGE** or **HWMCA_EVENT_OPSYS_MESSAGE** value is specified in the *ulEventMask* field of the **HWMCA_INITIALIZE_T** structure, then the application will be notified only of hardware or operating system message events, respectively.

In addition, the **HWMCA_EVENT_NO_REFRESH_MESSAGE** value can be specified with the above values to control whether the application should be notified of **HWMCA_EVENT_MESSAGE** events for refreshed messages. If the **HWMCA_EVENT_NO_REFRESH_MESSAGE** value is specified in the *ulEventMask* field of the **HWMCA_INITIALIZE_T** structure, then the application will not be notified of **HWMCA_EVENT_MESSAGE** events for refreshed messages.

The additional data for this event can take on two different formats. The format being received can be determined through examining the first object identifier/value pair. The object identifier/value pairs for each of the two formats follows:

An HWMCA_TYPE_INTEGER that specifies whether the message is a hardware or operating system message (HWMCA_HARDWARE_MESSAGE or HWMCA_OPSYS_MESSAGE).

1. The remaining object identifier/value pair for hardware messages is:
   a. An HWMCA_TYPE_OCTETSTRING that specifies the new or refreshed hardware message text.
   b. An HWMCA_TYPE_INTEGER that specifies whether the message is a new (HWMCA_FALSE) or refresh message (HWMCA_TRUE).
   c. An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the new or refreshed hardware message.
   d. An HWMCA_TYPE_OCTETSTRING that specifies the names of the CPC Image object(s) associated with the object that generated the new or refreshed hardware message. This HWMCA_TYPE_OCTETSTRING is a null terminated, blank delimited list of the CPC Image name(s).

      When receiving this event from a Support Element Console, this value contains the name(s) of the CPC Images that are running on the CPC that the Support Element Console is controlling.

      When receiving this event from a Hardware Management Console, this value:
      - Contains no CPC Image names for hardware messages for the Hardware Management Console itself
      - Contains no CPC Image names for Optical Network related hardware messages
      - Contains the name(s) of the CPC Images that are running on the CPC that the hardware message pertains to.
   e. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.
2. The remaining object identifier/value pairs for operating system messages are:
   a. An HWMCA_TYPE_OCTETSTRING that specifies the new or refreshed operating system message text.

Note: If the operating system message text contains multiple lines, then each additional line is delimited from the next line with the character sequence of a carriage return (\r) and a new line (\n).

b. An HWMCA_TYPE_OCTETSTRING that specifies the message identifier of the new operating system message.

c. An HWMCA_TYPE_OCTETSTRING that specifies the date of the new operating system message or an HWMCA_TYPE_NULL indicating that there is no date value for this new operating system message.

d. An HWMCA_TYPE_OCTETSTRING that specifies the time of the new operating system message or an HWMCA_TYPE_NULL indicating that there is no time value for this new operating system message.

e. An HWMCA_TYPE_INTEGER that specifies whether the new operating system message should cause the alarm to be sounded (HWMCA_TRUE or HWMCA_FALSE).

f. An HWMCA_TYPE_INTEGER that specifies whether the new operating system message is a priority message or not (HWMCA_TRUE or HWMCA_FALSE).

g. An HWMCA_TYPE_INTEGER that specifies whether the new operating system message is a held message or not (HWMCA_TRUE or HWMCA_FALSE).

h. An HWMCA_TYPE_OCTETSTRING that specifies the prompt text that should be associated with the new operating system message or an HWMCA_TYPE_NULL indicating that there is no prompt text for this new operating system message.

i. An HWMCA_TYPE_OCTETSTRING that specifies the name of the operating system that generated this new operating system message or an HWMCA_TYPE_NULL indicating that there is no operating system name associated with this new operating system message.

j. An HWMCA_TYPE_INTEGER that specifies whether the message is a new (HWMCA_FALSE) or refresh message (HWMCA_TRUE).

k. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_STATUS_CHANGE: Used to notify the application that an object managed by the Console application has changed status. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of two object identifier/value pairs that describe the following:

1. An HWMCA_TYPE_INTEGER that specifies the new status value
2. An HWMCA_TYPE_INTEGER that specifies the old status value.
3. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_NAME_CHANGE: Used to notify the application that an object managed by the Console application has had a name change. This event notification can be useful when an application retains the object identifiers for objects it is interested in, since the name of an object is used to build the unique portion of the object identifier. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of two object identifier/value pairs that describe the following:

1. An HWMCA_TYPE_OCTETSTRING that specifies the new object name
2. An HWMCA_TYPE_OCTETSTRING that specifies the old object name.

HWMCA_EVENT_ACTIVATE_PROF_CHANGE: Used to notify the application that an object managed by the Console application has changed which activation profile is associated with it.

The additional data for this event consists of two object identifier/value pairs that describe the following:

1. An HWMCA_TYPE_OCTETSTRING that specifies the new activation profile name

2. An HWMCA_TYPE_OCTETSTRING that specifies the old activation profile name.
3. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_CREATED: Used to notify the application that a new object managed by the Console application has been defined or instantiated.

The additional data for this event consists of an object identifier/value pair for an HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_DESTROYED: Used to notify the application that an object managed by the Console application has been undefined.

The additional data for this event consists of an object identifier/value pair for an HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_EXCEPTION_STATE: Used to notify the application that an object managed by the Console application has either entered into or out of an exception state. An object is considered in an exception state when its status is not considered acceptable as defined by the acceptable status attribute of the object. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of two object identifier/value pairs that describe the following:
1. An HWMCA_TYPE_INTEGER that specifies whether the object is entering into an exception state (HWMCA_TRUE) or leaving an exception state (HWMCA_FALSE).
2. An HWMCA_TYPE_INTEGER that specifies the status value for the object.
3. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_STARTED: Used to notify the application that the Console application has started and is now ready to handle Data Exchange APIs and Commands API request.

The additional data for this event consists of an object identifier/value pair for an HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_ENDED: Used to notify the application that the Console application is ending.

The additional data for this event consists of the following object identifier/value pairs:
1. An HWMCA_TYPE_INTEGER that specifies the reason for the event. The possible values are:
   - HWMCA_ENDED_USER - the event was initiated by a user,
   - HWMCA_ENDED_AUTOMATION - the event was initiated by automation, or
   - HWMCA_ENDED_OTHER - the event was initiated by the Console application itself (for example, recovery action, change management, etc.)
2. An HWMCA_TYPE_OCTETSTRING that specifies the name of the Console application component that caused the event.
3. An HWMCA_TYPE_INTEGER that specifies the shutdown type for the event. The possible values are:
   - HWMCA_SHUTDOWN_CONSOLE - the console has been shut down and will take manual intervention to be restarted,
   - HWMCA_RESTART_APPLICATION - the console application has been stopped and will automatically be restarted, or
   - HWMCA_RESTART_CONSOLE - the console has been stopped and will automatically be restarted.
4. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

HWMCA_EVENT_HARDWARE_MESSAGE_DELETE: Used to notify the application that a hardware message associated with an object managed by the Console application or the Console application itself, has been deleted. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of the following object identifier/value pairs:

1. An HWMCA_TYPE_INTEGER that specifies that the message being deleted is a hardware message (HWMCA_HARDWARE_MESSAGE).
2. An HWMCA_TYPE_OCTETSTRING that specifies the message text for the hardware message being deleted.
3. An HWMCA_TYPE_INTEGER that is always set to HWMCA_FALSE for this event.
4. An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the hardware message being deleted.
5. An HWMCA_TYPE_OCTETSTRING that specifies the names of the CPC Image object(s) associated with the object for which the hardware message is being deleted. This HWMCA_TYPE_OCTETSTRING is a null terminated, blank delimited list of the CPC Image name(s).

   When receiving this event from a Support Element Console, this value contains the name(s) of the CPC Images that are running on the CPC that the Support Element Console is controlling.

   When receiving this event from a Hardware Management Console, this value:

   - Contains no CPC Image names for hardware messages for the Hardware Management Console itself
   - Contains no CPC Image names for Optical Network related hardware messages
   - Contains the name(s) of the CPC Images that are running on the CPC that the hardware message pertains to.
6. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

Note: The application should ensure that it provides a buffer that is at least large enough to hold the **HWMCA_DATATYPE_T** structures and associated data for the event notification object identifier and type. A constant, **HWMCA_MIN_EVENT_BUF_SIZE** is provided to the application for this purpose. In addition, another constant, **HWMCA_MAX_EVENT_BUF_SIZE** is provided to the application. This constant can be used to allocate a buffer large enough to hold any event notification. It is important to note that although the **HWMCA_MAX_EVENT_BUF_SIZE** constant can be used to allocate a buffer large enough for any event, it is not intended to indicate a buffer of this size is large enough for all *HwmcaGet* requests.

HWMCA_EVENT_SECURITY_EVENT: Used to notify the application that a security event has been logged.

The additional data for this event consists of the following object identifier/value pairs:

1. An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the security log.
2. An HWMCA_TYPE_OCTETSTRING that specifies the text of the security log.
3. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case the console itself).

HWMCA_EVENT_CAPACITY_CHANGE: Used to notify the application that the processing capacity for a Defined CPC object has changed in some manner. The additional data for this event consists of the following object identifier/value pairs:

1. An HWMCA_TYPE_INTEGER that specifies the type of capacity change that occurred, using one of the following constants:
   - HWMCA_CAPACITY_FENCED_BOOK A processor book has been fenced and is not longer usable.
   - HWMCA_CAPACITY_DEFECTIVE_PROCESSOR A processor has become defective.

- HWMCA_CAPACITY_CONCURRENT_BOOK_REPLACE A concurrent processor book replacement has been performed.
- HWMCA_CAPACITY_CONCURRENT_BOOK_ADD A concurrent processor book addition has been performed.
- HWMCA_CAPACITY_CHECK_STOP A processor has gone into a check stopped state.
- HWMCA_CAPACITY_CHANGES_ALLOWED A user has configured the APIs to be allowed to perform capacity changes.
- HWMCA_CAPACITY_CHANGES_NOT_ALLOWED A user has configured the APIs to no longer be allowed to perform capacity changes.

2. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case a Defined CPC object).

HWMCA_EVENT_CAPACITY_RECORD_CHANGE: Used to notify the application that a change has occurred to a temporary capacity record. The additional data for this event consists of the following object identifier/value pairs:

1. An HWMCA_TYPE_INTEGER that specifies the type of capacity record change that occurred, using one of the following constants:
   - HWMCA_CAPACITY_RECORD_ADD The capacity record has been added to the machine.
   - HWMCA_CAPACITY_RECORD_DELTA The capacity record has been modified.
   - HWMCA_CAPACITY_RECORD_DELETE The capacity record has been deleted.
   - HWMCA_CAPACITY_RECORD_ACCOUNTING
   - HWMCA_CAPACITY_ACTIVATION_LEVEL The capacity record has changed it's level of activation (either more resources from this record have been added or removed from the machine).
   - HWMCA_CAPACITY_PRIORITY_PENDING Additional capacity has been added for the capacity record, with priority, but not enough resources were available to allow for all the capacity specified to be put into effect. As resources become available they will be added for this record in order to completely satisfy the original request for additional capacity.
   - HWMCA_CAPACITY_RECORD_OTHER The capacity record has changed in some other manner.
2. An HWMCA_TYPE_OCTETSTRING for the temporary capacity record identifier that has changed.
3. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case a Defined CPC object).

HWMCA_EVENT_DISABLED_WAIT: Used to notify the application that a CPC Image object has entered a disabled wait state. The additional data for this event consists of the following object identifier/value pairs:

1. An HWMCA_TYPE_OCTETSTRING for the name of the Defined CPC that is associated with the CPC Image that entered a disabled wait state.
2. An HWMCA_TYPE_OCTETSTRING for the disabled wait PSW value.
3. An HWMCA_TYPE_INTEGER for the partition identifier of the CPC Image that entered a disabled wait state.
4. An HWMCA_TYPE_INTEGER for the number of the processor that entered a disabled wait state.
5. An HWMCA_TYPE_OCTETSTRING for the serial number of the Defined CPC that is associated with the CPC Image that entered a disabled wait state.
6. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case a CPC Image object).
7. An HWMCA_TYPE_INTEGER that specifies if the disabled wait event was due to an SCP initiated reset (HWMCA_TRUE) or not (HWMCA_FALSE).

HwmcaTerminate

Used to perform any cleanup tasks required by any of the other APIs. An application should always perform an *HwmcaTerminate* whenever a successful *HwmcaInitialize* has been done after the application

has completed all the activities that are required using the Data Exchange APIs and Commands API. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to the **HWMCA_INITIALIZE_T** structure that was used on the *HwmcaInitialize* API.

ulTimeOut
Used to specify the amount of time that the calling application wants to wait for the *HwmcaTerminate* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *HwmcaTerminate* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the terminate request was successfully delivered and processed by the Console application. A value of **HWMCA_DE_NO_ERROR** indicates successful completion.

Once the *HwmcaTerminate* has been successfully called, the **HWMCA_INITIALIZE_T** structure can then be used for another purpose or freed, depending on the needs of the application.

HwmcaBuildId

A convenience routine to aid the application program in constructing an object identifier for any object supported by the Console. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pszBuffer
A pointer to a buffer where the built object identifier string is to be placed. It is recommended that this buffer be at least **HWMCA_MAX_ID_LEN** bytes in length.

pszPrefix
A pointer to the prefix string to be used for the object identifier to be built. Any of the valid prefixes defined in the Data Exchange APIs include file can be used, such as:
- HWMCA_CONSOLE_ID
- HWMCA_CFG_CPC_GROUP_ID
- HWMCA_CFG_CPC_ID
- HWMCA_CPC_IMAGE_GROUP_ID
- HWMCA_CPC_IMAGE_ID
- HWMCA_GROUPS_GROUP_ID
- HWMCA_GROUPS_OBJECT_ID
- HWMCA_COMMAND_PREFIX
- HWMCA_ACT_RESET_OBJECT_ID
- HWMCA_ACT_IMAGE_OBJECT_ID
- HWMCA_ACT_LOAD_OBJECT_ID
- HWMCA_ACT_GROUP_OBJECT_ID
- HWMCA_CAPACITY_RECORD_OBJECT_ID
- HWMCA_CFG_VM_GROUP_ID
- HWMCA_VM_OBJECT_ID

pszAttribute
A pointer to the attribute suffix string to be used for the object identifier to be built. This can be specified as NULL, when building an identifier for an object itself, as opposed to an attribute object identifier. (Any of the **HWMCA_*_SUFFIX** constants can be specified in this argument.)

pszGroupName
A pointer to the group name to be used for building the object identifier. This can be specified as NULL, when building an object identifier for a predefined group or an object contained within a predefined group.

pszObjectName
A pointer to the object name to be used for building the object identifier. This can be specified as NULL, when building an object identifier for a predefined or user-defined group object.

Note: Refer to "Console application object identifier conventions" on page 75 for more information on the conventions used for the object identifiers for objects managed by the Console.

HwmcaBuildAttributeId

A convenience routine to aid the application program in constructing an attribute object identifier for any object supported by the Console, based on the object identifier of the object itself. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pszBuffer
A pointer to a buffer where the built object identifier string is to be placed. It is recommended that this buffer be at least **HWMCA_MAX_ID_LEN** bytes in length.

pszObjectID
A pointer to the object identifier of the object for which the attribute identifier is to be built.

pszAttribute
A pointer to the attribute suffix string to be used for the object identifier to be built. (Any of the **HWMCA_*_SUFFIX** constants can be specified in this argument.)

Note: Refer to "Console application object identifier conventions" on page 75 for more information on the conventions used for the object identifiers for objects managed by the Console.

Commands API

Allows other applications, local or remote, the ability to execute commands against the objects that the Console application manages. Specifically, this support will allow other applications to request the Console applications to perform the following commands:

- Activate
- Reset Normal
- Reset Clear
- Deactivate
- Send Operating System command
- Start
- Stop
- PSW Restart
- Load
- Hardware Message Refresh
- Hardware Message Delete
- Activate CBU
- Undo CBU
- Import Profile
- Export Profile
- Reserve
- External Interrupt
- SCSI Load
- SCSI Dump
- Shutdown/Restart
- Activate On/Off CoD
- Undo On/Off CoD
- Add Temporary Capacity
- Remove Temporary Capacity
- Swap Current Time Server
- Set STP Configuration

- Change STP-only CTN
- Join STP-only CTN
- Leave STP-only CTN

The Commands API uses the Simple Network Management Protocol (SNMP) as the transport mechanism. The underlying SNMP protocol is encapsulated in the *HwmcaCommand* API in order to reduce the complexities for the application programmer. Refer to following pages for additional information about the *HwmcaCommand*.

HwmcaCommand

Used to perform a command against a specific object managed by the Console. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) The arguments specified for this API are:

pInitialize
A pointer to the **HWMCA_INITIALIZE_T** structure that was used on the *HwmcaInitialize* API.

pszObjectID
A pointer to a null terminated object identifier string for the target object of the command. Refer to Chapter 4, "Console application managed objects," on page 75 for more information about the object identifiers that the Console manages.

pszCommandID
A pointer to a null terminated object identifier string for the object identifier of the command that is to be executed. Valid values for this argument are:
- HWMCA_ACTIVATE_COMMAND
- HWMCA_DEACTIVATE_COMMAND
- HWMCA_RESETNORMAL_COMMAND
- HWMCA_START_COMMAND
- HWMCA_STOP_COMMAND
- HWMCA_PSWRESTART_COMMAND
- HWMCA_SEND_OPSYS_COMMAND
- HWMCA_LOAD_COMMAND
- HWMCA_HW_MESSAGE_REFRESH_COMMAND
- HWMCA_RESETCLEAR_COMMAND
- HWMCA_HW_MESSAGE_DELETE_COMMAND
- HWMCA_ACTIVATE_CBU_COMMAND
- HWMCA_UNDO_CBU_COMMAND
- HWMCA_IMPORT_PROFILE_COMMAND
- HWMCA_EXPORT_PROFILE_COMMAND
- HWMCA_RESERVE_COMMAND
- HWMCA_EXTERNAL_INTERRUPT_COMMAND
- HWMCA_SCSI_LOAD_COMMAND
- HWMCA_SCSI_DUMP_COMMAND
- HWMCA_SHUTDOWN_RESTART_COMMAND
- HWMCA_ACTIVATE_OOCOD_COMMAND
- HWMCA_UNDO_OOCOD_COMMAND
- HWMCA_ADD_CAPACITY_COMMAND
- HWMCA_REMOVE_CAPACITY_COMMAND
- HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND
- HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND
- HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_ COMMAND
- HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND
- HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_ COMMAND

pDatatype
A pointer to a linked list of **HWMCA_DATATYPE_T** structures used to represent the arguments to be passed to the specified command.

The *HwmcaCommand* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the command request was successfully delivered for execution to the Console application. A value of **HWMCA_CMD_NO_ERROR** indicates successful completion.

Once the application determines that the command request has been successfully delivered to the Console, it must wait for one or more **HWMCA_EVENT_COMMAND_RESPONSE** event notification(s) for this command request. This is accomplished through the use of the *HwmcaWaitEvent*. All applications are implicitly registered for this event type. The **HWMCA_EVENT_COMMAND_RESPONSE** event notification will contain:

- Object identifier of the object for which command request was targeted,
- Object identifier for the command that was requested to be executed,
- Return code value that can be used to determine the success or failure of the command request, and
- An indication of whether this event is the last **HWMCA_EVENT_COMMAND_RESPONSE** event notification that should be expected for this command.

Refer to "HwmcaWaitEvent" on page 13 for more details regarding the data returned from the *HwmcaWaitEvent* for the **HWMCA_EVENT_COMMAND_RESPONSE** event notification.

The exceptions to this rule are **HWMCA_HW_MESSAGE_REFRESH_COMMAND** and **HWMCA_HW_MESSAGE_DELETE_COMMAND** commands. There is no need to wait for a **HWMCA_EVENT_COMMAND_RESPONSE** event notification for these commands. These commands are finished once the HwmcaCommand has completed.

HwmcaCorrelatedCommand

Used to perform a command against a specific object managed by the Console. (Refer to "Function prototypes" on page 59 for the C function prototype for this API.) While similar to the *HwmcaCommand* API, this API call is intended to be used to allow the caller to specify some unique correlator data that will then be provided back to the caller as part of the **HWMCA_EVENT_COMMAND_RESPONSE** event, so that the caller can be sure that the event was a result of the command that it requested to be executed. The arguments specified for this API are:

pInitialize
: A pointer to the HWMCA_INITIALIZE_T structure that was used on the HwmcaInitialize API.

pszObjectId
: A pointer to a null terminated object identifier string for the target object of the command. Refer to Chapter 4, "Console application managed objects," on page 75 for more information about the object identifiers that the Console manages.

pszCommandId
: A pointer to a null terminated object identifier string for the object identifier of the command that is to be executed.

pDataType
: A pointer to a linked list of **HWMCA_DATATYPE_T** structures used to represent the arguments to be passed to the specified command.

ulTimeout
: Used to specify the amount of time that the calling application wants to wait for the *HwmcaCorrelatedCommand* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

pCorrelator
: A pointer to the data to be used as a correlator for the specified command.

correlatorSize
: The length of the correlator data.

The *HwmcaCorrelatedCommand* API returns an unsigned long integer return code value to the calling application. This return code lets the calling application know if the command request was successfully delivered for execution to the Console application. A value of **HWMCA_CMD_NO_ERROR** indicates successful completion. Once the application determines that the command request has been successfully delivered to the Console, it must wait for one or more **HWMCA_EVENT_COMMAND_RESPONSE** event notification(s) for this command request. This is accomplished through the use of the *HwmcaWaitEvent*. All applications are implicitly registered for this event type. The **HWMCA_EVENT_COMMAND_RESPONSE** event notification will contain:

- Object identifier of the object for which command request was targeted,
- Object identifier for the command that was requested to be executed,
- Return code value that can be used to determine the success or failure of the command request, and
- An indication of whether this event is the last **HWMCA_EVENT_COMMAND_RESPONSE** event notification that should be expected for this command.
- The command correlator specified when the command was invoked.

Refer to "HwmcaWaitEvent" on page 13 for more details regarding the data returned from the *HwmcaWaitEvent* for the **HWMCA_EVENT_COMMAND_RESPONSE** event notification.

Command arguments

The acceptable and/or required arguments for each command are as follows.

HWMCA_ACTIVATE_COMMAND

No arguments are required, but the following arguments can optionally be specified:

Activation profile name
Name of the activation profile to be used for the Activate command. The default is to use the profile name specified in the Activation profile name attribute for the specified object.

Force indicator
An indicator used to request conditional processing of the Activate command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Either one or both of these arguments can be specified, but they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the **HWMCA_DATATYPE_T** structure used to describe the argument should be specified as follows:

ucType
Should be set to HWMCA_TYPE_NULL.

ulLength
Should be set to zero.

pData A pointer value of zero.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

The default for any argument can be overridden by specifying the HWMCA_DATATYPE_T structure used to describe the argument as follows:

Activation profile name

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the activation profile name (including the null terminator).

pData A pointer to the activation profile name itself.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Force Indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

HWMCA_DEACTIVATE_COMMAND
No arguments are required, but optionally a Force indicator can be specified for the Deactivate command. If this argument is not specified, then the default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is. The fields of the HWMCA_DATATYPE_T structure used to describe the optional Force indicator are:

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

HWMCA_RESETNORMAL_COMMAND
No arguments are required, but the following arguments can optionally be specified:

Force indicator
An indicator used to request conditional processing of the Reset Normal command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE-TRUE) no matter what the state of the target object is.

IPL Token
An IPL token to associate with the Reset Normal command. The default is to not associate an IPL token with the command.

Either one or both of these arguments can be specified, but they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the HWMCA_DATATYPE_T structure used to describe the argument should be specified as follows:

ucType
Should be set to HWMCA_TYPE_NULL.

ulLength
Should be set to zero.

pData A pointer value of zero.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

The default for any argument can be overridden by specifying the HWMCA_DATATYPE_T structure used to describe the argument as follows:

Force Indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

IPL Token

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the IPL token.

pData A pointer to the IPL token itself.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

HWMCA_RESETCLEAR_COMMAND
No arguments are required, but the following arguments can optionally be specified:

Force indicator
An indicator used to request conditional processing of the Reset Clear command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE-TRUE) no matter what the state of the target object is.

IPL Token
An IPL token to associate with the Reset Clear command. The default is to not associate an IPL token with the command.

Either one or both of these arguments can be specified, but they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the HWMCA_DATATYPE_T structure used to describe the argument should be specified as follows:

ucType
Should be set to HWMCA_TYPE_NULL.

ulLength
Should be set to zero.

pData A pointer value of zero.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

The default for any argument can be overridden by specifying the HWMCA_DATATYPE_T structure used to describe the argument as follows:

Force Indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

IPL Token

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the IPL token.

pData A pointer to the IPL token itself.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

HWMCA_START_COMMAND
No arguments are accepted or required.

HWMCA_STOP_COMMAND
No arguments are accepted or required.

HWMCA_PSWRESTART_COMMAND
No arguments are accepted or required.

HWMCA_SEND_OPSYS_COMMAND
This command requires the following two arguments:
- An indication of whether this is a priority operating system command
- The text of the operating system command.

The fields of the **HWMCA_DATATYPE_T** structures used to describe these two arguments are:

Priority Indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for priority operating system commands or HWMCA_FALSE for nonpriority operating system commands.

pNext Should be set to the address of the **HWMCA_DATATYPE_T** structure used to describe the text for the operating system command itself.

Operating System Command Text

ucType
Should be set to HWMCA_TYPE_OCTETSTRING

ulLength
Should be set to the length of the operating system command (including the null terminator).

Note: The operating system command itself should have a length of at least one byte, not including the null terminator.

pData Should be a pointer to the operating system command itself.

pNext Should be set to NULL, since this is the last argument expected for this command.

HWMCA_LOAD_COMMAND

No arguments are required, but the following arguments can optionally be specified:

Load address
Hexadecimal address to be used when performing the Load. The default will be to use the Load address last used when a Load was performed for the object.

Load parameter
Parameter string to be used when performing the Load. The default will be to use the Load parameter last used when a Load was performed for the object.

Clear indicator
Whether or not memory should be cleared before performing the Load. The default is to clear memory before performing the Load.

Timeout
Amount of time (in seconds) to wait for the Load to complete. The default timeout is 60 seconds.

Store status indicator
Whether or not status should be stored before performing the Load. The default is not to store status before performing the Load.

Force indicator
An indicator used to request conditional processing of the Load command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

IPL Token
An IPL token to associate with the Load command. The default is to not associate an IPL token with the command.

Any number of arguments can be specified; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the **HWMCA_DATATYPE_T** structure used to describe the argument should be specified as follows:

ucType
Should be set to HWMCA_TYPE_NULL.

ulLength
Should be set to zero.

pData A pointer value of zero.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

The default for any argument can be overridden by specifying the **HWMCA_DATATYPE_T** structure used to describe the argument as follows:

Load address

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the address string to be used when performing the Load (including the null terminator). This string (including the null terminator) must be less than or equal to 6 characters.

pData Should be a pointer to a field containing the address string to be used when performing the Load. This string must consist of only hexadecimal characters.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

Load parameter

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the parameter string to be used when performing the Load (including the null terminator). This string (including the null terminator) must be less than or equal to nine characters.

pData Should be a pointer to a field containing the parameter string to be used when performing the Load.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

Clear indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for memory to be cleared before performing the Load or HWMCA_FALSE to bypass the clearing of memory before performing the Load.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

Timeout

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the timeout value that is to be used when performing the Load. This value must be between 60 seconds and 600 seconds.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

Store status indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for status to be stored before performing the Load or HWMCA_FALSE to bypass the storing of status before performing the Load.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

Force indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

IPL Token

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the IPL token.

pData A pointer to the IPL token itself.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the **HWMCA_DATATYPE_T** structure used to describe the next argument.

HWMCA_HW_MESSAGE_REFRESH_COMMAND
No arguments are accepted or required.

HWMCA_HW_MESSAGE_DELETE_COMMAND
This command requires the following argument:
- The time stamp of the hardware message.

The fields of the **HWMCA_DATATYPE_T** structure used to describe the time stamp value are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the time stamp (including the null terminator).

pData A pointer to the time stamp string itself.

pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_ACTIVATE_CBU_COMMAND
This command has one required and one optional argument:
- An indicator of whether a real or test CBU activation should be performed is required.
- The password used to validate the CBU activation is optional. If not specified, the password will be obtained automatically from the IBM support system.

The fields of the **HWMCA_DATATYPE_T** structure used to describe these arguments are:

Real/Test Indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for a real CBU activation or HWMCA_FALSE for a test CBU activation.

pNext Should be set to NULL, if this is the last argument to being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Password

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the password (including the null terminator).

pData A pointer to the password string itself.

pNext Should be set to NULL, if this is the last argument expected for this command.

HWMCA_UNDO_CBU_COMMAND
No arguments are accepted or required.

HWMCA_IMPORT_PROFILE_COMMAND
This command requires the following argument:
- The profile area to be imported.

The fields of the **HWMCA_DATATYPE_T** structure used to describe the profile area are:

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData Should be an integer value greater than or equal to 1 and less than or equal to 4, indicating the profile area to be imported.

pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_EXPORT_PROFILE_COMMAND
This command requires the following argument:
- The profile area to be exported.

The fields of the **HWMCA_DATATYPE_T** structure used to describe the profile area are:

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData Should be an integer value greater than or equal to 1 and less than or equal to 4, indicating the profile area to be exported.

pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_RESERVE_COMMAND
Note: This command is available only on a Support Element console. After successfully issuing this command to request the reserve, all API command requests and the majority of other API requests will be blocked, including those from the issuer of the reserve request, until the reserve is released.

This command requires the following arguments:
- An indicator of whether the reserve is being requested or released.
- The name of the application requesting/releasing the reserve (exclusive control).

The fields of the **HWMCA_DATATYPE_T** structure used to describe these two arguments are:

Request/Release Indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE when requesting the reserve or HWMCA_FALSE when releasing the reserve.

pNext Should be set to the address of the HWMCA_DATATYPE_T structure used to describe the text for the application name.

Application Name

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the application name (including the null terminator). The length of this field including the null terminator must be less than or equal to 9 characters.

pData A pointer to the application itself.

pNext Should be set to NULL, since this is the last argument expected for this command.

HWMCA_EXTERNAL_INTERRUPT_COMMAND
This command requires the following argument:

- The number of the processor that is the target of the external interrupt command. This is a number between zero and the maximum number of processors for the target CPC Image object.

The fields of the **HWMCA_DATATYPE_T** structure used to describe the application name are:

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to the processor number.

pNext Should be set to NULL, since this command only accepts one argument.

ulTimeOut
Used to specify the amount of time that the calling application wants to wait for the *HwmcaCommand* to complete. This value is specified in milliseconds and the value of **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

HWMCA_SCSI_LOAD_COMMAND
No arguments are required, but the following arguments can optionally be specified:

Load address
Hexadecimal address to be used when performing the SCSI Load. The default will be to use the Load address last used when a SCSI Load was performed for the object.

Load parameter
Parameter string to be used when performing the SCSI Load. The default will be to use the Load parameter last used when a SCSI Load was performed for the object.

Worldwide port name
The worldwide port name (WWPN) to be used for the SCSI Load. The default will be to use the worldwide port name last used when a SCSI Load was performed for the object.

Logical unit number
The logical unit number (LUN) to be used for the SCSI Load. The default will be to use the logical unit number last used when a SCSI Load was performed for the object.

Boot program selector
The boot program selector to be used for the SCSI Load. The default will be to use the boot program selector last used when a SCSI Load was performed for the object.

Operating system specific load parameters
The operating system specific load parameters to be used for the SCSI Load. The default will be to use the operating system specific load parameters last used when a SCSI Load was performed for the object.

Boot record logical block address
The boot record logical block address to be used for the SCSI Load. The default will be to use the boot record logical block address last used when a SCSI Load was performed for the object.

Force indicator
An indicator used to request conditional processing of the SCSI Load command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Any number of arguments can be specified; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the HWMCA_DATATYPE_T structure used to describe the argument should be specified as follows:

ucType
Should be set to HWMCA_TYPE_NULL.

ulLength
Should be set to zero.

pData
A pointer value of zero.

pNext Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

The default for any argument can be overridden by specifying the HWMCA_DATATYPE_T structure used to describe the argument as follows:

Load address

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the address string to be used when performing the SCSI Load (including the null terminator). This string (including the null terminator) must be less than or equal to 5 characters.

pData Should be a pointer to a field containing the address string to be used when performing the SCSI Load. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Load parameter

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the parameter string to be used when performing the SCSI Load (including the null terminator). This string (including the null terminator) must be less than or equal to 9 characters.

pData Should be a pointer to a field containing the parameter string to be used when performing the SCSI Load.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Worldwide port name

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the worldwide port name string to be used when performing the SCSI Load (including the null terminator). This string (including the null terminator) must be less than or equal to 17 characters.

pData Should be a pointer to a field containing the worldwide port name string to be used when performing the SCSI Load. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Logical unit number

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the logical unit number string to be used when performing the SCSI Load (including the null terminator). This string (including the null terminator) must be less than or equal to 17 characters.

pData Should be a pointer to a field containing the logical unit number string to be used when performing the SCSI Load. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Disk Partition Identifer

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the boot program selector value, which can be in the range 0 - 30, inclusive.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Operating system specific load parameters

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the operating system specific parameters string to be used when performing the SCSI Load (including the null terminator). This string (including the null terminator) must be less than or equal to 257 characters.

pData Should be a pointer to a field containing the operating system specific parameters string to be used when performing the SCSI Load.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Boot record logical block address

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the boot record logical block address string to be used when performing the SCSI Load (including the null terminator). This string (including the null terminator) must be less than or equal to 17 characters.

pData Should be a pointer to a field containing the boot record logical block address string to be used when performing the SCSI Load. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Force indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

HWMCA_SCSI_DUMP_COMMAND
No arguments are required, but the following arguments can optionally be specified:

Load address
Hexadecimal address to be used when performing the SCSI Dump. The default will be to use the Load address last used when a SCSI Dump was performed for the object.

Load parameter
Parameter string to be used when performing the SCSI Dump. The default will be to use the Load parameter last used when a SCSI Dump was performed for the object.

Worldwide port name
The worldwide port name (WWPN) to be used for the SCSI Dump. The default will be to use the worldwide port name last used when a SCSI Dump was performed for the object.

Logical unit number
The logical unit number (LUN) to be used for the SCSI Dump. The default will be to use the logical unit number last used when a SCSI Dump was performed for the object.

Boot program selector
The boot program selector to be used for the SCSI Dump. The default will be to use the boot program selector last used when a SCSI Dump was performed for the object.

Operating system specific load parameters
The operating system specific load parameters to be used for the SCSI Dump. The default will be to use the operating system specific load parameters last used when a SCSI Dump was performed for the object.

Boot record logical block address
The boot record logical block address to be used for the SCSI Dump. The default will be to use the boot record logical block address last used when a SCSI Dump was performed for the object.

Force indicator
An indicator used to request conditional processing of the SCSI Dump command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Any number of arguments can be specified; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the HWMCA_DATATYPE_T structure used to describe the argument should be specified as follows:

ucType
Should be set to HWMCA_TYPE_NULL.

ulLength
Should be set to zero.

pData A pointer value of zero.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

The default for any argument can be overridden by specifying the HWMCA_DATATYPE_T structure used to describe the argument as follows:

Load address

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the address string to be used when performing the SCSI Dump (including the null terminator). This string (including the null terminator) must be less than or equal to 5 characters.

pData Should be a pointer to a field containing the address string to be used when performing the SCSI Dump. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Load parameter

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the parameter string to be used when performing the SCSI Dump (including the null terminator). This string (including the null terminator) must be less than or equal to 9 characters.

pData Should be a pointer to a field containing the parameter string to be used when performing the SCSI Dump.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Worldwide port name

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the worldwide port name string to be used when performing the SCSI Dump (including the null terminator). This string (including the null terminator) must be less than or equal to 17 characters.

pData Should be a pointer to a field containing the worldwide port name string to be used when performing the SCSI Dump. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Logical unit number

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the logical unit number string to be used when performing the SCSI Dump (including the null terminator). This string (including the null terminator) must be less than or equal to 17 characters.

pData Should be a pointer to a field containing the logical unit number string to be used when performing the SCSI Dump. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Disk Partition Identifer

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the boot program selector value, which can be in the range 0 to 30, inclusive.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Operating system specific load parameters

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the operating system specific parameters string to be used when performing the SCSI Dump (including the null terminator). This string (including the null terminator) must be less than or equal to 257 characters.

pData Should be a pointer to a field containing the operating system specific parameters string to be used when performing the SCSI Dump.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Boot record logical block address

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the boot record logical block address string to be used when performing the SCSI Dump (including the null terminator). This string (including the null terminator) must be less than or equal to 17 characters.

pData Should be a pointer to a field containing the boot record logical block address string to be used when performing the SCSI Dump. This string must consist of only hexadecimal characters.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

Force indicator

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext
Should be set to NULL if this is the last argument being specified, or this should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

HWMCA_SHUTDOWN_RESTART_COMMAND
This command requires the following argument:

- An indicator of the type of shutdown or restart to be performed.

The fields of the **HWMCA_DATATYPE_T** structure used to describe this shutdown/restart type are:

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing one of the following values:

- HWMCA_RESTART_APPLICATION - Used to indicate the Console application is to be restarted.

  Note: For Support Element consoles, this value will implicitly cause the Console to be restarted.
- HWMCA_RESTART_CONSOLE - Used to indicate the Console is to be restarted.
- HWMCA_SHUTDOWN_CONSOLE - Used to indicate the Console is to be shutdown/powered off.
- HWMCA_RESTART_APPLICATION_ALTERNAT E - Used to indicate the Alternate Support Element Console application is to be restarted. This option is only valid for the Support Element Console.
- HWMCA_RESTART_CONSOLE_ALTERNATE -Used to indicate the Alternate Support Element Console is to be restarted. This option is only valid for the Support Element Console.

  Note: This value will implicitly cause the Alternate Console to be restarted.
- HWMCA_SHUTDOWN_CONSOLE_ALTERNATE - Used to indicate the Alternate Support Element Console is to be shutdown/powered off. This option is only valid for the Support Element Console.

pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_ACTIVATE_OOCOD_COMMAND

This command requires the following argument:

- The order number of the On/Off Capacity on Demand (On/Off CoD) record to be activated.

  The fields of the **HWMCA_DATATYPE_T** structure used to describe the order number are:

  ucType
  Should be set to HWMCA_TYPE_OCTETSTRING.

  ulLength
  Should be set to length of the order number string (including the null terminator).

  pData A pointer to the string itself.

  pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_UNDO_OOCOD_COMMAND

No arguments are accepted or required.

HWMCA_ADD_CAPACITY_COMMAND

This command, which is used to add temporary capacity to a Defined CPC object, requires the following argument:

- An XML fragment describing the temporary capacity to be added. This XML is used to describe:
  - the identifier of the capacity record to be used,
  - the software model to be used for the capacity addition (optional),
  - the delta processor information to be used for the capacity addition (optional),
  - an indicator for whether the capacity addition is a priority request, (optional, default false), and
  - an indicator for whether the additional capacity is to be added as test or real.

  Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

The fields of the **HWMCA_DATATYPE_T** structure used to describe the capacity information XML are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to length of the capacity information XML string.

pData A pointer to a the capacity information XML string.

pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_REMOVE_CAPACITY_COMMAND
This command, which is used to remove temporary capacity from a Defined CPC object, requires the following argument:

- An XML fragment describing the temporary capacity to be removed. This XML is used to describe:
  - the identifier of the capacity record to be used,
  - the software model to be used for the capacity removal (optional), and
  - the delta processor information to be used for the capacity removal (optional).

  Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

The fields of the HWMCA_DATATYPE_T structure used to describe the capacity information XML are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to length of the capacity information XML string.

pData A pointer to a the capacity information XML string.

pNext Should be set to NULL, since this command only accepts one argument.

HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND
In a configured STP-only Coordinated Timing Network (CTN), one CPC has the role of Current Time Server (CTS). If the CTN has both a Preferred Time Server and a Backup Time Server configured, either one can be the CTS. This command swaps the role of CTS from Preferred Time Server to Backup Time Server or vice versa. The target system must be the system that will become the CTS.

This command requires the following argument:

STP ID
A string representing the current STP identifier for the Defined CPC object.

The fields of the HWMCA_DATATYPE_T structure used to describe the STP ID are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the STP ID. This string (including the null terminator) must be less than or equal to nine characters.

pData A pointer to a field containing the STP ID string.

pNext Should be set to NULL since this command only accepts one argument.

HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND
This command sets the configuration for an STP-only Coordinated Timing Network (CTN). The target system must be the system that will become the Current Time Server (CTS).

This command requires the following arguments:

STP ID

A string representing the current STP identifier for the Defined CPC object. This is used to verify that the CPC is a member of correct CTN.

The fields of the HWMCA_DATATYPE_T structure used to describe the STP ID are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the STP ID. This string (including the null terminator) must be less than or equal to nine characters.

pData A pointer to a field containing the STP ID string.

pNext Should be set to the HWMCA_DATATYPE_T structure used to describe the next argument.

Force Indicator

An indicator used to request conditional processing of the command depending on the state of the target object.

The fields of the HWMCA_DATATYPE_T structure used to describe the Force Indicator are:

ucType
Should be set to HWMCA_TYPE_INTEGER.

ulLength
Should be set to 2.

pData A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

pNext Should point to the HWMCA_DATATYPE_T structure used to describe the next argument.

STP Config XML

An XML fragment describing the configuration for the STP-only CTN. This XML describes:

- the identifier for the STP-only CTN (optional)
- the identity of the CPC to act as Preferred Time Server for the CTN
- the identity of the CPC to act as Backup Time Server for the CTN (optional)
- the identity of the CPC to act as Arbiter for the CTN (optional)
- an indicator of which CPC has the role of Current Time Server (Preferred Time Server or Backup Time Server)

Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

The fields of the HWMCA_DATATYPE_T structure used to describe the STP-only CTN configuration are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the STP Configuration data XML string.

pData A pointer to the STP Configuration data XML string.

pNext Should be set to NULL since this is the last argument accepted by this command.

HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_COMMAND
This command, sent to the Defined CPC with the role of Current Time Server (CTS) in an STP-only Coordinated Timing Network (CTN), changes the STP ID portion of the CTN ID for the entire STP-only CTN.

This command requires the following argument:

STP ID
A string representing the desired STP identifier for the Defined CPC object and all CPCs that are members of the same STP-only CTN.

The fields of the HWMCA_DATATYPE_T structure used to describe the STP ID are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the STP ID. This string (including the null terminator) must be less than or equal to nine characters.

pData A pointer to a field containing the STP ID string.

pNext Should be set to NULL since this command only accepts one argument.

HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND
This command allows a CPC to join an STP-only Coordinated Timing Network (CTN). The target system cannot be the Current Time Server. If the CPC is already participating in an STP-only CTN, it will be removed from that CTN and join the specified one. If the CPC has an ETR ID, it will be removed.

This command requires the following argument:

STP ID
A string representing the STP identifier of the CTN that the Defined CPC object is joining.

The fields of the HWMCA_DATATYPE_T structure used to describe the STP ID are:

ucType
Should be set to HWMCA_TYPE_OCTETSTRING.

ulLength
Should be set to the length of the STP ID. This string (including the null terminator) must be less than or equal to nine characters.

pData A pointer to a field containing the STP ID string.

pNext Should be set to NULL since this command only accepts one argument.

HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_COMMAND
This command removes a CPC from an STP-only Coordinated Timing Network (CTN). The target system cannot be the Current Time Server.

No arguments are accepted or required.

Data exchange APIs and commands API structures and definitions

The following structure and constant definitions can be found in the Data Exchange APIs. The most up to date copy of this code is available on Resource Link at *http://www.ibm.com/servers/resourcelink*. Click **Services**, and then Click **API**.

Constant definitions

```
/*****************************************************************************/
/* Defines for the Console Data Exchange                                     */
/* Return Code Values.                                                       */
/*****************************************************************************/
#define HWMCA_DE_NO_ERROR             0
#define HWMCA_DE_NO_SUCH_OBJECT       1
#define HWMCA_DE_INVALID_DATA_TYPE    2
#define HWMCA_DE_INVALID_DATA_LENGTH  3
#define HWMCA_DE_INVALID_DATA_PTR     4
#define HWMCA_DE_INVALID_DATA_VALUE   5
#define HWMCA_DE_INVALID_INIT_PTR     6
#define HWMCA_DE_INVALID_ID_PTR       7
#define HWMCA_DE_INVALID_BUF_PTR      8
#define HWMCA_DE_INVALID_BUF_SIZE     9
#define HWMCA_DE_INVALID_DATATYPE_PTR 10
#define HWMCA_DE_INVALID_TARGET       11
#define HWMCA_DE_INVALID_EVENT_MASK   12
#define HWMCA_DE_INVALID_PARAMETER    13
#define HWMCA_DE_READ_ONLY_OBJECT     14
#define HWMCA_DE_SNMP_INIT_ERROR      15
#define HWMCA_DE_INVALID_OBJECT_ID    16
#define HWMCA_DE_REQUEST_ALLOC_ERROR  17
#define HWMCA_DE_REQUEST_SEND_ERROR   18
#define HWMCA_DE_TIMEOUT              19
#define HWMCA_DE_REQUEST_RECV_ERROR   20
#define HWMCA_DE_SNMP_ERROR           21
#define HWMCA_DE_INVALID_TIMEOUT      22
#define HWMCA_DE_OBJECT_BUSY          24
#define HWMCA_DE_INVALID_HOST         28
#define HWMCA_DE_INVALID_COMMUNITY    29
#define HWMCA_DE_INVALID_QUALIFIER    30
#define HWMCA_DE_PROTOCOL_ERROR       31
#define HWMCA_DE_INVALID_EVENT_ERROR  32
#define HWMCA_DE_INVALID_STACKNAME    97
#define HWMCA_DE_REQUIRES_QUALIFIER   98
#define HWMCA_DE_TRANSPORT_ERROR      99
```

```
/*****************************************************************************/
/* Defines for the Console Command Return Code Values                        */
/*****************************************************************************/
#define HWMCA_CMD_NO_ERROR                     0
#define HWMCA_CMD_NO_SUCH_OBJECT               1
#define HWMCA_CMD_INVALID_DATA_TYPE            2
#define HWMCA_CMD_INVALID_DATA_LENGTH          3
#define HWMCA_CMD_INVALID_DATA_PTR             4
#define HWMCA_CMD_INVALID_DATA_VALUE           5
#define HWMCA_CMD_INVALID_INIT_PTR             6
#define HWMCA_CMD_INVALID_ID_PTR               7
#define HWMCA_CMD_INVALID_DATATYPE_PTR        10
#define HWMCA_CMD_INVALID_PARAMETER           13
#define HWMCA_CMD_REQUEST_ALLOC_ERROR         17
#define HWMCA_CMD_REQUEST_SEND_ERROR          18
#define HWMCA_CMD_TIMEOUT                     19
#define HWMCA_CMD_REQUEST_RECV_ERROR          20
#define HWMCA_CMD_SNMP_ERROR                  21
#define HWMCA_CMD_INVALID_TIMEOUT             22
#define HWMCA_CMD_INVALID_CMD                 23
#define HWMCA_CMD_OBJECT_BUSY                 24
#define HWMCA_CMD_INVALID_OBJECT              25
#define HWMCA_CMD_COMMAND_FAILED              26
#define HWMCA_CMD_INITTERM_OK                 27
#define HWMCA_CMD_CBU_DISRUPTIVE_OK           28
#define HWMCA_CMD_CBU_PARTIAL_HW              29
#define HWMCA_CMD_CBU_NO_SPARES               30
#define HWMCA_CMD_CBU_TEMPORARY               31
#define HWMCA_CMD_CBU_NOT_ENABLED             32
#define HWMCA_CMD_CBU_NOT AUTHORIZED          33
#define HWMCA_CMD_CBU_FAILED                  34
#define HWMCA_CMD_CBU_ALREADY_ACTIVE          35
#define HWMCA_CMD_CBU_INPROGRESS              36
#define HWMCA_CMD_CBU_CPSAP_SPLIT_CHG         37
#define HWMCA_CMD_INVALID_MACHINE_STATE       38
#define HWMCA_CMD_NO_RECORDID                 39
#define HWMCA_CMD_NO_SW_MODEL                 40
#define HWMCA_CMD_NOT_ENOUGH_RESOURCES        41
#define HWMCA_CMD_NOT_ENOUGH_ACTIVE_RESOURCES 42
#define HWMCA_CMD_ACT_LESS_RESOURCES          43
#define HWMCA_CMD_DEACT_MORE_RESOURCES        44
#define HWMCA_CMD_ACT_TYPE_MISMATCH           45
#define HWMCA_CMD_API_NOT_ALLOWED             46
#define HWMCA_CMD_CDU_IN_PROGRESS             47
#define HWMCA_CMD_MIRRORING_RUNNING           48
#define HWMCA_CMD_COMMUNICATIONS_NOT_ACTIVE   49
#define HWMCA_CMD_RECORD_EXPIRED              50
#define HWMCA_CMD_PARTIAL_CAPACITY            51
#define HWMCA_CMD_INVALID_REQUEST             52
#define HWMCA_CMD_ALREADY_ACTIVE              53
```

```
#define HWMCA_CMD_RESERVE_HELD                          54
#define HWMCA_CMD_GENERAL_XML_PARSING_ERROR             55
#define HWMCA_CMD_STP_NOT_ENABLED                       56
#define HWMCA_CMD_STP_MUST_TARGET_CTS                   57
#define HWMCA_CMD_STP_INVALID_CONFIG_SPECIFIED          58
#define HWMCA_CMD_STP_WRONG_CTN                         59
#define HWMCA_CMD_STP_NOT_VALID_FOR_CTS                 60
#define HWMCA_CMD_STP_IN_ETR_MIGRATION                  61
#define HWMCA_CMD_STP_NODE_NOT_FOUND_IN_SYSTEM_LIST     62
#define HWMCA_CMD_STP_CTNID_TAG_ERROR                   63
#define HWMCA_CMD_STP_NODE_TAG_ERROR                    64
#define HWMCA_CMD_STP_CONFIG_TAG_NOT_FOUND              65
#define HWMCA_CMD_STP_ACTIVE_CTS_TAG_ERROR              66
#define HWMCA_CMD_STP_INITIALIZE_INCOMPLETE             67
#define HWMCA_CMD_STP_INVALID_STP_ID                    68
#define HWMCA_CMD_STP_LINKS_ERROR                       69
#define HWMCA_CMD_STP_REQUIRES_FORCE_TO_CONFIGURE       70


/*****************************************************************************/
/* Defines for the Console Rexx I/F Return Code Value                        */
/*****************************************************************************/
#define HWMCA_RX_INVALID_STEM_VAR      1000

/*****************************************************************************/
/* Miscellaneous defines for the Console APIs.                               */
/*****************************************************************************/
#define HWMCA_INFINITE_WAIT        -1
#define HWMCA_MAX_ID_LEN           80
#define HWMCA_MAX_COMMUNITY_LEN    16
#define HWMCA_MIN_EVENT_BUF_SIZE   ((sizeof(HWMCA_DATATYPE_T)*2)+4+HWMCA_MAX_ID_LEN)
#define HWMCA_MAX_EVENT_BUF_SIZE   (HWMCA_MIN_EVENT_BUF_SIZE+4+9+8+9+4+4+4+9+4+4096+\
                                    (((sizeof(HWMCA_DATATYPE_T)*2)+HWMCA_MAX_ID_LEN)*11))
#define HWMCA_TRUE                 1
#define HWMCA_FALSE                0
#define HWMCA_API_PORT             3161

/*****************************************************************************/
/* Defines for the Console Object Data Types.                                */
/*****************************************************************************/
#define HWMCA_TYPE_SEQUENCE     0x30
#define HWMCA_TYPE_INTEGER      0x02
#define HWMCA_TYPE_OCTETSTRING  0x04
#define HWMCA_TYPE_NULL         0x05
#define HWMCA_TYPE_OBJECTID     0x06
#define HWMCA_TYPE_IPADDRESS    0x40
#define HWMCA_TYPE_COUNTER      0x41
#define HWMCA_TYPE_GAUGE        0x42
#define HWMCA_TYPE_TIMETICKS    0x43
```

```
/*****************************************************************************/
/* Defines for the Console Event Notification Types.                         */
/*****************************************************************************/
#define HWMCA_EVENT_COMMAND_RESPONSE              0x00000000
#define HWMCA_EVENT_MESSAGE                       0x00000001
#define HWMCA_EVENT_STATUS_CHANGE                 0x00000002
#define HWMCA_EVENT_NAME_CHANGE                   0x00000004
#define HWMCA_EVENT_ACTIVATE_PROF_CHANGE          0x00000008
#define HWMCA_EVENT_CREATED                       0x00000010
#define HWMCA_EVENT_DESTROYED                     0x00000020
#define HWMCA_EVENT_EXCEPTION_STATE               0x00000040
#define HWMCA_EVENT_ENDED                         0x00000080
#define HWMCA_EVENT_HARDWARE_MESSAGE              0x00000100
#define HWMCA_EVENT_OPSYS_MESSAGE                 0x00000200
#define HWMCA_EVENT_NO_REFRESH_MESSAGE            0x00000400
#define HWMCA_EVENT_STARTED                       0x00000800
#define HWMCA_EVENT_HARDWARE_MESSAGE_DELETE       0x00001000
#define HWMCA_EVENT_SECURITY_EVENT                0x00004000
#define HWMCA_EVENT_CAPACITY_CHANGE               0x00008000
#define HWMCA_EVENT_CAPACITY_RECORD_CHANGE        0x00010000
#define HWMCA_EVENT_DISABLED_WAIT                 0x00040000
#define HWMCA_EVENT_ALL_EVENTS                    0x0005FFFF
#define HWMCA_DIRECT_INITIALIZE                   0x20000000
#define HWMCA_FORCE_CLIENT_PATH                   0x10000000
#define HWMCA_SNMP_VERSION_2                      0X08000000
#define HWMCA_TOLERATE_LOST_EVENTS                0X02000000
#define HWMCA_QUALIFIER_SPECIFIED                 0x00800000
#define HWMCA_SNMP_USING_TCP                      0x00400000
#define HWMCA_NO_EVENTS                           0x00200000
#define HWMCA_RESEND_OPSYS_MESSAGES               0x00100000
#define HWMCA_EVENT_NO_COMMAND_RESPONSE           0x00020000


/*****************************************************************************/
/* Defines for the Console Static Object IDs.                                */
/*****************************************************************************/
#define HWMCA_OBJECT_PREFIX            "1.3.6.1.4.1.2.6.42."
#define HWMCA_CONSOLE_ID               "1.3.6.1.4.1.2.6.42.0"   /* .x.x       */
#define HWMCA_CFG_CPC_GROUP_ID         "1.3.6.1.4.1.2.6.42.1"   /* .x.x       */
#define HWMCA_CFG_CPC_ID               "1.3.6.1.4.1.2.6.42.1.0" /* .x.x.*     */
#define HWMCA_CPC_IMAGE_GROUP_ID       "1.3.6.1.4.1.2.6.42.2"   /* .x.x       */
#define HWMCA_CPC_IMAGE_ID             "1.3.6.1.4.1.2.6.42.2.0" /* .x.x.*     */
#define HWMCA_GROUPS_GROUP_ID          "1.3.6.1.4.1.2.6.42.3"   /* .x.x.*     */
#define HWMCA_GROUPS_OBJECT_ID         "1.3.6.1.4.1.2.6.42.3.0" /* .x.x.*.*   */
#define HWMCA_COMMAND_PREFIX           "1.3.6.1.4.1.2.6.42.4."
#define HWMCA_ACT_RESET_OBJECT_ID      "1.3.6.1.4.1.2.6.42.5.0" /* .x.x.*.* */
#define HWMCA_ACT_IMAGE_OBJECT_ID      "1.3.6.1.4.1.2.6.42.6.0" /* .x.x.*.* */
#define HWMCA_ACT_LOAD_OBJECT_ID       "1.3.6.1.4.1.2.6.42.7.0" /* .x.x.*.* */
#define HWMCA_ACT_GROUP_OBJECT_ID      "1.3.6.1.4.1.2.6.42.8.0" /* .x.x.*.* */
#define HWMCA_CAPACITY_RECORD_OBJECT_ID "1.3.6.1.4.1.2.6.42.9.0"  /* .x.x.*.* */
#define HWMCA_CFG_VM_GROUP_ID           "1.3.6.1.4.1.2.6.42.10"   /* .x.x */
#define HWMCA_VM_OBJECT_ID              "1.3.6.1.4.1.2.6.42.10.0" /* .x.x.* */
```

```
/*****************************************************************************/
/* Defines for the Hardware Management Console Object Attribute ID suffix    */
/*****************************************************************************/
#define HWMCA_COMMAND_OBJECT_ID_SUFFIX      "0.1"
#define HWMCA_COMMAND_CONDITION_CODE_SUFFIX "0.2"
#define HWMCA_COMMAND_LAST_INDICATOR_SUFFIX "0.3"
#define HWMCA_ENDED_REASON_SUFFIX           "0.4"
#define HWMCA_ENDED_COMPONENT_SUFFIX        "0.5"
#define HWMCA_ENDED_TYPE_SUFFIX             "0.6"
#define HWMCA_COMMAND_CORRELATOR_SUFFIX     "0.7"
#define HWMCA_NAME_SUFFIX                   "1.0"
#define HWMCA_PARENT_NAME_SUFFIX            "2.0"
#define HWMCA_OPSYS_NAME_SUFFIX             "3.0"
#define HWMCA_OPSYS_TYPE_SUFFIX             "4.0"
#define HWMCA_OPSYS_LEVEL_SUFFIX            "5.0"
#define HWMCA_SYSPLEX_NAME_SUFFIX           "6.0"
#define HWMCA_STATUS_ERROR_SUFFIX           "7.0"
#define HWMCA_BUSY_SUFFIX                   "8.0"
#define HWMCA_MESSAGE_SUFFIX                "9.0"
#define HWMCA_MESSAGE_TYPE_SUFFIX           "9.1"
#define HWMCA_MESSAGE_TEXT_SUFFIX           "9.2"
#define HWMCA_MESSAGE_MSG_ID_SUFFIX         "9.3"
#define HWMCA_MESSAGE_DATE_SUFFIX           "9.4"
#define HWMCA_MESSAGE_TIME_SUFFIX           "9.5"
#define HWMCA_MESSAGE_ALARM_SUFFIX          "9.6"
#define HWMCA_MESSAGE_PRIORITY_SUFFIX       "9.7"
#define HWMCA_MESSAGE_HELD_SUFFIX           "9.8"
#define HWMCA_MESSAGE_PROMPT_TEXT_SUFFIX    "9.9"
#define HWMCA_MESSAGE_OSNAME_TEXT_SUFFIX    "9.10"
#define HWMCA_MESSAGE_REFRESH_SUFFIX        "9.11"
#define HWMCA_MESSAGE_TIMESTAMP             "9.12"
#define HWMCA_MESSAGE_IMAGE_LIST            "9.13"
#define HWMCA_STATUS_SUFFIX                 "10.0"
#define HWMCA_EXPECTED_STATUS_SUFFIX        "11.0"
#define HWMCA_IMLMODE_SUFFIX                "12.0"
#define HWMCA_ACTIVATION_PROFILE_SUFFIX     "13.0"
#define HWMCA_LAST_ACT_PROFILE_SUFFIX       "14.0"
#define HWMCA_IP_ADDRESS_SUFFIX             "15.0"
#define HWMCA_SNA_ADDRESS_SUFFIX            "16.0"
#define HWMCA_MODEL_SUFFIX                  "17.0"
#define HWMCA_TYPE_SUFFIX                   "18.0"
#define HWMCA_MACHINE_SERIAL_SUFFIX         "19.0"
#define HWMCA_CPC_SERIAL_SUFFIX             "20.0"
#define HWMCA_CPC_ID_SUFFIX                 "21.0"
#define HWMCA_OBJECT_TYPE_SUFFIX            "22.0"
#define HWMCA_GROUP_CONTENTS_SUFFIX         "23.0"
#define HWMCA_ACT_RESET_LIST_SUFFIX         "24.0"
#define HWMCA_ACT_IMAGE_LIST_SUFFIX         "25.0"
#define HWMCA_ACT_LOAD_LIST_SUFFIX          "26.0"
#define HWMCA_ACT_PROFILE_IOCDS_SUFFIX      "27.0"
#define HWMCA_ACT_PROFILE_IPLADDR_SUFFIX    "28.0"
#define HWMCA_ACT_PROFILE_IPLPARM_SUFFIX    "29.0"
#define HWMCA_WEIGHT_SUFFIX                 "30.0"
#define HWMCA_CAPPED_SUFFIX                 "31.0"
#define HWMCA_CBU_INSTALLED                 "32.0"
#define HWMCA_CBU_ACTIVATED                 "33.0"
#define HWMCA_CBU_ACTIVATION_DATE           "34.0"
#define HWMCA_CBU_EXPIRATION_DATE           "35.0"
#define HWMCA_NUMBER_CBU_TEST_LEFT          "36.0"
#define HWMCA_REAL_CBU_ACTIVATION_AVAILABLE "37.0"
#define HWMCA_MINIMUM_WEIGHT_SUFFIX         "38.0"
#define HWMCA_MAXIMUM_WEIGHT_SUFFIX         "39.0"
#define HWMCA_WLM_MANAGED_SUFFIX            "40.0"
#define HWMCA_CURRENT_WEIGHT_SUFFIX         "41.0"
#define HWMCA_CURRENT_CAPPED_SUFFIX         "42.0"
#define HWMCA_WORK_LOAD_UNITS_SUFFIX        "43.0"
#define HWMCA_RESERVE_ID_SUFFIX             "44.0"
#define HWMCA_ALERT_SUFFIX                  "45.0"
#define HWMCA_SERVICE_REQUIRED_SUFFIX       "46.0"
```

```
#define HWMCA_ALERT_SUFFIX                  "45.0"
#define HWMCA_SERVICE_REQUIRED_SUFFIX       "46.0"
#define HWMCA_DEGRADED_SUFFIX               "47.0"
#define HWMCA_CBU_ENABLED_SUFFIX            "48.0"
#define HWMCA_CLUSTER_NAME_SUFFIX           "49.0"
#define HWMCA_CLUSTER_LIST_SUFFIX           "50.0"
#define HWMCA_PARTITION_ID_SUFFIX           "51.0"
#define HWMCA_ACT_PROFILE_IPLTYPE_SUFFIX    "52.0"
#define HWMCA_ACT_PROFILE_WWPN_SUFFIX       "53.0"
#define HWMCA_ACT_PROFILE_BPS_SUFFIX        "54.0"
#define HWMCA_ACT_PROFILE_LUN_SUFFIX        "55.0"
#define HWMCA_ACT_PROFILE_BRLBA_SUFFIX      "56.0"
#define HWMCA_ACT_PROFILE_OSLOADPARM_SUFFIX "57.0"
#define HWMCA_EVENT_TEXT_SUFFIX             "58.0"
#define HWMCA_EVENT_TIMESTAMP_SUFFIX        "59.0"
#define HWMCA_IFA_WEIGHT_SUFFIX             "60.0"
#define HWMCA_IFA_CAPPED_SUFFIX             "61.0"
#define HWMCA_IFA_MINIMUM_WEIGHT_SUFFIX     "62.0"
#define HWMCA_IFA_MAXIMUM_WEIGHT_SUFFIX     "63.0"
#define HWMCA_IFA_CURRENT_WEIGHT_SUFFIX     "64.0"
#define HWMCA_IFA_CURRENT_CAPPED_SUFFIX     "65.0"
#define HWMCA_IFL_WEIGHT_SUFFIX             "66.0"
#define HWMCA_IFL_CAPPED_SUFFIX             "67.0"
#define HWMCA_IFL_MINIMUM_WEIGHT_SUFFIX     "68.0"
#define HWMCA_IFL_MAXIMUM_WEIGHT_SUFFIX     "69.0"
#define HWMCA_IFL_CURRENT_WEIGHT_SUFFIX     "70.0"
#define HWMCA_IFL_CURRENT_CAPPED_SUFFIX     "71.0"
#define HWMCA_ICF_WEIGHT_SUFFIX             "72.0"
#define HWMCA_ICF_CAPPED_SUFFIX             "73.0"
#define HWMCA_ICF_MINIMUM_WEIGHT_SUFFIX     "74.0"
#define HWMCA_ICF_MAXIMUM_WEIGHT_SUFFIX     "75.0"
#define HWMCA_ICF_CURRENT_WEIGHT_SUFFIX     "76.0"
#define HWMCA_ICF_CURRENT_CAPPED_SUFFIX     "77.0"
#define HWMCA_PROCESSOR_RUNNING_TIME_TYPE   "78.0"
#define HWMCA_PROCESSOR_RUNNING_TIME        "79.0"
#define HWMCA_END_TIMESLICE_IF_WAITSTATE    "80.0"
#define HWMCA_IIP_WEIGHT_SUFFIX             "81.0"
#define HWMCA_IIP_CAPPED_SUFFIX             "82.0"
#define HWMCA_IIP_MINIMUM_WEIGHT_SUFFIX     "83.0"
#define HWMCA_IIP_MAXIMUM_WEIGHT_SUFFIX     "84.0"
#define HWMCA_IIP_CURRENT_WEIGHT_SUFFIX     "85.0"
#define HWMCA_IIP_CURRENT_CAPPED_SUFFIX     "86.0"
#define HWMCA_OOCOD_INSTALLED_SUFFIX        "87.0"
#define HWMCA_OOCOD_ACTIVATED_SUFFIX        "88.0"
#define HWMCA_OOCOD_ENABLED_SUFFIX          "89.0"
#define HWMCA_OOCOD_ACTIVATION_DATE_SUFFIX  "90.0"
#define HWMCA_ACT_GROUP_LIST_SUFFIX         "91.0"
#define HWMCA_ACT_PROFILE_CAPACITY_SUFFIX   "92.0"
#define HWMCA_GROUP_PROFILE_NAME_SUFFIX     "93.0"
```

```
#define HWMCA_ACT_PROFILE_LOAD_AT_ACTIVATION_SUFFIX         "94.0"
#define HWMCA_ACT_PROFILE_CENTRAL_STORAGE_SUFFIX            "95.0"
#define HWMCA_ACT_PROFILE_CENTRAL_STORAGE_RESERVED_SUFFIX   "96.0"
#define HWMCA_ACT_PROFILE_EXPANDED_STORAGE_SUFFIX           "97.0"
#define HWMCA_ACT_PROFILE_EXPANDED_STORAGE_RESERVED_SUFFIX  "98.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_CP_SUFFIX           "99.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_CP_RESERVED_SUFFIX  "100.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_IFA_SUFFIX          "101.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_IFA_RESERVED_SUFFIX "102.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_IFL_SUFFIX          "103.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_IFL_RESERVED_SUFFIX "104.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_ICF_SUFFIX          "105.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_ICF_RESERVED_SUFFIX "106.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_IIP_SUFFIX          "107.0"
#define HWMCA_ACT_PROFILE_NUM_DEDICATED_IIP_RESERVED_SUFFIX "108.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_CP_SUFFIX              "109.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_CP_RESERVED_SUFFIX     "110.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_IFA_SUFFIX             "111.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_IFA_RESERVED_SUFFIX    "112.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_IFL_SUFFIX             "113.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_IFL_RESERVED_SUFFIX    "114.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_ICF_SUFFIX             "115.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_ICF_RESERVED_SUFFIX    "116.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_IIP_SUFFIX             "117.0"
#define HWMCA_ACT_PROFILE_NUM_SHARED_IIP_RESERVED_SUFFIX    "118.0"
#define HWMCA_CAPACITY_RECORD_LIST_SUFFIX                   "119.0"
#define HWMCA_PERM_SOFTWARE_MODEL_SUFFIX                    "120.0"
#define HWMCA_PERMBILL_SOFTWARE_MODEL_SUFFIX                "121.0"
#define HWMCA_PERMALL_SOFTWARE_MODEL_SUFFIX                 "122.0"
#define HWMCA_PERM_MSU_SUFFIX                               "123.0"
#define HWMCA_PERMBILL_MSU_SUFFIX                           "124.0"
#define HWMCA_PERMALL_MSU_SUFFIX                            "125.0"
#define HWMCA_GEN_PROCESSOR_NUM_SUFFIX                      "126.0"
#define HWMCA_SAP_PROCESSOR_NUM_SUFFIX                      "127.0"
#define HWMCA_IFA_PROCESSOR_NUM_SUFFIX                      "128.0"
#define HWMCA_IFL_PROCESSOR_NUM_SUFFIX                      "129.0"
#define HWMCA_ICF_PROCESSOR_NUM_SUFFIX                      "130.0"
#define HWMCA_IIP_PROCESSOR_NUM_SUFFIX                      "131.0"
#define HWMCA_DEFECTIVE_PROCESSOR_NUM_SUFFIX                "132.0"
#define HWMCA_SPARE_PROCESSOR_NUM_SUFFIX                    "133.0"
#define HWMCA_PENDING_PROCESSOR_NUM_SUFFIX                  "134.0"
#define HWMCA_RECORD_ID_SUFFIX                              "135.0"
#define HWMCA_RECORD_TYPE_SUFFIX                            "136.0"
#define HWMCA_RECORD_ACTIVATION_STATUS_SUFFIX               "137.0"
#define HWMCA_RECORD_ACTIVATION_DATE_SUFFIX                 "138.0"
#define HWMCA_RECORD_EXPIRE_DATE_SUFFIX                     "139.0"
#define HWMCA_RECORD_ACT_EXPIRE_DATE_SUFFIX                 "140.0"
#define HWMCA_RECORD_MAX_REAL_ACT_DAYS_SUFFIX               "141.0"
#define HWMCA_RECORD_MAX_TEST_ACT_DAYS_SUFFIX               "142.0"
#define HWMCA_RECORD_REM_REAL_ACT_DAYS_SUFFIX               "143.0"
```

```
#define HWMCA_RECORD_REM_TEST_ACT_DAYS_SUFFIX              "144.0"
#define HWMCA_CAPACITY_CHANGE_TYPE_SUFFIX                  "145.0"
#define HWMCA_RECORD_CHANGE_TYPE_SUFFIX                    "146.0"
#define HWMCA_RECORD_REM_REAL_COUNT_SUFFIX                 "147.0"
#define HWMCA_RECORD_REM_TEST_COUNT_SUFFIX                 "148.0"
#define HWMCA_CAPACITY_CHANGE_ALLOWED_SUFFIX               "149.0"
#define HWMCA_PSW_SUFFIX                                   "150.0"
#define HWMCA_PROCESSOR_SUFFIX                             "150.1"
#define HWMCA_SCP_INITIATE_RESET_SUFFIX                    "150.2"
#define HWMCA_VERSION_SUFFIX                               "151.0"
#define HWMCA_POWER_VERSION_INFO_SUFFIX                    "152.0"
#define HWMCA_POWER_BUFFER_TAG_SUFFIX                      "153.0"
#define HWMCA_POWER_STATUS_REGISTER_SUFFIX                 "154.0"
#define HWMCA_POWER_EVENT_REGISTER_SUFFIX                  "155.0"
#define HWMCA_POWER_ERROR_REGISTER_SUFFIX                  "156.0"
#define HWMCA_POWER_EXHAUST_HEAT_INDEX_SUFFIX              "157.0"
#define HWMCA_POWER_INLET_TEMP_SUFFIX                      "158.0"
#define HWMCA_POWER_AVG_POWER_SAMPLES_SUFFIX               "159.0"
#define HWMCA_POWER_PEAK_POWER_SAMPLES_SUFFIX              "160.0"
#define HWMCA_ALL_IP_ADDRESSES_SUFFIX                      "161.0"
#define HWMCA_EC_MCL_INFO_SUFFIX                           "162.0"
#define HWMCA_AUTO_SWITCH_ENABLED_SUFFIX                   "163.0"
#define HWMCA_IPL_TOKEN_SUFFIX                             "164.0"
#define HWMCA_SYSPLEX_TIME_STP_INFO_SUFFIX                 "165.0"
#define HWMCA_ACT_PROFILE_STORESTATUS_SUFFIX               "166.0"
#define HWMCA_ACT_PROFILE_LOADTYPE_SUFFIX                  "167.0"
#define HWMCA_CPU_COUNTER_BASIC_ENABLED_SUFFIX             "168.0"
#define HWMCA_CPU_COUNTER_PROBLEMSTATE_ENABLED_SUFFIX      "169.0"
#define HWMCA_CPU_COUNTER_CRYPTOACTIVITY_ENABLED_SUFFIX    "170.0"
#define HWMCA_CPU_COUNTER_EXTENDED_ENABLED_SUFFIX          "171.0"
#define HWMCA_CPU_COUNTER_COPROCGROUP_ENABLED_SUFFIX       "172.0"
#define HWMCA_CPU_SAMPLING_BASIC_ENABLED_SUFFIX            "173.0"
#define HWMCA_PENDING_GEN_PROCESSOR_NUM_SUFFIX             "175.0"
#define HWMCA_PENDING_SAP_PROCESSOR_NUM_SUFFIX             "176.0"
#define HWMCA_PENDING_IFA_PROCESSOR_NUM_SUFFIX             "177.0"
#define HWMCA_PENDING_IFL_PROCESSOR_NUM_SUFFIX             "178.0"
#define HWMCA_PENDING_ICF_PROCESSOR_NUM_SUFFIX             "179.0"
#define HWMCA_PENDING_IIP_PROCESSOR_NUM_SUFFIX             "180.0"
#define HWMCA_ZBX_CHASSIS_LIST_SUFFIX                      "181.0"
#define HWMCA_POWER_BUFFER_SIZE_SUFFIX                     "182.0"
#define HWMCA_ENCRYPT_AES_FUNCTIONS_SUFFIX                 "183.0"
#define HWMCA_ENCRYPT_DEA_FUNCTIONS_SUFFIX                 "184.0"
#define HWMCA_LABEL_POWER_SUFFIX                           "185.0"
#define HWMCA_POWER_SAMPLE_RATE_SUFFIX                     "186.0"
#define HWMCA_GROUP_PROFILE_CAPACITY_SUFFIX                "192.0"
#define HWMCA_LAST_USED_LOAD_ADDR_SUFFIX                   "201.0"
#define HWMCA_LAST_USED_LOAD_PARM_SUFFIX                   "202.0"
#define HWMCA_DESCRIPTION_SUFFIX                           "203.0"
#define HWMCA_OPERATING_MODE_SUFFIX                        "204.0"
#define HWMCA_CLOCK_TYPE_SUFFIX                            "205.0"
#define HWMCA_TIME_OFFSET_DAYS_SUFFIX                      "206.0"
#define HWMCA_TIME_OFFSET_HOURS_SUFFIX                     "207.0"
#define HWMCA_TIME_OFFSET_MINUTES_SUFFIX                   "208.0"
#define HWMCA_TIME_OFFSET_INCREASE_SUFFIX                  "209.0"
#define HWMCA_LICCC_VALIDATION_ENABLED_SUFFIX              "210.0"
#define HWMCA_GLOBAL_PERFORMANCE_DATA_CONTROL_SUFFIX       "211.0"
#define HWMCA_IO_CONFIGURATION_CONTROL_SUFFIX              "212.0"
#define HWMCA_CROSS_PARTITION_AUTHORITY_SUFFIX             "213.0"
#define HWMCA_LOGICAL_PARTITION_ISOLATION_SUFFIX           "214.0"
```

```
|  #define HWMCA_ABS_CAPPED_SUFFIX                    "217.0"
|  #define HWMCA_ABS_CAP_VALUE_SUFFIX                 "218.0"
|  #define HWMCA_IFA_ABS_CAPPED_SUFFIX                "219.0"
|  #define HWMCA_IFA_ABS_CAP_VALUE_SUFFIX             "220.0"
|  #define HWMCA_IFL_ABS_CAPPED_SUFFIX                "221.0"
|  #define HWMCA_IFL_ABS_CAP_VALUE_SUFFIX             "222.0"
|  #define HWMCA_ICF_ABS_CAPPED_SUFFIX                "223.0"
|  #define HWMCA_ICF_ABS_CAP_VALUE_SUFFIX             "224.0"
|  #define HWMCA_IIP_ABS_CAPPED_SUFFIX                "225.0"
|  #define HWMCA_IIP_ABS_CAP_VALUE_SUFFIX             "226.0"
```

```
/*****************************************************************************/
/* Defines for the Console Command Object IDs.                               */
/*****************************************************************************/
#define HWMCA_ACTIVATE_COMMAND                          "1.3.6.1.4.1.2.6.42.4.1"
#define HWMCA_DEACTIVATE_COMMAND                        "1.3.6.1.4.1.2.6.42.4.2"
#define HWMCA_SEND_OPSYS_COMMAND                        "1.3.6.1.4.1.2.6.42.4.3"
#define HWMCA_RESETNORMAL_COMMAND                       "1.3.6.1.4.1.2.6.42.4.4"
#define HWMCA_START_COMMAND                             "1.3.6.1.4.1.2.6.42.4.5"
#define HWMCA_STOP_COMMAND                              "1.3.6.1.4.1.2.6.42.4.6"
#define HWMCA_PSWRESTART_COMMAND                        "1.3.6.1.4.1.2.6.42.4.7"
#define HWMCA_INITIALIZE_API                            "1.3.6.1.4.1.2.6.42.4.8"
#define HWMCA_TERMINATE_API                             "1.3.6.1.4.1.2.6.42.4.9"
#define HWMCA_LOAD_COMMAND                             "1.3.6.1.4.1.2.6.42.4.10"
#define HWMCA_HW_MESSAGE_REFRESH_COMMAND               "1.3.6.1.4.1.2.6.42.4.11"
#define HWMCA_RESETCLEAR_COMMAND                       "1.3.6.1.4.1.2.6.42.4.12"
#define HWMCA_HW_MESSAGE_DELETE_COMMAND                "1.3.6.1.4.1.2.6.42.4.13"
#define HWMCA_ACTIVATE_CBU_COMMAND                     "1.3.6.1.4.1.2.6.42.4.14"
#define HWMCA_UNDO_CBU_COMMAND                         "1.3.6.1.4.1.2.6.42.4.15"
#define HWMCA_IMPORT_PROFILE_COMMAND                   "1.3.6.1.4.1.2.6.42.4.16"
#define HWMCA_EXPORT_PROFILE_COMMAND                   "1.3.6.1.4.1.2.6.42.4.17"
#define HWMCA_RESERVE_COMMAND                          "1.3.6.1.4.1.2.6.42.4.18"
#define HWMCA_EXTERNAL_INTERRUPT_COMMAND               "1.3.6.1.4.1.2.6.42.4.19"
#define HWMCA_SCSI_LOAD_COMMAND                        "1.3.6.1.4.1.2.6.42.4.20"
#define HWMCA_SCSI_DUMP_COMMAND                        "1.3.6.1.4.1.2.6.42.4.21"
#define HWMCA_SHUTDOWN_RESTART_COMMAND                 "1.3.6.1.4.1.2.6.42.4.22"
#define HWMCA_ACTIVATE_OOCOD_COMMAND                   "1.3.6.1.4.1.2.6.42.4.23"
#define HWMCA_UNDO_OOCOD_COMMAND                       "1.3.6.1.4.1.2.6.42.4.24"
#define HWMCA_ADD_CAPACITY_COMMAND                     "1.3.6.1.4.1.2.6.42.4.25"
#define HWMCA_REMOVE_CAPACITY_COMMAND                  "1.3.6.1.4.1.2.6.42.4.26"
#define HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND            "1.3.6.1.4.1.2.6.42.4.27"
#define HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND      "1.3.6.1.4.1.2.6.42.4.28"
#define HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_COMMAND "1.3.6.1.4.1.2.6.42.4.29"
#define HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND   "1.3.6.1.4.1.2.6.42.4.30"
#define HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_COMMAND  "1.3.6.1.4.1.2.6.42.4.31"
#define HWMCA_LOAD_FROM_CDROM_COMMAND                  "1.3.6.1.4.1.2.6.42.4.99"
#define HWMCA_ACTIVATE_COMMAND_SUFFIX                           "1"
#define HWMCA_DEACTIVATE_COMMAND_SUFFIX                         "2"
#define HWMCA_SEND_OPSYS_COMMAND_SUFFIX                         "3"
#define HWMCA_RESETNORMAL_COMMAND_SUFFIX                        "4"
#define HWMCA_START_COMMAND_SUFFIX                              "5"
#define HWMCA_STOP_COMMAND_SUFFIX                               "6"
#define HWMCA_PSWRESTART_COMMAND_SUFFIX                         "7"
#define HWMCA_INITIALIZE_API_SUFFIX                             "8"
#define HWMCA_TERMINATE_API_SUFFIX                              "9"
#define HWMCA_LOAD_COMMAND_SUFFIX                              "10"
#define HWMCA_HW_MESSAGE_REFRESH_COMMAND_SUFFIX                "11"
#define HWMCA_RESETCLEAR_COMMAND_SUFFIX                        "12"
#define HWMCA_HW_MESSAGE_DELETE_COMMAND_SUFFIX                 "13"
#define HWMCA_ACTIVATE_CBU_COMMAND SUFFIX                      "14"
#define HWMCA_UNDO_CBU_COMMAND SUFFIX                          "15"
#define HWMCA_IMPORT_PROFILE_COMMAND_SUFFIX                    "16"
#define HWMCA_EXPORT_PROFILE_COMMAND_SUFFIX                    "17"
#define HWMCA_RESERVE_COMMAND_SUFFIX                           "18"
#define HWMCA_EXTERNAL_INTERRUPT_COMMAND_SUFFIX                "19"
#define HWMCA_SCSI_LOAD_COMMAND_SUFFIX                         "20"
#define HWMCA_SCSI_DUMP_COMMAND_SUFFIX                         "21"
#define HWMCA_SHUTDOWN_RESTART_COMMAND_SUFFIX                  "22"
#define HWMCA_ACTIVATE_OOCOD_COMMAND_SUFFIX                    "23"
#define HWMCA_UNDO_OOCOD_COMMAND_SUFFIX                        "24"
#define HWMCA_ADD_CAPACITY_COMMAND_SUFFIX                      "25"
#define HWMCA_REMOVE_CAPACITY_COMMAND_SUFFIX                   "26"
#define HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND_SUFFIX             "27"
#define HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND_SUFFIX       "28"
#define HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_COMMAND_SUFFIX  "29"
#define HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND_SUFFIX    "30"
#define HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_COMMAND_SUFFIX   "31"
```

```
/*****************************************************************************/
/* Defines for the Console Message Event Types.                              */
/*****************************************************************************/
#define HWMCA_HARDWARE_MESSAGE         1
#define HWMCA_OPSYS_MESSAGE            2

/*****************************************************************************/
/* Defines for the CPC Managed Object Degraded Indicator                     */
/*****************************************************************************/
#define HWMCA_NOT_DEGRADED             0x0000
#define HWMCA_DEGRADED_MEM             0x0001
#define HWMCA_DEGRADED_MBA             0x0002
#define HWMCA_DEGRADED_NODE            0x0004
#define HWMCA_DEGRADED_RING            0x0008
#define HWMCA_DEGRADED_CBU             0x0010
#define HWMCA_DEGRADED_MRU             0x0020
#define HWMCA_DEGRADED_AMBIENT         0x0040
#define HWMCA_DEGRADED_MRU_IML         0x0080

/*****************************************************************************/
/* Defines for the Hardware Management Console Status Values.                */
/*****************************************************************************/
#define HWMCA_STATUS_OPERATING           0x00000001
#define HWMCA_STATUS_NOT_OPERATING       0x00000002
#define HWMCA_STATUS_NO_POWER            0x00000004
#define HWMCA_STATUS_NOT_ACTIVATED       0x00000008
#define HWMCA_STATUS_EXCEPTIONS          0x00000010
#define HWMCA_STATUS_STATUS_CHECK        0x00000020
#define HWMCA_STATUS_SERVICE             0x00000040
#define HWMCA_STATUS_LINKNOTACTIVE       0x00000080
#define HWMCA_STATUS_POWERSAVE           0x00000100
#define HWMCA_STATUS_SERIOUSALERT        0x00000200
#define HWMCA_STATUS_ALERT               0x00000400
#define HWMCA_STATUS_ENVALERT            0x00000800
#define HWMCA_STATUS_SERVICE_REQ         0x00001000
#define HWMCA_STATUS_DEGRADED            0x00002000
#define HWMCA_STATUS_STORAGE_EXCEEDED    0x01000000
#define HWMCA_STATUS_LOGOFF_TIMEOUT      0x02000000
#define HWMCA_STATUS_FORCED_SLEEP        0x04000000
#define HWMCA_STATUS_IMAGE_NOT_OPERATING 0x08000000
#define HWMCA_STATUS_IMAGE_NOT_ACTIVATED 0x10000000
#define HWMCA_STATUS_IMAGE_NOT_CAPABLE   0x20000000
#define HWMCA_STATUS_UNKNOWN             0x40000000

/*****************************************************************************/
/* Defines for the Hardware Management Console IML Mode Values.              */
/*****************************************************************************/
#define HWMCA_IML_ESA390_MODE             1
#define HWMCA_IML_S370_MODE               2
#define HWMCA_IML_FM_MODE                 6
#define HWMCA_IML_FMAE_MODE               7
#define HWMCA_IML_HM_MODE                 8
#define HWMCA_IML_HMEA_MODE               9
#define HWMCA_IML_HMEX_MODE              10
#define HWMCA_IML_LPAR_MODE              11
#define HWMCA_IML_ESA390TPF_MODE         12
#define HWMCA_IML_CF_PROD_MODE           13
#define HWMCA_IML_FMEX_MODE              14
#define HWMCA_IML_HMAS_MODE              15
#define HWMCA_IML_LINUXO_MODE            16
#define HWMCA_IML_ZVM_MODE               18
#define HWMCA_IML_ZAWARE_MODE            20
```

```
/*****************************************************************************/
/* Defines for the Hardware Management Console IPL Type Values.              */
/*****************************************************************************/
#define HWMCA_IPLTYPE_STANDARD          1
#define HWMCA_IPLTYPE_SCSI              2
#define HWMCA_IPLTYPE_SCSIDUMP          3


/*****************************************************************************/
/* Defines for the Console Object Type Values.                               */
/*****************************************************************************/
#define HWMCA_CPC_GROUP             1
#define HWMCA_CPC_IMAGE_GROUP       2
#define HWMCA_CPC_USER_GROUP        3
#define HWMCA_CPC_IMAGE_USER_GROUP  4
#define HWMCA_CPC_OBJECT            5
#define HWMCA_CPC_IMAGE_OBJECT      6
#define HWMCA_CF_OBJECT             7
#define HWMCA_ACT_PROFILE_RESET     8
#define HWMCA_ACT_PROFILE_IMAGE     9
#define HWMCA_ACT_PROFILE_LOAD      10
#define HWMCA_ACT_PROFILE_GROUP     11
#define HWMCA_CAPACITY_RECORD       12
#define HWMCA_VM_GROUP              13
#define HWMCA_VM_OBJECT             14

/*****************************************************************************/
/* Defines for the Hardware Management Console Shutdown/Restart Types.       */
/*****************************************************************************/
#define HWMCA_RESTART_APPLICATION           1
#define HWMCA_RESTART_CONSOLE               2
#define HWMCA_SHUTDOWN_CONSOLE              3
#define HWMCA_RESTART_APPLICATION_ALTERNATE 4
#define HWMCA_RESTART_CONSOLE_ALTERNATE     5
#define HWMCA_SHUTDOWN_CONSOLE_ALTERNATE    6

/*****************************************************************************/
/* Defines for the Hardware Management Console Ended Event Reasons.          */
/*****************************************************************************/
#define HWMCA_ENDED_USER                    1
#define HWMCA_ENDED_AUTOMATION              2
#define HWMCA_ENDED_OTHER                   3

/*****************************************************************************/
/* Defines for the Hardware Management Console Processor Running Time types. */
/*****************************************************************************/
#define HWMCA_DETERMINED_SYSTEM             0
#define HWMCA_DETERMINED_USER               1

/*****************************************************************************/
/* Defines for the type of capacity record.                                  */
/*****************************************************************************/
#define HWMCA_CAPACITY_RECORD_TYPE_CBU          1
#define HWMCA_CAPACITY_RECORD_TYPE_OOCOD        2
#define HWMCA_CAPACITY_RECORD_TYPE_PLANNED_EVENT 3
#define HWMCA_CAPACITY_RECORD_TYPE_LOANER       4
```

```
/*****************************************************************************/
/* Defines for the activation status of a capacity record.                   */
/*****************************************************************************/
#define HWMCA_CAPACITY_RECORD_STATUS_NOT_ACTIVATED    1
#define HWMCA_CAPACITY_RECORD_STATUS_REAL             2
#define HWMCA_CAPACITY_RECORD_STATUS_TEST             3
#define HWMCA_CAPACITY_RECORD_STATUS_CAN_BE_ACTIVATED 4

/*****************************************************************************/
/* Defines for the type of change for a HWMCA_EVENT_CAPACITY_CHANGE event.   */
/*****************************************************************************/
#define HWMCA_CAPACITY_FENCED_BOOK                0
#define HWMCA_CAPACITY_DEFECTIVE_PROCESSOR        1
#define HWMCA_CAPACITY_CONCURRENT_BOOK_REPLACE    2
#define HWMCA_CAPACITY_CONCURRENT_BOOK_ADD        3
#define HWMCA_CAPACITY_CHECK_STOP                 4
#define HWMCA_CAPACITY_CHANGES_ALLOWED            5
#define HWMCA_CAPACITY_CHANGES_NOT_ALLOWED        6

/*****************************************************************************/
/* Defines for the type of change for a HWMCA_EVENT_CAPACITY_RECORD_CHANGE   */
/* event.                                                                    */
/*****************************************************************************/
#define HWMCA_CAPACITY_RECORD_ADD         0
#define HWMCA_CAPACITY_RECORD_DELTA       1
#define HWMCA_CAPACITY_RECORD_DELETE      2
#define HWMCA_CAPACITY_RECORD_ACCOUNTING  3
#define HWMCA_CAPACITY_ACTIVATION_LEVEL   4
#define HWMCA_CAPACITY_PRIORITY_PENDING   5
#define HWMCA_CAPACITY_RECORD_OTHER       6


/*****************************************************************************/
/* Defines for the Image Activation Profile Operating Mode Values.           */
/*****************************************************************************/
#define HWMCA_ESA390_OPERATING_MODE           1
#define HWMCA_ESA390TPF_OPERATING_MODE        2
#define HWMCA_CF_OPERATING_MODE               3
#define HWMCA_LINUX_OPERATING_MODE            4
#define HWMCA_FMEX_OPERATING_MODE             5
#define HWMCA_HMEX_OPERATING_MODE             6
#define HWMCA_HMAS_OPERATING_MODE             7
#define HWMCA_ZVM_OPERATING_MODE              8
#define HWMCA_ZAWARE_OPERATING_MODE           9

/*****************************************************************************/
/*Defines for the Hardware Management Console Image Profile Clock Type Values.*/
/*****************************************************************************/
#define HWMCA_CLOCK_TYPE_STANDARD 0
#define HWMCA_CLOCK_TYPE_LPAR     1
```

```
/*****************************************************************************/
/* Defines for the type of capacity record.                                  */
/*****************************************************************************/
#define HWMCA_CAPACITY_RECORD_TYPE_CBU           1
#define HWMCA_CAPACITY_RECORD_TYPE_OOCOD         2
#define HWMCA_CAPACITY_RECORD_TYPE_PLANNED_EVENT 3
#define HWMCA_CAPACITY_RECORD_TYPE_LOANER        4

/*****************************************************************************/
/* Defines for the activation status of a capacity record.                   */
/*****************************************************************************/
#define HWMCA_CAPACITY_RECORD_STATUS_NOT_ACTIVATED    1
#define HWMCA_CAPACITY_RECORD_STATUS_REAL             2
#define HWMCA_CAPACITY_RECORD_STATUS_TEST             3
#define HWMCA_CAPACITY_RECORD_STATUS_CAN_BE_ACTIVATED 4
/*****************************************************************************/
/* Defines for the type of change for a HWMCA_EVENT_CAPACITY_CHANGE event.   */
/*****************************************************************************/
#define HWMCA_CAPACITY_FENCED_BOOK                0
#define HWMCA_CAPACITY_DEFECTIVE_PROCESSOR        1
#define HWMCA_CAPACITY_CONCURRENT_BOOK_REPLACE    2
#define HWMCA_CAPACITY_CONCURRENT_BOOK_ADD        3
#define HWMCA_CAPACITY_CHECK_STOP                 4
#define HWMCA_CAPACITY_CHANGES_ALLOWED            5
#define HWMCA_CAPACITY_CHANGES_NOT_ALLOWED        6

/*****************************************************************************/
/* Defines for the type of change for a HWMCA_EVENT_CAPACITY_RECORD_CHANGE   */
/* event.                                                                    */
/*****************************************************************************/
#define HWMCA_CAPACITY_RECORD_ADD         0
#define HWMCA_CAPACITY_RECORD_DELTA       1
#define HWMCA_CAPACITY_RECORD_DELETE      2
#define HWMCA_CAPACITY_RECORD_ACCOUNTING  3
#define HWMCA_CAPACITY_ACTIVATION_LEVEL   4
#define HWMCA_CAPACITY_PRIORITY_PENDING   5
#define HWMCA_CAPACITY_RECORD_OTHER       6
```

Data exchange APIs SNMP target structure (HWMCA_SNMP_TARGET_T)

```
/*****************************************************************************/
/* Console SNMP Target Structure                                             */
/*****************************************************************************/
struct HWMCA_SNMP_TARGET_S {
   PVOID                pHost;        /* A pointer to a null terminated      */
                                      /* string specifying the host name or  */
                                      /* internet address for the target     */
                                      /* Console.                            */
                                      /*                                     */
   CHAR szCommunity[HWMCA_MAX_COMMUNITY_LEN]; /* Community name to be used    */
                                      /* for requests.                       */
|  UINT ulSecurityVersion;            /* Security version used v2c or v3     */
|  CHAR szUsername[HWMCA_MAX_USERNAME_LEN]; /* Username to be used for v3 auth */
|
|  CHAR szPassword[HWMCA_MAX_USERNAME_LEN]; /* Password to be used for v3 auth */
|
|  UINT ulReserved;                   /* Reserved field.                      */
};

typedef struct HWMCA_SNMP_TARGET_S HWMCA_SNMP_TARGET_T;
typedef HWMCA_SNMP_TARGET_T  *     HWMCA_SNMP_TARGET_P;
#define HWMCA_SNMP_TARGET_SIZE sizeof(HWMCA_SNMP_TARGET_T)
```

Data exchange APIs initialize structure (HWMCA_INITIALIZE_T)

```
   /*****************************************************************************/
   /* Console Initialize Structure                                              */
   /*****************************************************************************/
   struct HWMCA_INITIALIZE_S {
      PVOID                pTarget;      /* Pointer to data specifying the      */
                                         /* target Hardware Management Console  */
                                         /* for the request.                    */
                                         /*                                     */
                                         /* For the SNMP APIs, this is an       */
                                         /* HWMCA_SNMP_TARGET_S structure.      */
                                         /*                                     */
|     UINT                   ulEventMask;  /* A mask specifying the event         */
                                         /* notifications that the application  */
                                         /* wants to register for.              */
                                         /*                                     */
                                         /* - HWMCA_EVENT_COMMAND_RESPONSE      */
                                         /* - HWMCA_EVENT_MESSAGE               */
                                         /* - HWMCA_EVENT_STATUS_CHANGE         */
                                         /* - HWMCA_EVENT_NAME_CHANGE           */
                                         /* - HWMCA_EVENT_ACTIVATE_PROF_CHANGE  */
                                         /* - HWMCA_EVENT_CREATED               */
                                         /* - HWMCA_EVENT_DESTROYED             */
                                         /* - HWMCA_EVENT_EXCEPTION_STATE       */
                                         /* - HWMCA_EVENT_ENDED                 */
                                         /* - HWMCA_EVENT_HARDWARE_MESSAGE      */
                                         /* - HWMCA_EVENT_OPSYS_MESSAGE         */
                                         /* - HWMCA_EVENT_NO_REFRESH_MESSAGE    */
                                         /* - HWMCA_EVENT_STARTED               */
                                         /* - HWMCA_EVENT_HARDWARE_MESSAGE_DELETE*/
                                         /* - HWMCA_DIRECT_INITIALIZE           */
                                         /* - HWMCA_FORCE_CLIENT_PATH           */
                                         /* - HWMCA_SNMP_VERSION_2              */
                                         /*                                     */
      ULONG                ulReserved;   /* Must be zero.                       */
      union {
         struct {
            INT              iAgentSocket; /* Socket used to communicate with the */
                                         /* SNMP agent on the target Console.
|           UINT             ulInetAddr;   /* Internet address for the SNMP agent.*/
|           UINT             uiSecVersion;
|           CHAR szCommunity[HWMCA_MAX_COMMUNITY_LEN]; /* Community name to be   */
|                                        /* used for requests.                  */
|           struct {
|              unsigned char bAuthEngineId[HWMCA_MAX_ID_LEN]; // Don't know what the real max is supposed to be
|              UINT ulAuthEngineIdLength;
|              CHAR szUsername[HWMCA_MAX_USERNAME_LEN];
|              CHAR szPassword[HWMCA_MAX_USERNAME_LEN];
|              UINT ulAuthEngineBoots;
|              UINT ulAuthEngineTime;
|              UINT ulMsgId;
|              unsigned char bPrivateKey[16];
|              UINT uiSalt;
|           } v3;
         } snmp;
      } protocol;
   };

   typedef struct HWMCA_INITIALIZE_S HWMCA_INITIALIZE_T;
   typedef HWMCA_INITIALIZE_T  *     HWMCA_INITIALIZE_P;
   #define HWMCA_INITIALIZE_SIZE sizeof(HWMCA_INITIALIZE_T)
```

Data exchange APIs datatype structure (HWMCA_DATATYPE_T)

```
/*****************************************************************************/
/* Console Data Type Structure                                               */
/*****************************************************************************/
struct HWMCA_DATATYPE_S {
   UCHAR                ucType;       /* Type of the data:                   */
                                      /*  - HWMCA_TYPE_SEQUENCE              */
                                      /*  - HWMCA_TYPE_INTEGER               */
                                      /*  - HWMCA_TYPE_OCTETSTRING           */
                                      /*  - HWMCA_TYPE_NULL                  */
                                      /*  - HWMCA_TYPE_OBJECTID              */
                                      /*  - HWMCA_TYPE_IPADDRESS             */
   ULONG                ulLength;     /* Length of the data.                 */
   PVOID                pData;        /* Pointer to the data itself.         */
   struct HWMCA_DATATYPE_S *pNext;       /* Pointer to next data type structure */
};

typedef struct HWMCA_DATATYPE_S HWMCA_DATATYPE_T;
typedef HWMCA_DATATYPE_T  *     HWMCA_DATATYPE_P;
#define HWMCA_DATATYPE_SIZE sizeof(HWMCA_DATATYPE_T)
/*****************************************************************************/
/* Hardware Management Console Event Qualifier Structure                     */
/*****************************************************************************/
struct HWMCA_EVENT_QUALIFIER_S {
|    unsigned int ulEventMask;           /* Event mask for qualifier            */
|    unsigned int ulType;               /* Qualifier type                      */
   union {
      char szName[256];                 /* Image name for OS msgs events       */
      char cReserved[256];              /* Reserved space                      */
   } type;                              /* union of qualifier data             */
   struct HWMCA_EVENT_QUALIFIER_S *pNext;/* Pointer to next qualifier struct  */
};
typedef struct HWMCA_EVENT_QUALIFIER_S HWMCA_EVENT_QUALIFIER_T;
typedef HWMCA_EVENT_QUALIFIER_T  *     HWMCA_EVENT_QUALIFIER_P;
#define HWMCA_EVENT_QUALIFIER_SIZE sizeof(HWMCA_EVENT_QUALIFIER_T)
#define HWMCA_QUALIFIER_TYPE_NAME      0x00000001
```

Function prototypes

```
/*****************************************************************************/
/* Console Data Exchange Function Prototypes                                 */
/*****************************************************************************/
extern ULONG EXPENTRY HwmcaInitialize(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */

extern ULONG EXPENTRY HwmcaGet(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       PSZ,                      /* Pointer to null terminated object ID      */
                                 /* string.                                   */
       PVOID,                    /* Pointer to an output buffer for the       */
                                 /* returned data.                            */
       ULONG,                    /* Size of the output buffer.                */
       PULONG,                   /* Pointer to an area where the number of    */
                                 /* total bytes needed for this Get request   */
                                 /* is returned.                              */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */
```

```
extern ULONG EXPENTRY HwmcaGetNext(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       PSZ,                      /* Pointer to null terminated object ID      */
                                 /* string.                                   */
       PVOID,                    /* Pointer to an output buffer for the       */
                                 /* returned data.                            */
       ULONG,                    /* Size of the output buffer.                */
       PULONG,                   /* Pointer to an area where the number of    */
                                 /* total bytes needed for this Get request   */
                                 /* is returned.                              */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */

extern ULONG EXPORTTYPE HwmcaGetBulk(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       HWMCA_DATATYPE_P,         /* Pointer to a linked list of               */
                                 /* HWMCA_DATATYPE_T structures used to       */
                                 /* specify the object IDs to use in the      */
                                 /* GetBulk request.                          */
       UINT,                     /* Count of non-repeaters for the request.   */
       UINT,                     /* Maximum repititions for the request.      */
       PVOID,                    /* Pointer to an output buffer for the       */
                                 /* returned data.                            */
       ULONG,                    /* Size of the output buffer.                */
       PULONG,                   /* Pointer to an area where the number of    */
                                 /* total bytes needed for this Get request   */
                                 /* is returned.                              */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */
```

```
extern ULONG EXPENTRY HwmcaSet(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       PSZ,                      /* Pointer to null terminated object ID      */
                                 /* string.                                   */
       HWMCA_DATATYPE_P,         /* Pointer to a linked list of               */
                                 /* HWMCA_DATATYPE_T structures used to       */
                                 /* represent the data.                       */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */
extern ULONG EXPENTRY HwmcaWaitEvent(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       PVOID,                    /* Pointer to an output buffer for the       */
                                 /* returned data.                            */
       ULONG,                    /* Size of the output buffer.                */
       PULONG,                   /* Pointer to an area where the number of    */
                                 /* total bytes needed for this Get request   */
                                 /* is returned.                              */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */
extern ULONG EXPENTRY HwmcaTerminate(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */

extern ULONG EXPENTRY HwmcaCommand(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                 /* structure.                                */
       PSZ,                      /* Pointer to null terminated object ID      */
                                 /* string that the command target.           */
       PSZ,                      /* Pointer to null terminated object ID      */
                                 /* string that command identifier.           */
       HWMCA_DATATYPE_P,         /* Pointer to a linked list of               */
                                 /* HWMCA_DATATYPE_T structures used to       */
                                 /* represent the argument data.              */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */
extern ULONG EXPORTTYPE HwmcaCorrelatedCommand(
      HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
                                /* structure.                                */
      PSZ,                      /* Pointer to null terminated object ID      */
                                /* string that the command target.           */
      PSZ,                      /* Pointer to null terminated object ID      */
                                /* string that command identifier.           */
      HWMCA_DATATYPE_P,         /* Pointer to a linked list of               */
                                /* HWMCA_DATATYPE_T structures used to       */
                                /* represent the argument data.              */
      ULONG,                    /* Time to wait for the next event           */
                                /* notification (in milliseconds).           */
      void *,                   /* Pointer to correlator data.               */
      unsigned int);            /* Size of correlator data.                  */

extern ULONG EXPORTTYPE HwmcaRegister(
       HWMCA_INITIALIZE_P,       /* Pointer to data exchange initialization   */
       UINT,                     /* New event mask to be used                 */
       HWMCA_EVENT_QUALIFIER_P,  /* New event qualifiers to be used           */
       ULONG);                   /* Time to wait for the next event           */
                                 /* notification (in milliseconds).           */
```

```
extern ULONG EXPENTRY HwmcaBuildId(
       PSZ,                     /* Pointer to a buffer where the built object*/
                                /* identifier string is to be placed.        */
       PSZ,                     /* Pointer to the prefix string to be used   */
                                /* for the object identifier to be built.    */
                                /*  - HWMCA_CONSOLE_ID                       */
                                /*  - HWMCA_CFG_CPC_GROUP_ID                 */
                                /*  - HWMCA_CFG_CPC_ID                       */
                                /*  - HWMCA_CPC_IMAGE_GROUP_ID               */
                                /*  - HWMCA_CPC_IMAGE_ID                     */
                                /*  - HWMCA_GROUPS_GROUP_ID                  */
                                /*  - HWMCA_COMMAND_PREFIX                   */
                                /*  - HWMCA_ACT_RESET_OBJECT_ID              */
                                /*  - HWMCA_ACT_IMAGE_OBJECT_ID              */
                                /*  - HWMCA_ACT_LOAD_OBJECT_ID               */
       PSZ,                     /* Pointer to the attribute suffix string to */
                                /* be used for the object identifier to be   */
                                /* build.  This can be specified as NULL,    */
                                /* when building an ID for an object itself, */
                                /* as opposed to an attribute object ID.     */
       PSZ,                     /* Pointer to the Group name to be used for  */
                                /* building the object identifier.  This can */
                                /* be specified as NULL, when building an ID */
                                /* for a predefined group or an object from  */
                                /* a predefined group.                       */
       PSZ);                    /* Pointer to the Object name to be used for */
                                /* building the object identifier.  This can */
                                /* be specified as NULL, when building an ID */
                                /* for a group object.                       */

extern ULONG EXPENTRY HwmcaBuildAttributeId(
       PSZ,                     /* Pointer to a buffer where the built object*/
                                /* identifier string for the attribute is to */
                                /* be placed.                                */
       PSZ,                     /* Pointer to the object identifier for the  */
                                /* object for which the attribute identifier */
                                /* is to be built.                           */
       PSZ);                    /* Pointer to the attribute suffix string to */
                                /* be used for the attribute identifier to be*/
                                /* build.                                    */
```

Data exchange APIs and commands API example

Refer to the following pages for some example code using the Console Data Exchange APIs and Commands API. A copy of this code can be found on Resource Link at *http://www.ibm.com/servers/resourcelink*. Click **Services**, and then Click **API**.

For more information about the parameters required for this example, simply execute the program with no arguments. This will print out help information to the screen. Some sample invocations for this example program are:

- HWMCATST 1 9.130.1.1 1.3.6.1.4.1.2.6.42.0.23.0

  This will perform a get operation for the *Group Contents* attribute of the Console object.
- HWMCATST 1 9.130.1.1 1.3.6.1.4.1.2.6.42.1.23.0

  Performs a get operation for the *Group Contents* attribute of the Defined CPC Group object.
- HWMCATST 1 9.130.1.1 1.3.6.1.4.1.2.6.42.2.23.0

  Performs a get operation for the *Group Contents* attribute of the CPC Images Group object.
- HWMCATST 1 9.130.1.1 1.3.6.1.4.1.2.6.42.1.0.10.0.3362806951

  Performs a get operation for the *Status* attribute of the Defined CPC object named CPC01.

- HWMCATST 4 9.130.1.1 1.3.6.1.4.1.2.6.42.1.0.3362806951 1.3.6.1.4.1.2.6.42.4.1
  Sends an Activate command request to the Defined CPC object named CPC01.
- HWMCATST 5 9.130.1.1 255 -1
  Waits forever for all types of event notifications.

```
/**********************   Defines        **********************************/
#define INCL_DOS

#define HWMCAAPI_TIMEOUT 30000
#define COMMUNITY "public"

/**********************   Include Files  **********************************/
#include <os2.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include "hwmcaapi.h"

/**********************   Function Prototypes  ****************************/
VOID parse_and_print_get(HWMCA_DATATYPE_P);
VOID parse_and_print_get_group_contents(HWMCA_DATATYPE_P);


/**********************   Main  *******************************************/
void main(argc, argv, envp)
   int argc;
   char *argv[];
   char *envp[];
{
   APIRET              usRc;               /* Local return code from API calls       */
   ULONG               ulLength;           /* Number of bytes passed to an API call  */
   ULONG               ulBytesNeeded;      /* Number of bytes needed for an API call */
   USHORT              usfContinue;        /* Local continue processing flag         */
   HWMCA_DATATYPE_T    tHwmcaDataType;     /* HWMCA DataType structure                 */
   HWMCA_DATATYPE_P    pHwmcaDataType;     /* Ptr to a HWMCA DataType structure        */
   HWMCA_DATATYPE_T    aHwmcaDataType[10]; /* HWMCA DataType structure                 */
   ULONG               aulCmdData[10];     /* array of command integer data          */
   HWMCA_INITIALIZE_T  tHwmcaInitialize;   /* Structure for HwmcaInitialize API call   */
   HWMCA_SNMP_TARGET_T tHwmcaSnmpTarget;   /* Target structure for HwmcaInitialize call*/
   INT                 i, j;               /* loop variables                         */
   CHAR                cEventBuf[HWMCA_MAX_EVENT_BUF_SIZE];
   CHAR                szOID[HWMCA_MAX_ID_LEN];
   PSZ                 pszAttribute, pszGroupName, pszObjectName;

   usfContinue  = TRUE;
   memset(&tHwmcaInitialize,'\0',HWMCA_INITIALIZE_SIZE);
   if (argc >= 4) {  /* Proper number of initial arguments passed */
      switch (atoi(argv[1])) {
         case 1: /* Get request */
            break;
         case 2: /* Get-Next request */
            break;
         case 3: /* Set request */
            if (argc != 6 ) { /* Proper number of arguments passed */
               usfContinue = FALSE;
            } /* endif */
            break;
```

```
   case 4: /* Command request */
      if (argc < 5 ) { /* Proper number of arguments passed */
         usfContinue = FALSE;
      } /* endif */
      break;
   case 5: /* WaitEvent request */
      if (argc < 5 ) { /* Proper number of arguments passed */
         usfContinue = FALSE;
      } else {
         tHwmcaInitialize.ulEventMask = (ULONG)atol(argv[3]);
      } /* endif */
      break;
   case 6: /* BuildId request */
      break;
   case 7: /* BuildAttributeId request */
      if (argc < 5 ) { /* Proper number of arguments passed */
         usfContinue = FALSE;
      } /* endif */
      break;
   default:
      usfContinue = FALSE;
      break;
} /* endswitch */
if (usfContinue) {
   tHwmcaInitialize.pTarget = &tHwmcaSnmpTarget;
   tHwmcaSnmpTarget.pHost   = argv[2];
   strcpy(tHwmcaSnmpTarget.szCommunity,COMMUNITY);
   usRc = HwmcaInitialize(&tHwmcaInitialize,(ULONG)HWMCAAPI_TIMEOUT);
   if (!usRc) {  /* Initialize with HWMCA API server successful */
      printf("HwmcaInitialize call was successful\n");
      printf("HwmcaInitialize target host            = %s\n",
             tHwmcaSnmpTarget.pHost);
      printf("HwmcaInitialize target community name  = %s\n",
             tHwmcaInitialize.protocol.snmp.szCommunity);
      printf("HwmcaInitialize socket                 = %ld\n",
             tHwmcaInitialize.protocol.snmp.iAgentSocket);
      printf("HwmcaInitialize agent Internet address = %x\n",
             tHwmcaInitialize.protocol.snmp.ulInetAddr);
      switch ((atoi(argv[1]))) {
```

```
case 1: /* Get request */
   ulLength     = HWMCA_DATATYPE_SIZE;
   pHwmcaDataType = (HWMCA_DATATYPE_P)NULL;
   memset(&tHwmcaDataType,'\0',HWMCA_DATATYPE_SIZE);
   usRc = HwmcaGet(&tHwmcaInitialize,argv[3],&tHwmcaDataType,
                ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
   if (!usRc) { /* Data returned from HwmcaGet */
      /* Need a larger buffer for the Get request */
      if (ulBytesNeeded > ulLength) {
         pHwmcaDataType = (HWMCA_DATATYPE_P)(malloc(ulBytesNeeded));
         if (pHwmcaDataType) {
            memset(pHwmcaDataType,'\0',ulBytesNeeded);
            ulLength = ulBytesNeeded;
            usRc = HwmcaGet(&tHwmcaInitialize,argv[3],pHwmcaDataType,
                            ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
            if (!usRc) { /* Get request successful */
               /* Check if it is a Group contents Get */
               if (strstr(argv[3],HWMCA_GROUP_CONTENTS_SUFFIX)) {
                  parse_and_print_get_group_contents(pHwmcaDataType);
               } else {
                  parse_and_print_get(pHwmcaDataType);
               } /* endif */
               free(pHwmcaDataType);
            } else {
               printf("Error in HwmcaGet call return code = %ld\n",usRc);
            } /* endif */
         } else {
            printf("Error in allocating %ld bytes for an HwmcaGet call",
                   ulBytesNeeded);
         } /* endif */
      } else {
         /* Check if it is a Group contents Get */
         if (strstr(argv[3],HWMCA_GROUP_CONTENTS_SUFFIX)) {
            parse_and_print_get_group_contents(&tHwmcaDataType);
         } else {
            parse_and_print_get(&tHwmcaDataType);
         } /* endif */
      } /* endif */
   } else {
      printf("Error in HwmcaGet call return code = %ld\n",usRc);
   } /* endif */
   break;
```

```
case 2: /* Get-Next request */
   ulLength     = HWMCA_DATATYPE_SIZE;
   pHwmcaDataType = (HWMCA_DATATYPE_P)NULL;
   memset(&tHwmcaDataType,'\0',HWMCA_DATATYPE_SIZE);
   usRc = HwmcaGetNext(&tHwmcaInitialize,argv[3],&tHwmcaDataType,
                       ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
   if (!usRc) { /* Data returned from HwmcaGetNext */
      /* Need a larger buffer for the Get request */
      if (ulBytesNeeded > ulLength) {
         pHwmcaDataType = (HWMCA_DATATYPE_P)(malloc(ulBytesNeeded));
         if (pHwmcaDataType) {
            memset(pHwmcaDataType,'\0',ulBytesNeeded);
            ulLength = ulBytesNeeded;
            usRc = HwmcaGetNext(&tHwmcaInitialize,argv[3],
                                pHwmcaDataType,ulLength,
                                &ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
            if (!usRc) { /* Get request successful */
               /* Check if it is a Group contents Get-Next */
               if ((pHwmcaDataType->ucType == HWMCA_TYPE_OBJECTID) &&
                   (strstr(pHwmcaDataType->pData,HWMCA_GROUP_CONTENTS_SUFFIX))) {
                  parse_and_print_get_group_contents(pHwmcaDataType);
               } else {
                  parse_and_print_get(pHwmcaDataType);
               } /* endif */
               free(pHwmcaDataType);
            } else {
               printf("Error in HwmcaGetNext call return code = %ld\n",usRc);
            } /* endif */
         } else {
            printf("Error in allocating %ld bytes for an HwmcaGet call",
                   ulBytesNeeded);
         } /* endif */
      } else {
         /* Check if it is a Group contents Get-Next */
         if ((pHwmcaDataType->ucType == HWMCA_TYPE_OBJECTID) &&
             (strstr(pHwmcaDataType->pData,HWMCA_GROUP_CONTENTS_SUFFIX))) {
            parse_and_print_get_group_contents(&tHwmcaDataType);
         } else {
            parse_and_print_get(&tHwmcaDataType);
         } /* endif */
      } /* endif */
   } else {
      printf("Error in HwmcaGetNext call return code = %ld\n",usRc);
   } /* endif */
   break;
```

```
case 3: /* Set request */
   ulLength     = HWMCA_DATATYPE_SIZE;
   pHwmcaDataType = (HWMCA_DATATYPE_P)NULL;
   memset(&tHwmcaDataType,'\0',HWMCA_DATATYPE_SIZE);
   tHwmcaDataType.ucType = (UCHAR)atoi(argv[4]);
   if (tHwmcaDataType.ucType == HWMCA_TYPE_OCTETSTRING) {
      tHwmcaDataType.ulLength = strlen(argv[5])+1;
      tHwmcaDataType.pData    = argv[5];
   } else {
      tHwmcaDataType.ulLength = sizeof(ULONG);
      ulBytesNeeded         = atol(argv[5]);
      tHwmcaDataType.pData    = &ulBytesNeeded;
   } /* endif */
   usRc = HwmcaSet(&tHwmcaInitialize,argv[3],&tHwmcaDataType,(ULONG)HWMCAAPI_TIMEOUT);
   if (usRc) {
      printf("Error in HwmcaSet call return code = %ld\n",usRc);
   } /* endif */
   break;
case 4: /* Command request */
   for (i=5, j=0; (((i+2) <= argc) && (j < 10)); i+=2, j++) {
      memset(&ahWmcaDataType[j],'\0',HWMCA_DATATYPE_SIZE);
      aHwmcaDataType[j].pNext = &(aHwmcaDataType[j+1]);
      aHwmcaDataType[j].ucType = (UCHAR)atoi(argv[i]);
      switch (aHwmcaDataType[j].ucType) {
         case HWMCA_TYPE_OCTETSTRING:
            aHwmcaDataType[j].ulLength = strlen(argv[i+1])+1;
            aHwmcaDataType[j].pData    = argv[i+1];
            break;
         case HWMCA_TYPE_NULL:
            aHwmcaDataType[j].ulLength = 0;
            aHwmcaDataType[j].pData    = (PVOID)NULL;
            break;
         default:
            aHwmcaDataType[j].ulLength = sizeof(ULONG);
            aulCmdData[j]            = atol(argv[i+1]);
            aHwmcaDataType[j].pData    = &aulCmdData[j];
            break;
      } /* endswitch */
   } /* endfor */
   if (j == 0) {
      pHwmcaDataType = (HWMCA_DATATYPE_P)NULL;
   } else {
      aHwmcaDataType[j-1].pNext = (HWMCA_DATATYPE_P)NULL;
      pHwmcaDataType = aHwmcaDataType;
   } /* endif */
   usRc = HwmcaCommand(&tHwmcaInitialize,argv[3],argv[4],
                       pHwmcaDataType,(ULONG)HWMCAAPI_TIMEOUT);
   if (!usRc) {
      printf("HwmcaCommand request was successful; waiting for the command response event.\n");
      while (!usRc) {
         usRc = HwmcaWaitEvent(&tHwmcaInitialize,cEventBuf,sizeof(cEventBuf),
                               &ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
         if (!usRc) { /* WaitEvent request successful */
            if (ulBytesNeeded <= sizeof(cEventBuf)) {
               parse_and_print_get((HWMCA_DATATYPE_P)cEventBuf);
            } else {
               printf("Event buffer not large enough!\n");
            } /* endif */
```

```
         } else {
            printf("Error in HwmcaWaitEvent call return code = %ld\n",usRc);
         } /* endif */
      } /* endwhile */
   } else {
      printf("Error in HwmcaCommand call return code = %ld\n",usRc);
   } /* endif */
   break;
case 5: /* WaitEvent request */
   usRc = 0;
   while ((!usRc) && (argc >= 5)) {
      usRc = HwmcaWaitEvent(&tHwmcaInitialize,cEventBuf,sizeof(cEventBuf),
                            &ulBytesNeeded,(ULONG)atol(argv[4]));
      if (!usRc) { /* WaitEvent request successful */
         if (ulBytesNeeded <= sizeof(cEventBuf)) {
            parse_and_print_get((HWMCA_DATATYPE_P)cEventBuf);
         } else {
            printf("Event buffer not large enough!\n");
         } /* endif */
      } else {
         printf("Error in HwmcaWaitEvent call return code = %ld\n",usRc);
      } /* endif */
   } /* endwhile */
   break;
case 6: /* Build Id request */
   pszAttribute = pszGroupName = pszObjectName = (PSZ)NULL;
   switch (argc) {
      case 7:
         if (strlen(argv[6])) {
            pszObjectName = argv[6];
         } /* endif */
      case 6:
         if (strlen(argv[5])) {
            pszGroupName = argv[5];
         } /* endif */
      case 5:
         if (strlen(argv[4])) {
            pszAttribute = argv[4];
         } /* endif */
         break;
      default:
         break;
   } /* endswitch */
   usRc = HwmcaBuildId(szOID,argv[3],pszAttribute,pszGroupName,pszObjectName);
   if (!usRc) {
      printf("HwmcaBuildId build object identifier %s.\n",szOID);
      ulLength     = HWMCA_DATATYPE_SIZE;
      pHwmcaDataType = (HWMCA_DATATYPE_P)NULL;
      memset(&tHwmcaDataType,'\0',HWMCA_DATATYPE_SIZE);
      usRc = HwmcaGet(&tHwmcaInitialize,szOID,&tHwmcaDataType,
                      ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
      if (!usRc) { /* Data returned from HwmcaGet */
```

```
      /* Need a larger buffer for the Get request */
      if (ulBytesNeeded > ulLength) {
         pHwmcaDataType = (HWMCA_DATATYPE_P)(malloc(ulBytesNeeded));
         if (pHwmcaDataType) {
            memset(pHwmcaDataType,'\0',ulBytesNeeded);
            ulLength = ulBytesNeeded;
            usRc = HwmcaGet(&tHwmcaInitialize,szOID,pHwmcaDataType,
                            ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
            if (!usRc) { /* Get request successful */
               /* Check if it is a Group contents Get */
               if (strstr(argv[3],HWMCA_GROUP_CONTENTS_SUFFIX)) {
                  parse_and_print_get_group_contents(pHwmcaDataType);
               } else {
                  parse_and_print_get(pHwmcaDataType);
               } /* endif */
               free(pHwmcaDataType);
            } else {
               printf("Error in HwmcaGet call return code = %ld\n",usRc);
            } /* endif */
         } else {
            printf("Error in allocating %ld bytes for an HwmcaGet call",
                   ulBytesNeeded);
         } /* endif */
      } else {
         /* Check if it is a Group contents Get */
         if (strstr(argv[3],HWMCA_GROUP_CONTENTS_SUFFIX)) {
            parse_and_print_get_group_contents(&tHwmcaDataType);
         } else {
            parse_and_print_get(&tHwmcaDataType);
         } /* endif */
      } /* endif */
   } else {
      printf("Error in HwmcaGet call return code = %ld\n",usRc);
   } /* endif */
} else {
   printf("Error in HwmcaBuildId call return code = %ld\n",usRc);
} /* endif */
break;
```

```
case 7: /* Build Attribute Id request */
   usRc = HwmcaBuildAttributeId(szOID,argv[3],argv[4]);
   if (!usRc) {
      printf("HwmcaBuildAttributeId build object identifier %s.\n",szOID);
      ulLength     = HWMCA_DATATYPE_SIZE;
      pHwmcaDataType = (HWMCA_DATATYPE_P)NULL;
      memset(&tHwmcaDataType,'\0',HWMCA_DATATYPE_SIZE);
      usRc = HwmcaGet(&tHwmcaInitialize,szOID,&tHwmcaDataType,
                      ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
      if (!usRc) { /* Data returned from HwmcaGet */
         /* Need a larger buffer for the Get request */
         if (ulBytesNeeded > ulLength) {
            pHwmcaDataType = (HWMCA_DATATYPE_P)(malloc(ulBytesNeeded));
            if (pHwmcaDataType) {
               memset(pHwmcaDataType,'\0',ulBytesNeeded);
               ulLength = ulBytesNeeded;
               usRc = HwmcaGet(&tHwmcaInitialize,szOID,pHwmcaDataType,
                               ulLength,&ulBytesNeeded,(ULONG)HWMCAAPI_TIMEOUT);
               if (!usRc) { /* Get request successful */
                  /* Check if it is a Group contents Get */
                  if (strstr(argv[3],HWMCA_GROUP_CONTENTS_SUFFIX)) {
                     parse_and_print_get_group_contents(pHwmcaDataType);
                  } else {
                     parse_and_print_get(pHwmcaDataType);
                  } /* endif */
                  free(pHwmcaDataType);
               } else {
                  printf("Error in HwmcaGet call return code = %ld\n",usRc);
               } /* endif */
            } else {
               printf("Error in allocating %ld bytes for an HwmcaGet call",
                      ulBytesNeeded);
            } /* endif */
         } else {
            /* Check if it is a Group contents Get */
            if (strstr(argv[3],HWMCA_GROUP_CONTENTS_SUFFIX)) {
               parse_and_print_get_group_contents(&tHwmcaDataType);
            } else {
               parse_and_print_get(&tHwmcaDataType);
            } /* endif */
         } /* endif */
      } else {
         printf("Error in HwmcaGet call return code = %ld\n",usRc);
      } /* endif */
   } else {
      printf("Error in HwmcaBuildId call return code = %ld\n",usRc);
   } /* endif */
   break;
```

```
         default:
           break;
         } /* endswitch */
         usRc = HwmcaTerminate(&tHwmcaInitialize,(ULONG)HWMCAAPI_TIMEOUT);
         if (!usRc) {  /* Terminate with HWMCA API server successful */
            printf("HwmcaTerminate socket                = %ld\n",
                   tHwmcaInitialize.protocol.snmp.iAgentSocket);
            printf("HwmcaTerminate agent Internet address = %x\n",
                   tHwmcaInitialize.protocol.snmp.ulInetAddr);
         } else {
            printf("Error in HwmcaTerminate call return code = %ld\n",usRc);
         } /* endif */
      } else {
         printf("Error in HwmcaInitialize call return code = %ld\n",usRc);
      } /* endif */
   } /* endif */
} else {
   usfContinue = FALSE;
} /* endif */
if (!usfContinue) {
```

```
      printf("*********************************************************\n");
      printf("*** Program requires the following parameters:       ***\n");
      printf("***                                                   ***\n");
      printf("***    Type of request: (use 1 - 7 as the parameter  ***\n");
      printf("***                      for the type of request)    ***\n");
      printf("***       1 - Get               request              ***\n");
      printf("***       2 - Get-Next          request              ***\n");
      printf("***       3 - Set               request              ***\n");
      printf("***       4 - Command           request              ***\n");
      printf("***       5 - Wait Event        request              ***\n");
      printf("***       6 - Build Id          request              ***\n");
      printf("***       7 - Build Attribute Id request             ***\n");
      printf("***                                                   ***\n");
      printf("***    Internet address of the Console Application   ***\n");
      printf("***        9.130.1.133                                ***\n");
      printf("***                                                   ***\n");
      printf("***    Request specific parameters:                   ***\n");
      printf("***       For a Get or Get-Next request:              ***\n");
      printf("***          Object ID (1.3.6.1.etc)                  ***\n");
      printf("***                                                   ***\n");
      printf("***       For a Set request:                          ***\n");
      printf("***          Object ID (1.3.6.1.etc)                  ***\n");
      printf("***          Set data type (2-integer,4-string,etc)   ***\n");
      printf("***          Set data                                 ***\n");
      printf("***                                                   ***\n");
      printf("***       For a Command request:                      ***\n");
      printf("***          Target Object ID (1.3.6.1.etc)           ***\n");
      printf("***          Command Object ID (1.3.6.1.etc)          ***\n");
      printf("***                                                   ***\n");
      printf("***       For a Wait Event request:                   ***\n");
      printf("***          Event mask                               ***\n");
      printf("***          Timeout value in milliseconds            ***\n");
      printf("***             (-1 --> forever)                      ***\n");
      printf("***                                                   ***\n");
      printf("***       For a Build Id request:                     ***\n");
      printf("***          Object ID Prefix (1.3.6.1.etc)           ***\n");
      printf("***          Attribute suffix (optional)              ***\n");
      printf("***          Group name       (optional)              ***\n");
      printf("***          Object name      (optional)              ***\n");
      printf("***                                                   ***\n");
      printf("***       For a Build Attribute Id request:           ***\n");
      printf("***          Object ID (1.3.6.1.etc)                  ***\n");
      printf("***          Attribute suffix                         ***\n");
      printf("*********************************************************\n");
   } /* endif */

}  /* end main */
```

```
VOID parse_and_print_get(HWMCA_DATATYPE_P pHwmcaDataType)
{
   HWMCA_DATATYPE_P   pLoopHwmcaDataType;

   pLoopHwmcaDataType = pHwmcaDataType;
   while (pLoopHwmcaDataType) {
      switch (pLoopHwmcaDataType->ucType) {
      case HWMCA_TYPE_SEQUENCE:
         break;
      case HWMCA_TYPE_INTEGER:
         printf("HWMCA_TYPE_INTEGER returned size = %d and pData = %d\n",
                pLoopHwmcaDataType->ulLength,*((PINT)(pLoopHwmcaDataType->pData)));
         break;
      case HWMCA_TYPE_OCTETSTRING:
         printf("HWMCA_TYPE_OCTETSTRING returned size = %d and pData = %s\n",
                pLoopHwmcaDataType->ulLength,pLoopHwmcaDataType->pData);
         break;
      case HWMCA_TYPE_NULL:
         printf("HWMCA_TYPE_NULL returned size = %d\n",pLoopHwmcaDataType->ulLength);
         break;
      case HWMCA_TYPE_OBJECTID:
         printf("HWMCA_TYPE_OBJECTID returned size = %d and pData = %s\n",
                pLoopHwmcaDataType->ulLength,pLoopHwmcaDataType->pData);
         break;
      case HWMCA_TYPE_IPADDRESS:
         printf("HWMCA_TYPE_IPADDRESS returned size = %d and pData = %x\n",
                pLoopHwmcaDataType->ulLength,*((PINT)(pLoopHwmcaDataType->pData)));
         break;
      default:
         printf("UNKNOWN Data type returned = %d\n",pLoopHwmcaDataType->ucType);
        break;
      } /* endswitch */
      pLoopHwmcaDataType = pLoopHwmcaDataType->pNext;
   } /* endwhile */

} /* end of parse_and_print_get */
```

```
VOID parse_and_print_get_group_contents(HWMCA_DATATYPE_P pHwmcaDataType)
{
   PSZ              pszGroupContents;
   PUCHAR           pBlank;
   HWMCA_DATATYPE_P   pLoopHwmcaDataType;

   pLoopHwmcaDataType = pHwmcaDataType;
   while (pLoopHwmcaDataType) {
      switch (pLoopHwmcaDataType->ucType) {
      case HWMCA_TYPE_OBJECTID:
         printf("HWMCA_TYPE_OBJECTID returned size = %d and pData = %s\n",
                pLoopHwmcaDataType->ulLength,pLoopHwmcaDataType->pData);
         break;
      case HWMCA_TYPE_OCTETSTRING:
         printf("HWMCA_TYPE_OCTETSTRING returned size = %d and pData = %s\n",
                pLoopHwmcaDataType->ulLength,pLoopHwmcaDataType->pData);
         pszGroupContents = (PSZ)pLoopHwmcaDataType->pData;
         pBlank = pszGroupContents;
         pBlank = strchr(pBlank,' ');
         while (pBlank) {
            *pBlank = '\0';
            printf("Group contents Object ID = %s\n",pszGroupContents);
            pBlank++;
            pszGroupContents = pBlank;
            pBlank = strchr(pBlank,' ');
         } /* endwhile */
         printf("Group contents Object ID = %s\n",pszGroupContents);
         break;
      case HWMCA_TYPE_NULL:
         printf("HWMCA_TYPE_NULL returned size = %d\n",pLoopHwmcaDataType->ulLength);
         break;
      default:
         printf("UNKNOWN Data type returned = %d\n",pLoopHwmcaDataType->ucType);
        break;
      } /* endswitch */
      pLoopHwmcaDataType = pLoopHwmcaDataType->pNext;
   } /* endwhile */

} /* end of parse_and_print_get_group_contents */
```

Chapter 4. Console application managed objects

This chapter contains definitions of the objects the Console application manages. Each object contains the following:

- **Object Type**
- **Object Name Bindings:** Shows the name of the base object that is used in the Management Commands API.
- **Object Attributes:** Describes each attribute an object contains and the operations supported against that attribute. The operations supported are:

  Get: Retrieve the current attribute value of an object

  Set: The attribute value of an object

  It also shows the attribute name of an object (SNMP MIB name) that is used in the management APIs.

  Important information about object attributes: Unless otherwise specified in Appendix E, "Object Attribute Availability," on page 215, it can be assumed that each object attribute described in this chapter is valid for any level of object. For any object attribute that is not valid for all levels, Table 1 on page 215 defines the level of objects required for the attribute.
- **Object Relationship:** Describes any pertinent relationships the object contains with other objects.
- **Commands that can be performed on that object:** Describes each command that is valid for the object and also shows the name of the command that is used in the Management Commands API when requesting a command to be performed on the object. For the SNMP version, the command name is called the SNMP MIB Name.
- **Emitted object asynchronous notifications:** Describes the significant notifications an object will emit to a registered application.

Console application object identifier conventions

All the objects managed by the Console application follow the same object identifier naming scheme. The naming scheme used by the Console breaks the object identifiers into four distinct portions:

prefix.attribute.group.object

The meanings and options for each of these portions are described in the following pages:

prefix

This portion of the object identifier must be one of the following:

1.3.6.1.4.1.2.6.42.0

An attribute of the Console object or the Console object itself.

1.3.6.1.4.1.2.6.42.1

An attribute of the Defined CPCs group object or the Defined CPCs group object itself.

1.3.6.1.4.1.2.6.42.1.0

An attribute of a Defined CPC object or a Defined CPC object itself.

1.3.6.1.4.1.2.6.42.2

An attribute of the CPC Images group object or the CPC Images group object itself.

1.3.6.1.4.1.2.6.42.2.0

An attribute of a CPC Image object or a CPC Image object itself.

1.3.6.1.4.1.2.6.42.3
An attribute of a user-defined group object or a user-defined group object itself.

1.3.6.1.4.1.2.6.42.3.0
An attribute of an object contained within a user-defined group object or an object contained within a user-defined group object itself.

1.3.6.1.4.1.2.6.42.4
A Console application command object.

1.3.6.1.4.1.2.6.42.5
An attribute of a Reset Activation Profile or a Reset Activation Profile object itself.

1.3.6.1.4.1.2.6.42.6
An attribute of an Image Activation Profile or an Image Activation Profile object itself.

1.3.6.1.4.1.2.6.42.7
An attribute of a Load Activation Profile or a Load Activation Profile object itself.

1.3.6.1.4.1.2.6.42.8
An attribute of a Group Activation Profile or a Group Activation Profile object itself.

1.3.6.1.4.1.2.6.42.9.0
An attribute of a Capacity Record object or a Capacity Record object itself.

1.3.6.1.4.1.2.6.42.10
An attribute of the Managed z/VM® Virtual Machines group object or the Managed z/VM Virtual Machines group object itself.

1.3.6.1.4.1.2.6.42.10.0
An attribute of a z/VM Virtual Machine object or a z/VM Virtual Machine object itself.

attribute
This portion of the object identifier is used when specifying an object identifier for an attribute of an object. It is optional and when not specified results in an object identifier for the object itself.

group
This portion of the object identifier is used to uniquely specify which user-defined group this object identifier pertains to. It is optional and should only be used for the following object identifiers:

- User-defined groups
- User-defined group attributes
- Objects contained within user-defined groups
- Attributes of objects contained within user-defined groups
- Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects (in this case the group value is used to identify the CPC object that the activation profile pertains to)
- Attributes of Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects (in this case the group value is used to identify the CPC object that the activation profile attribute pertains to).

This value is generated using the name attribute of the group object.

object
This portion of the object identifier is used to uniquely specify which object within a group this object identifier pertains to. It is optional and should only be used for the following object identifiers:

- Objects contained within a group
- Attributes of objects contained within a group
- Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects
- Attributes of Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects.

This value is generated using the name attribute of the object.

Console application object

Console application name bindings

Console object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.x.x

Where ***x.x*** equals the attribute identifier for the object.

Console attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.1.0

SNA address

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING (The OCTET string returned contains the SNA address in the form NetId.Name)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.16.0

Group contents

Get: Null terminated collection of blank separated object identifier strings.

- Data type(s) returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- Defined CPCs Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1
- CPC Images Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2
- CPC User Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.*
- CPC Images User Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.*
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.23.0

Version

Get: The version number for the console.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.151.0

Internet Protocol (IP) addresses

Get: A null terminated list of blank separated IP addresses being used by the console.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.161.0

Engineering Change (EC)/Microcode Level (MCL)

Get: An XML string that describes the EC and MCL levels for the console.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING

  Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.0.162.0

Console application commands

Hardware message refresh
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.11 (HWMCA_HW_MESSAGE_REFRESH_COMMAND)

Hardware message delete
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.13 (HWMCA_HW_MESSAGE_DELETE_COMMAND)

Shutdown/Restart
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.22 (HWMCA_SHUTDOWN_RESTART_COMMAND)

Console application notifications

Security log event (HWMCA_EVENT_SECURITY_EVENT)
- An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the security log.
- An HWMCA_TYPE_OCTETSTRING that specifies the text of the security log.

Console application started (HWMCA_EVENT_STARTED)
This event has no additional data.

Console application ended (HWMCA_EVENT_ENDED)
Used to notify the application that the Console application is ending.

The additional data for this event consists of the following object identifier/value pairs:
1. An HWMCA_TYPE_INTEGER that specifies the reason for the event. The possible values are:
   - HWMCA_ENDED_USER - the event was initiated by a user,
   - HWMCA_ENDED_AUTOMATION - the event was initiated by automation, or
   - HWMCA_ENDED_OTHER - the event was initiated by the Console application itself (for example, recovery action, change management, etc.)
2. An HWMCA_TYPE_OCTETSTRING that specifies the name of the Console application component that caused the event.
3. An HWMCA_TYPE_INTEGER that specifies the shutdown type for the event. The possible values are:
   - HWMCA_SHUTDOWN_CONSOLE - the console has been shut down and will take manual intervention to be restarted,
   - HWMCA_RESTART_APPLICATION - the console application has been stopped and will automatically be restarted, or
   - HWMCA_RESTART_CONSOLE - the console has been stopped and will automatically be restarted.
4. An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to.

Message (HWMCA_EVENT_MESSAGE)
- An HWMCA_TYPE_INTEGER that specifies that the message is a Console or Optical Network message (HWMCA_HARDWARE_MESSAGE).
- An HWMCA_TYPE_OCTETSTRING that specifies a description of the new or refreshed Console or Optical Network message.
- An HWMCA_TYPE_INTEGER that specifies whether the message is a new (HWMCA_FALSE) or refresh message (HWMCA_TRUE).
- An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the new or refresh message.
- An HWMCA_TYPE_OCTETSTRING that specifies the name(s) of the CPC Image object(s) associated with the object that generated the new or refresh message.

Message deletion (HWMCA_EVENT_HARDWARE_MESSAGE_DELETE)

- An HWMCA_TYPE_INTEGER that specifies that the message being deleted is a Console or Optical Network message (HWMCA_HARDWARE_MESSAGE).
- An HWMCA_TYPE_OCTETSTRING that specifies the message text of the Console or Optical Network message being deleted.
- An HWMCA_TYPE_INTEGER which is always HWMCA_FALSE for this event.
- An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the message being deleted.
- An HWMCA_TYPE_OCTETSTRING that specifies the name(s) of the CPC Image object(s) associated with the object for which the message is being deleted.

Group

Group name bindings

Defined CPCs group object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1

CPC images group object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2

CPC user group object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3

CPC images user group object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3

Managed z/VM virtual machines group object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10

Group attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Defined CPCs Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.1.0
- CPC Images Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.1.0
- CPC User Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.1.0.*
- CPC Images User Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.1.0.*
- Managed z/VM Virtual Machines Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.1.0.*

Status error

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE

  Contains one or more objects that are in a state which is not an acceptable status.

  HWMCA_FALSE

  All objects contained within the group are in an acceptable status state.
- Defined CPCs Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.7.0
- CPC Images Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.7.0
- CPC User Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.7.0.*
- CPC Images User Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.7.0.*
- Managed z/VM Virtual Machines Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.7.0.*

Busy

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object in a busy state (currently performing a task).

  HWMCA_FALSE
  Object not in a busy state.
- Defined CPCs Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.8.0
- CPC Images Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.8.0
- CPC User Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.8.0.*
- CPC Images User Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.8.0.*
- Managed z/VM Virtual Machines Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.8.0.*

Object type

Get: This returns the type of object the object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  One of the following values:
  - HWMCA_CPC_GROUP
  - HWMCA_CPC_IMAGE_GROUP
  - HWMCA_CPC_USER_GROUP
  - HWMCA_CPC_IMAGE_USER_GROUP
  - HWMCA_VM_GROUP
- Defined CPCs Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.22.0
- CPC Images Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.22.0
- CPC User Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.22.0.*
- CPC Images User Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.22.0.*
- Managed z/VM Virtual Machines Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.22.0.*

Contents

Get: Null terminated collection of blank separated object identifier strings.

- Data type(s) returned on Get: HWMCA_TYPE_OCTETSTRING
- Defined CPCs Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.23.0
- CPC Images Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.23.0
- CPC User Group Object SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.23.0.*
- CPC Images User Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.3.23.0.*
- Managed z/VM Virtual Machines Group SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.23.0.*

Note: In some cases the size of the data associated with this attribute is larger than what many applications can traditionally handle. In this situation the same information can be determined by issuing a series of GetNext requests to build the collection of object identifier strings.

Group commands

H/W (CPC) group

Commands that can be performed on this group are the same as the commands listed in the Defined CPC object's definition in "Defined CPC commands" on page 90 except for the HWMCA_HW_MESSAGE_REFRESH_COMMAND and HWMCA_HW_MESSAGE_DELETE_COMMAND commands.

CPC image group

Commands that can be performed on this group are the same as the commands listed in CPC image object's definition in "CPC image commands" on page 106. However, the send operating system 1.3.6.1.4.1.2.6.42.4.3 (HWMCA_SEND_OPSYS_COMMAND) listed in this chapter is not valid for sending to a group.

CF image group

Commands that can be performed on this group are the same as the commands listed in CF image object's definition in "Coupling facility commands" on page 115. However, the send operating system 1.3.6.1.4.1.2.6.42.4.3 (HWMCA_SEND_OPSYS_COMMAND) listed in this chapter is not valid for sending to a group.

Group notifications

Object created (HWMCA_EVENT_CREATED)

This event has no additional data. The object identifier can be used with the *HwmcaGet* to get any data required for this newly created object.

Object destruction (HWMCA_EVENT_DESTROYED)

This event has no additional data.

Defined CPC

Defined CPC name bindings

CPC object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.x.x.*

Where ***x.x.*** equals the attribute identifier for the object and * equals a unique number for that specific instance of the CPC Object.

Defined CPC attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.1.0.*

Status error

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object is in a state which is not an acceptable status.

  HWMCA_FALSE
  Object is in an acceptable status state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.7.0.*

Busy

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object is in a busy state (currently performing a task).

  HWMCA_FALSE
  Object is not in a busy state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.8.0.*

Message indicator

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object has a hardware message.

HWMCA_FALSE
Object does not have a hardware message.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.9.0.*

Status

Get:
- Data type returned on Get: HWMCA_TYPE_INTEGER

  One of the following bit values will be set to on:
  - HWMCA_STATUS_OPERATING
  - HWMCA_STATUS_NOT_OPERATING
  - HWMCA_STATUS_NO_POWER
  - HWMCA_STATUS_EXCEPTIONS
  - HWMCA_STATUS_STATUS_CHECK
  - HWMCA_STATUS_SERVICE
  - HWMCA_STATUS_LINKNOTACTIVE
  - HWMCA_STATUS_POWERSAVE
  - HWMCA_STATUS_SERVICE_REQ
  - HWMCA_STATUS_DEGRADED
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.10.0.*

Acceptable status

Get/Set:
- Data type returned on Get: HWMCA_TYPE_INTEGER
- Data type for Set: HWMCA_TYPE_INTEGER

  One or more of the following bit values will be set to on:
  - HWMCA_STATUS_OPERATING
  - HWMCA_STATUS_NOT_OPERATING
  - HWMCA_STATUS_NO_POWER
  - HWMCA_STATUS_EXCEPTIONS
  - HWMCA_STATUS_STATUS_CHECK
  - HWMCA_STATUS_SERVICE
  - HWMCA_STATUS_LINKNOTACTIVE
  - HWMCA_STATUS_POWERSAVE
  - HWMCA_STATUS_SERVICE_REQ
  - HWMCA_STATUS_DEGRADED
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.11.0.*

IML mode

Get:
- Data type returned on Get: HWMCA_TYPE_INTEGER
  - HWMCA_IML_ESA390_MODE
  - HWMCA_IML_LPAR_MODE
  - HWMCA_IML_ESA390TPF_MODE
  - HWMCA_IML_LINUX_MODE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.12.0.*

Activation profile name

Get/Set (Reset or Load profile):
- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A maximum length of 17 bytes is allowed for the activation profile name, including the null terminator.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.13.0.*

Last used activation profile

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.14.0.*

Internet address

Get:

- Data type returned on Get: HWMCA_TYPE_IPADDRESS
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.15.0.*

SNA address

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING (The OCTET string returned will contain the SNA address in the form NetId.Name.)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.16.0.*

Computer (machine) model

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.17.0.*

Computer (machine) type

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.18.0.*

Computer (machine) serial

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.19.0.*

CPC serial number

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.20.0.*

CPC identifier

Get: Node descriptor identifier calculated by using location within computer (machine).

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.21.0.*

Object type

Get: This returns the type of object the object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER
  HWMCA_CPC_OBJECT
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.22.0.*

List of reset activation profiles

Get: This returns a null terminated collection of blank separated object identifiers for each Reset Activation profile.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.24.0.*

List of image activation profiles

Get: This returns a null terminated collection of blank separated object identifiers for each Image Activation profile.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.25.0.*

List of load activation profiles

Get: This returns a null terminated collection of blank separated object identifiers for each Load Activation profile.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.26.0.*

CBU installed

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  CBU is installed.

  HWMCA_FALSE
  CBU is not installed.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.32.0.*

CBU activated

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  CBU is activated.

  HWMCA_FALSE
  CBU is not activated.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.33.0.*

CBU activation date

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.34.0.*

CBU expiration date

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.35.0.*

Number of CBU tests left

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.36.0.*

Real CBU activation available

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Real CBU is available.

  HWMCA_FALSE
  Real CBU is not available.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.37.0.*

Reserve ID

Note: This attribute is available only on a Support Element console.

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING (The OCTET string returned contains the name of the application that currently holds the reserve. A zero length string implies that no application holds the reserve.)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.44.0.*

Service required indicator

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Service Required indicator is on.

  HWMCA_FALSE
  Service Required indicator is not on.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.46.0.*

Degraded indicator

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER
  - HWMCA_NOT_DEGRADED
  - HWMCA_DEGRADED_MEM
  - HWMCA_DEGRADED_MBA
  - HWMCA_DEGRADED_NODE
  - HWMCA_DEGRADED_RING
  - HWMCA_DEGRADED_CBU
  - HWMCA_DEGRADED_MRU
  - HWMCA_DEGRADED_AMBIENT
  - HWMCA_DEGRADED_MRU_IML
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.47.0.*

CBU enabled

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  CBU is enabled.

  HWMCA_FALSE
  CBU is not enabled.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.48.0.*

List of cluster members

Get: This returns a null terminated collection of blank separated SNA addresses for all other Support Elements considered to be within the same cluster.

Note: This attribute is available only when targeting a Support Element.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.50.0.*

Processor running time type

Get/Set: Defines whether the processor running time is dynamically determined by the system or set to a constant value for the Defined CPC object.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  0 (HWMCA_DETERMINED_SYSTEM)
  The processor running is dynamically determined by the system.

  1 (HWMCA_DETERMINED_USER)
  The processor running time is set to a constant value.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.78.0.*

Processor running time

Get/Set: Defines the amount of continuous time allowed for logical processors to perform jobs on shared processors for the Defined CPC object.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  A value 1 - 100 for the user-defined processor running time.

  Note: This value can only be set if the processor running time type is set to 1 (HWMCA_DETERMINED_USER).
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.79.0.*

End timeslice if CPC image enters a wait state

Get/Set: Defines whether CPC Images lose their share of running time when they enter a wait state.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that a CPC Image should lose its share of running time when it enters a wait state.

  HWMCA_FALSE
  Indicates that a CPC Image should not lose its share of running time when it enters a wait state.

  Note: This value can only be set if the processor running time type is set to 1 (HWMCA_DETERMINED_USER).
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.80.0.*

On/Off Capacity on Demand (On/Off CoD) installed

Get: Defines whether On/Off Capacity on Demand is installed for the Defined CPC object.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  On/Off CoD is installed.

  HWMCA_FALSE
  On/Off CoD is not installed.

  Note: The attribute On/Off Capacity on Demand (On/Off CoD) Installed and attribute On/Off Capacity on Demand (On/Off CoD) Activated always have the same value, either HWMCA_TRUE or HWMCA_FALSE.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.87.0.*

On/Off Capacity on Demand (On/Off CoD) activated

Get: Defines whether On/Off Capacity on Demand is currently activated for the Defined CPC object.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  On/Off CoD is activated.

  HWMCA_FALSE
  On/Off CoD is not activated

  Note: The attribute On/Off Capacity on Demand (On/Off CoD) Installed and attribute On/Off Capacity on Demand (On/Off CoD) Activated always have the same value, either HWMCA_TRUE or HWMCA_FALSE.
  .
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.88.0.*

On/Off Capacity on Demand (On/Off CoD) enabled

Get: Defines whether On/Off Capacity on Demand is enabled for the Defined CPC object.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  On/Off CoD is enabled.

  HWMCA_FALSE
  On/Off CoD is not enabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.89.0.*

On/Off Capacity on Demand (On/Off CoD) activation date

Get: Defines the time stamp for when On/Off CoD was activated for the Defined CPC object.

- Data type for Get: HWMCA_TYPE_OCTETSTRING

  A time stamp string describing when On/Off CoD was activated or an empty string if On/Off CoD is not activated.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.90.0.*

List of group profiles

Get: This returns a null terminated collection of blank separated object identifiers for each Group profile.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.91.0.*

Temporary capacity records

Get: A blank separated list of SNMP object identifiers for the installed temporary capacity records.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.119.0.*

Permanent software model

Get: The software model based on the permanent processors.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.120.0.*

Permanent plus billable software model

Get: The software model based on the permanent plus billable processors.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.121.0.*

Permanent plus all temporary software model

Get: The software model based on the permanent plus all temporary processors.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.122.0.*

Permanent MSU

Get: The MSU value associated with the software model based on the permanent processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.123.0.*

Permanent plus billable MSU

Get: The MSU value associated with the software model based on the permanent plus billable processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.124.0.*

Permanent plus all temporary MSU

Get: The MSU value associated with the software model based on the permanent plus all temporary processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.125.0.*

General purpose processors

Get: The count of general purpose processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.126.0.*

Service assist processors

Get: The count of service assist processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.127.0.*

Application Assist Processor (AAP) processors

Get: The count of Application Assist Processor (AAP) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.128.0.*

Integrated Facility for Linux (IFL) processors

Get: The count of Integrated Facility for Linux (IFL) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.129.0.*

Internal Coupling Facility (ICF) processors

Get: The count of Internal Coupling Facility (ICF) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.130.0.*

Integrated Information Processors (zIIP) processors

Get: The count of Integrated Information Processors (zIIP) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.131.0.*

Defective processors

Get: The count of defective processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.132.0.*

Spare processors

Get: The count of spare processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.133.0.*

Pending processors

Get: The count of processors pending activation.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.134.0.*

Temporary capacity change allowed

Get: This value is used to determine if API applications are allowed to make changes to temporary capacity.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  API applications are allowed to perform temporary capacity changes.

  HWMCA_FALSE
  API applications are not allowed to perform temporary capacity changes.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.149.0.*

Version

Get: The version number for the Defined CPC.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.151.0.*

Internet Protocol (IP) addresses

Get: A null terminated list of blank separated IP addresses being used by the defined CPC object.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.161.0

Engineering Change (EC)/Microcode Level (MCL)

Get: An XML string that describes the EC and MCL levels for the defined CPC object. For more information, see Appendix F, "XML descriptions," on page 219.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.162.0

Automatic switch enabled

Get: This value is used to determine if automatic switching between primary and alternate Support Elements is enabled for the Defined CPC object.

- Data type for Get: HWMCA_TYPE_INTEGER
  HWMCA_TRUE
  Automatic switching is enabled.
  HWMCA_FALSE
  Automatic switching is disabled.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.163.0.*

Server Time Protocol (STP) configuration

Get: An XML string that describes the STP configuration for the defined CPC object. For more information, see Appendix F, "XML descriptions," on page 219.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.165.0

Pending General Purpose Processors

Get: The count of pending general purpose processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.175.0.*

Pending Service Assist Processors

Get: The count of pending service assist processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.176.0.*

Pending Application Assist Processor (AAP) Processors

Get: The count of pending Application Assist Processor (AAP) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.177.0.*

Pending Integrated Facility for Linux (IFL) Processors

Get: The count of pending Integrated Facility for Linux (IFL) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.178.0.*

Pending Internal Coupling Facility (ICF) Processors

Get: The count of pending Internal Coupling Facility (ICF) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.179.0.*

Pending Integrated Information Processors (zIIP) Processors

Get: The count of pending Integrated Information Processors (zIIP) processors.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.1.0.180.0.*

Defined CPC relationships

Cluster (String)

A CPC is a member of a cluster. There are one or more CPCs per cluster.

Support Element

A CPC has a one-to-one relationship with a Support Element (provider of services).

CF/CPC image

A CPC contains one or more images. This is determined by the activation reset profile.

Defined CPC commands

Activate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.1 (HWMCA_ACTIVATE_COMMAND)

Deactivate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.2 (HWMCA_DEACTIVATE_COMMAND)

Hardware message refresh

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.11 (HWMCA_HW_MESSAGE_REFRESH_COMMAND)

Hardware message delete

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.13 (HWMCA_HW_MESSAGE_DELETE_COMMAND)

Activate CBU

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.14 (HWMCA_ACTIVATE_CBU_COMMAND)

Undo CBU

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.15 (HWMCA_UNDO_CBU_COMMAND)

Import profiles

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.16 (HWMCA_IMPORT_PROFILE_COMMAND)

Export profiles
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.17 (HWMCA_EXPORT_PROFILE_COMMAND)

Reserve
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.18 (HWMCA_RESERVE_COMMAND)

Activate On/Off Capacity on Demand (On/Off CoD)
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.23 (HWMCA_ACTIVATE_OOCOD_COMMAND)

Undo On/Off Capacity on Demand (On/Off CoD)
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.24 (HWMCA_UNDO_OOCOD_COMMAND)

Add temporary capacity
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.25 (HWMCA_ADD_CAPACITY_COMMAND)

Remove temporary capacity
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.26 (HWMCA_REMOVE_CAPACITY_COMMAND)

Swap Current Time Server
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.27(HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND)

Set STP configuration
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.28(HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND)

Change STP-only CTN
SNMP MIB Name:
1.3.6.1.4.1.2.6.42.4.29(HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_COMMAND)

Join STP-only CTN
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.30(HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND)

Leave STP-only CTN
SNMP MIB Name:
1.3.6.1.4.1.2.6.42.4.31(HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_COMMAND)

Defined CPC notifications

Message (HWMCA_EVENT_MESSAGE)
An HWMCA_TYPE_INTEGER that specifies whether the message is a hardware or operating system message (HWMCA_HARDWARE_MESSAGE or HWMCA_OPSYS_MESSAGE).

For hardware messages:

- An HWMCA_TYPE_OCTETSTRING that specifies a description of the new or refreshed hardware message.
- An HWMCA_TYPE_INTEGER that specifies whether the message is a new (HWMCA_FALSE) or refresh message (HWMCA_TRUE).
- An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the new or refresh message.
- An HWMCA_TYPE_OCTETSTRING that specifies the name(s) of the CPC Image object(s) associated with the object that generated the new or refresh message.

Message deletion (HWMCA_EVENT_HARDWARE_MESSAGE_DELETE)
- An HWMCA_TYPE_INTEGER that specifies that the message being deleted is a CPC-related hardware message (HWMCA_HARDWARE_MESSAGE).
- An HWMCA_TYPE_OCTETSTRING that specifies the message text of the hardware message being deleted.

- An HWMCA_TYPE_INTEGER which is always HWMCA_FALSE for this event.
- An HWMCA_TYPE_OCTETSTRING that specifies the time stamp of the message being deleted.
- An HWMCA_TYPE_OCTETSTRING that specifies the name(s) of the CPC Image object(s) associated with the object for which the message is being deleted.

Status change (HWMCA_EVENT_STATUS_CHANGE)
- An HWMCA_TYPE_INTEGER that specifies the new status value
- An HWMCA_TYPE_INTEGER that specifies the old status value.

Object's name change (HWMCA_EVENT_NAME_CHANGE)
- An HWMCA_TYPE_OCTETSTRING that specifies the new object name
- An HWMCA_TYPE_OCTETSTRING that specifies the old object name.

Object's activation profile change (HWMCA_EVENT_ACTIVATE_PROF_CHANGE)
- An HWMCA_TYPE_OCTETSTRING that specifies the new activation profile name
- An HWMCA_TYPE_OCTETSTRING that specifies the old activation profile name.

Object created (HWMCA_EVENT_CREATED)
This event has no additional data. The object identifier can be used with the *HwmcaGet* to get any data required for this newly created object.

Object destruction (HWMCA_EVENT_DESTROYED)
This event has no additional data.

Object entered an exception state (HWMCA_EVENT_EXCEPTION_STATE)
- An HWMCA_TYPE_INTEGER that specifies whether the object is entering into an exception state (HWMCA_TRUE) or leaving an exception state (HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies the status value for the object.

Capacity change (HWMCA_EVENT_CAPACITY_CHANGE)
- An HWMCA_TYPE_INTEGER that specifies the type of capacity change that occurred.
- An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case a Defined CPC object).

Capacity record change (HWMCA_EVENT_CAPACITY_RECORD_CHANGE)
- An HWMCA_TYPE_INTEGER that specifies the type of capacity record change that occurred.
- An HWMCA_TYPE_OCTETSTRING for the temporary capacity record that has changed.
- An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case a Defined CPC object).

CPC image

CPC image name bindings

Image Object Identifier
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.x.x.*

Where ***x.x.*** equals the attribute identifier for the object and * equals a unique number for that specific instance of the CPC Image Object.

CPC image attributes

Name
Get:
- Data type returned on Get: HWMCA_TYPE_OCTETSTRING

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.1.0.*

Parent's name

Get (CPC's logical name):

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.2.0.*

Operating system name

Get: Name of Operating System running in image, if known.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.3.0.*

Operating system type

Get: Type of Operating System running in image, if known.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.4.0.*

Operating system level

Get: Level of Operating System running in image, if known.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.5.0.*

Sysplex name

Get: Applicable only for MVS™, if known.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.6.0.*

Status error

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object is in a state which is not an acceptable status.

  HWMCA_FALSE
  Object is in an acceptable status state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.7.0.*

Busy

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object in a busy state (currently performing a task)

  HWMCA_FALSE
  Object not in a busy state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.8.0.*

Message indicator

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object has an operating system message.

  HWMCA_FALSE
  Object does not have an operating system message.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.9.0.*

Status

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

One of the following bit values will be set to on:
  – HWMCA_STATUS_OPERATING
  – HWMCA_STATUS_NOT_OPERATING
  – HWMCA_STATUS_NOT_ACTIVATED
  – HWMCA_STATUS_EXCEPTIONS
  – HWMCA_STATUS_STATUS_CHECK
  – HWMCA_STATUS_POWERSAVE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.10.0.*

Acceptable status

Get/Set:

- Data type returned on Get: HWMCA_TYPE_INTEGER
- Data type for Set: HWMCA_TYPE_INTEGER

  One or more of the following bit values will be set to on:
  – HWMCA_STATUS_OPERATING
  – HWMCA_STATUS_NOT_OPERATING
  – HWMCA_STATUS_NOT_ACTIVATED
  – HWMCA_STATUS_EXCEPTIONS
  – HWMCA_STATUS_STATUS_CHECK
  – HWMCA_STATUS_POWERSAVE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.11.0.*

IML/partition activation mode

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  One of the following bit values will be set to on:
  – HWMCA_IML_ESA390_MODE
  – HWMCA_IML_S370_MODE
  – HWMCA_IML_ESA390TPF_MODE
  – HWMCA_IML_CF_PROD_MODE
  – HWMCA_IML_LINUX_MODE
  – HWMCA_IML_ZVM_MODE
  – HWMCA_IML_ZAWARE_MODE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.12.0.*

Activation profile name

Get/Set (Image or Load profile):

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A maximum length of 17 bytes is allowed for the activation profile name, including the null terminator.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.13.0.*

Last used activation profile

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.14.0.*

Object type

Get: This returns the type of object the object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER
  HWMCA_CPC_IMAGE_OBJECT
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.22.0.*

Initial processing weight

Get/Set: The relative amount of shared general purpose processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the relative amount of shared general purpose processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared general purpose processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.30.0.*

Initial processing weight capped

Get/Set: Whether or not the initial processing weight for general purpose processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE (1)
  Indicates that the initial general purpose processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of general purpose processor resources, regardless of the availability of excess general purpose processor resources.

  HWMCA_FALSE (0)
  Indicates that the initial general purpose processor processing weight for the CPC Image is not capped. It represents the share of general purpose processor resources guaranteed to a logical partition when all general purpose processor resources are in use. Otherwise, when excess general purpose processor resources are available, the logical partition can use them if necessary.

  Note: The initial general purpose processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated general purpose processor.

  This attribute and the **Workload manager enabled** attribute are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this attribute it might be necessary to first disable the **Workload manager enabled** attribute. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and the **Workload manager enabled** attribute being disabled automatically in a single operation.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.31.0.*

Minimum processing weight

Get/Set: The minimum relative amount of shared general purpose processor resources allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the minimum relative amount of shared general purpose processor resources allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

A value 1 - 999 and less than or equal to the Initial Processing Weight used to define the minimum relative amount of shared general purpose processor resources allocated to the CPC Image object. A value of zero can also be specified to indicate that there is no minimum value for the processing weight.

Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.38.0.*

Maximum processing weight

Get/Set: The maximum relative amount of shared general purpose processor resources allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the maximum relative amount of shared general purpose processor resources allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 and greater than or equal to the Initial Processing Weight used to define the maximum relative amount of shared general purpose processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.39.0.*

Current processing weight

Get: The relative amount of shared general purpose processor resources currently allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the relative amount of shared general purpose processor resources currently allocated to the CPC Image object.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.41.0.*

Current processing weight capped

Get: Whether or not the current general purpose processing weight is a limit or a target.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the current general purpose processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the current general purpose processing weight for the CPC Image is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

Note: The current general purpose processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.42.0.*

Workload Manager enabled

Get/Set: Whether or not WorkLoad Manager is allowed to change processing weight related attributes.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that WorkLoad Manager is allowed to change processing weight related attributes for this CPC Image object.

  HWMCA_FALSE
  Indicates that WorkLoad Manager is not allowed to change processing weight related attributes for this CPC Image object.

  This attribute and the various capping attributes are mutually exclusive and cannot be enabled at the same time. Therefore in order to enable this attribute it may be necessary to first disable any capping attribute that is currently enabled. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and all capping attributes being disabled automatically in a single operation.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.40.0.*

Defined capacity

Get/Set: The defined capacity expressed in terms of Millions of Service Units (MSU)s per hour. MSUs is a measure of processor resource consumption. The amount of MSUs a logical partition consumes is dependent on the model, the number of logical processors available to the partition, and the amount of time the logical partition is dispatched. The defined capacity value specifies how much capacity the logical partition is to be managed to by WorkLoad Manager for the purpose of software pricing.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  0 No defined capacity is specified for this logical partition.

  1 - nnnn
  Represents the amount of defined capacity specified for this logical partition.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.43.0.*

Cluster name

Get: LPAR cluster name.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.50.0*

Partition identifier

Get: The partition identifier for the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.51.0*

Initial Application Assist Processor processing weight

Get/Set: The relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1 - 999 Represents the relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Application Assist Processor (AAP) processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.60.0.*

Initial Application Assist Processor processing weight capped

Get/Set: Whether or not the initial processing weight for Application Assist Processor (AAP) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE (1)
  Indicates that the initial Application Assist Processor (AAP) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Application Assist Processor (AAP) processor resources, regardless of the availability of excess Application Assist Processor (AAP) processor resources.

  HWMCA_FALSE (0)
  Indicates that the initial Application Assist Processor (AAP) processor processing weight for the CPC Image is not capped. It represents the share of Application Assist Processor (AAP) processor resources guaranteed to a logical partition when all Application Assist Processor (AAP) processor resources are in use. Otherwise, when excess Application Assist Processor (AAP) processor resources are available, the logical partition can use them if necessary.

  Note: The initial Application Assist Processor (AAP) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  This attribute and the **Workload manager enabled** attribute are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this attribute it may be necessary to first disable the **Workload manager enabled** attribute. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and the **Workload manager enabled** attribute being disabled automatically in a single operation.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.61.0.*

Minimum Application Assist Processor processing weight

Get/Set: The minimum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1-999 Represents the minimum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.
- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Application Assist Processor (AAP) processor resources allocated to the CPC Image object. A value of zero can also be specified to indicate that there is no minimum value for the processing weight.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.62.0.*

Maximum Application Assist Processor processing weight

Get/Set: The maximum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1-999 Represents the maximum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Application Assist Processor (AAP) processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.63.0.*

Current Application Assist Processor processing weight

Get: The current relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1-999 Represents the current relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.64.0.*

Current Application Assist Processor processing weight capped

Get: Whether or not the current Application Assist Processor (AAP) processing weight is a limit or a target.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the current Application Assist Processor (AAP) processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the current Application Assist Processor (AAP) processing weight for the CPC Image is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

  Note: The current Application Assist Processor (AAP) processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.65.0.*

Initial Integrated Facility for Linux processing weight

Get/Set: The relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

1-999 Represents the relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.66.0.*

Initial Integrated Facility for Linux processing weight capped

Get/Set: Whether or not the initial processing weight for Integrated Facility for Linux (IFL) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE (1)
  Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Integrated Facility for Linux (IFL) processor resources, regardless of the availability of excess Integrated Facility for Linux (IFL) processor resources.

  HWMCA_FALSE (0)
  Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the CPC Image is not capped. It represents the share of Integrated Facility for Linux (IFL) processor resources guaranteed to a logical partition when all Integrated Facility for Linux (IFL) processor resources are in use. Otherwise, when excess Integrated Facility for Linux (IFL) processor resources are available, the logical partition can use them if necessary.

  Note: The initial Integrated Facility for Linux (IFL) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

  This attribute and the **Workload manager enabled** attribute are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this attribute it might be necessary to first disable the **Workload manager enabled** attribute. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and the **Workload manager enabled** attribute being disabled automatically in a single operation.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.67.0.*

Minimum Integrated Facility for Linux processing weight

Get/Set: The minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

  1-999 Represents the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.
- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the CPC Image object. A value of zero can also be specified to indicate that there is no minimum value for the processing weight.

Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.68.0.*

Maximum Integrated Facility for Linux processing weight

Get/Set: The maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object. The maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

  1-999 Represents the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.69.0.*

Current Integrated Facility for Linux processing weight

Get: The current relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

  1-999 Represents the current relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.70.0.*

Current Integrated Facility for Linux Processing weight capped

Get: Whether or not the current Integrated Facility for Linux (IFL) processing weight is a limit or a target.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the current Integrated Facility for Linux (IFL) processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the current Integrated Facility for Linux (IFL) processing weight for the CPC Image is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

  Note: The current Integrated Facility for Linux (IFL) processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.71.0.*

Initial Integrated Information Processors processing weight

Get/Set: The relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.81.0.*

Initial Integrated Information Processors processing weight capped

Get/Set: Whether or not the initial processing weight for Integrated Information Processors (zIIP) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE (1)
  Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Integrated Information Processors (zIIP) processor resources, regardless of the availability of excess Integrated Information Processors (zIIP) processor resources.

  HWMCA_FALSE (0)
  Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the CPC Image is not capped. It represents the share of Integrated Information Processors (zIIP) processor resources guaranteed to a logical partition when all Integrated Information Processors (zIIP) processor resources are in use. Otherwise, when excess Integrated Information Processors (zIIP) processor resources are available, the logical partition can use them if necessary.

  Note: The initial Integrated Information Processors (zIIP) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  This attribute and the **Workload manager enabled** attribute are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this attribute it may be necessary to first disable the **Workload manager enabled** attribute. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and the **Workload manager enabled** attribute being disabled automatically in a single operation.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.82.0.*

Minimum Integrated Information Processors processing weight

Get/Set: The minimum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the CPC Image object. A value of zero can also be specified to indicate that there is no minimum value for the processing weight.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.83.0.*

Maximum Integrated Information Processors Processing Weight

Get/Set: The maximum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.
- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.84.0.*

Current Integrated Information Processors processing weight

Get: The current relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the current relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.85.0.*

Current Integrated Information Processors processing weight capped

Get: Whether or not the current Integrated Information Processors (zIIP) processing weight is a limit or a target.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the current Integrated Information Processors (zIIP) processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the current Integrated Information Processors (zIIP) processing weight for the CPC Image is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

  Note: The current Integrated Information Processors (zIIP) processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the CPC Image does

not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.86.0.*

Group profile name

Get/Set: Defines the name of the group capacity profile that is being used for the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.93.0.*.*

Program Status Word (PSW) information

Get: An XML string that describes the current PSW information for each logical processor associated with the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING

  Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.150.0

IPL Token

Get: The Token used in the last IPL.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.164.0

Group Profile capacity

Get/Set: The current capacity value of the Group Profile the CPC Image object is associated with.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.192.0

Last Used Load Address

Get: The load addressed used in the last IPL.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.201.0

Last Used Load Parameter

Get: The load parameter used in the last IPL.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.202.0

Absolute capping type

Get/Set: The type of absolute capping to perform.

- Data type returned on Get/Set: HWMCA_TYPE_INTEGER

  0 None

  1 Absolute capping in number of processors
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.217.0.*

Absolute capping value

Get/Set: The value used for absolute capping (if enabled).

- Data type returned on Get/Set: HWMCA_TYPE_OCTETSTRING

  0 None

  1-nnnn
  Represents the number of processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.218.0.*

Application Assist Processor absolute capping type

Get/Set: The type of absolute capping to perform for Application Assist Processor (AAP) processors.

- Data type returned on Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Application Assist Processor (AAP) processors

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.219.0.*

Application Assist Processor absolute capping value

Get/Set: The value used for Application Assist Processor (AAP) absolute capping.

- Data type returned on Get/Set: HWMCA_TYPE_OCTETSTRING

0 None

1-nnnn
Represents the number of Application Assist Processor (AAP) processors when capping in number of processors is enabled.

Note: Though this is an integer, value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.220.0.*

Integrated Facility for Linux absolute capping type

Get/Set: The type of absolute capping to perform for Integrated Facility for Linux (IFL) processors.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Integrated Facility for Linux (IFL) processors.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.221.0.*

Integrated Facility for Linux absolute capping value

Get/Set: The value used for Integrated Facility for Linux (IFL) absolute capping (if enabled).

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

0 None

1-nnnn
Represents the number of Integrated Facility for Linux (IFL) processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.222.0.*

Integrated Information Processor absolute capping type

Get/Set: The type of absolute capping to perform for Integrated Information Processor (zIIP) processors.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Integrated Information Processor (zIIP) processors.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.225.0.*

Integrated Information Processor absolute capping value

Get/Set: The value used for Integrated Information Processor (zIIP) absolute capping.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

0 Absolute capping not enabled.

1-nnnn
Represents the number of Integrated Information Processor (zIIP) processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.226.0.*

CPC image relationships

CPC (H/W image)

A CPC image is a member of a CPC (H/W image); there can be from ***1*** to ***n*** CPC Images. ***N*** is determined by the Licensed Internal Code.

Software image

A CPC image has one software image running in it.

Note: Some operating systems support running guest operating systems within themselves.

CPC image commands

Activate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.1 (HWMCA_ACTIVATE_COMMAND)

Reset normal

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.4 (HWMCA_RESETNORMAL_COMMAND)

Deactivate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.2 (HWMCA_DEACTIVATE_COMMAND)

Start

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.5 (HWMCA_START_COMMAND)

Stop

SNMP MIB Name - 1.3.6.1.4.1.2.6.42.4.6 (HWMCA_STOP_COMMAND)

PSW restart

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.7 (HWMCA_PSWRESTART_COMMAND)

Send operating system command

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.3 (HWMCA_SEND_OPSYS_COMMAND)

Load

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.10 (HWMCA_LOAD_COMMAND)

Reset clear

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.12 (HWMCA_RESETCLEAR_COMMAND)

External interrupt
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.19 (HWMCA_EXTERNAL_INTERRUPT_COMMAND)

SCSI load
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.20 (HWMCA_SCSI_LOAD_COMMAND)

SCSI dump
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.21 (HWMCA_SCSI_DUMP_COMMAND)

CPC image notifications

Message (operating system - HWMCA_EVENT_MESSAGE)
An HWMCA_TYPE_INTEGER that specifies whether the message is a hardware or operating system message (HWMCA_HARDWARE_MESSAGE or HWMCA_OPSYS_MESSAGE).

For operating system messages:
- An HWMCA_TYPE_OCTETSTRING that specifies the new or refreshed operating system message text.
- An HWMCA_TYPE_OCTETSTRING that specifies the message identifier of the new operating system message.
- An HWMCA_TYPE_OCTETSTRING that specifies the date of the new operating system message.
- An HWMCA_TYPE_OCTETSTRING that specifies the time of the new operating system message.
- An HWMCA_TYPE_INTEGER that specifies whether the new operating system message should cause the alarm to be sounded (HWMCA_TRUE or HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies whether the new operating system message is a priority message or not (HWMCA_TRUE or HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies whether the new operating system message is a held message or not (HWMCA_TRUE or HWMCA_FALSE).
- An HWMCA_TYPE_OCTETSTRING that specifies the prompt text that should be associated with the new operating system message or an HWMCA_TYPE_NULL indicating that there is no prompt text for this new operating system message.
- An HWMCA_TYPE_OCTETSTRING that specifies the name of the operating system that generated this new operating system message or an HWMCA_TYPE_NULL indicating that there is no operating system name associated with this new operating system message.
- An HWMCA_TYPE_INTEGER that specifies whether the message is a new (HWMCA_FALSE) or refresh message (HWMCA_TRUE).

Status change (HWMCA_EVENT_STATUS_CHANGE)
- An HWMCA_TYPE_INTEGER that specifies the new status value
- An HWMCA_TYPE_INTEGER that specifies the old status value.

Object name change (HWMCA_EVENT_NAME_CHANGE)
- An HWMCA_TYPE_OCTETSTRING that specifies the new object name
- An HWMCA_TYPE_OCTETSTRING that specifies the old object name.

Object activation profile change (HWMCA_EVENT_ACTIVATE_PROF_CHANGE)
- An HWMCA_TYPE_OCTETSTRING that specifies the new activation profile name
- An HWMCA_TYPE_OCTETSTRING that specifies the old activation profile name.

Object created (HWMCA_EVENT_CREATED)
This event has no additional data. The object identifier can be used with the *HwmcaGet* to get any data required for this newly created object.

Object destruction (HWMCA_EVENT_DESTROYED)
This event has no additional data.

Object entered an exception state (HWMCA_EVENT_EXCEPTION_STATE)

- An HWMCA_TYPE_INTEGER that specifies whether the object is entering into an exception state (HWMCA_TRUE) or leaving an exception state (HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies the status value for the object.

Disabled wait (HWMCA_EVENT_DISABLED_WAIT)

- An HWMCA_TYPE_OCTETSTRING for the name of the Defined CPC that is associated with the CPC Image that entered a disabled wait state.
- An HWMCA_TYPE_OCTETSTRING for the disabled wait PSW value.
- An HWMCA_TYPE_INTEGER for the partition identifier of the CPC Image that entered a disabled wait state.
- An HWMCA_TYPE_INTEGER for number of the processor that entered a disabled wait state.
- An HWMCA_TYPE_OCTETSTRING for the serial number of the Defined CPC that is associated with the CPC Image that entered a disabled wait state.
- An HWMCA_TYPE_OCTETSTRING that specifies the name of the object that the event pertains to (in this case a CPC Image object).

Coupling facility

Coupling facility name bindings

Coupling facility object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.x.x.*

Where ***x.x.*** equals the attribute identifier for the object and an * equals a unique number for that specific instance of the Coupling Facility Object.

Coupling facility attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.1.0.*

Parent name

Get (CPC's logical name):

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.2.0.*

Status error

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object is in a state which is not an acceptable status.

  HWMCA_FALSE
  Object is in an acceptable status state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.7.0.*

Busy

Get:

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object in a busy state (currently performing a task).

HWMCA_FALSE
Object not in a busy state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.8.0.*

Message indicator

Get:
- Data type returned on Get: HWMCA_TYPE_INTEGER

HWMCA_TRUE
Object has an operating system message.

HWMCA_FALSE
Object does not have an operating system message.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.9.0.*

Status

Get:
- Data type returned on Get: HWMCA_TYPE_INTEGER

  One of the following bit values will be set to on:
  - HWMCA_STATUS_OPERATING
  - HWMCA_STATUS_NOT_OPERATING
  - HWMCA_STATUS_NOT_ACTIVATED
  - HWMCA_STATUS_EXCEPTIONS
  - HWMCA_STATUS_STATUS_CHECK
  - HWMCA_STATUS_POWERSAVE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.10.0.*

Acceptable status

Get/Set:
- Data type returned on Get: HWMCA_TYPE_INTEGER
- Data type for Set: HWMCA_TYPE_INTEGER
- One or more of the following bit values will be set to on:
  - HWMCA_STATUS_OPERATING
  - HWMCA_STATUS_NOT_OPERATING
  - HWMCA_STATUS_NOT_ACTIVATED
  - HWMCA_STATUS_EXCEPTIONS
  - HWMCA_STATUS_STATUS_CHECK
  - HWMCA_STATUS_POWERSAVE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.11.0.*

Activation profile name

Get (always the Image profile):
- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.13.0.*

Last used activation profile

Get:
- Data type returned on Get: HWMCA_TYPE_OCTETSTRING or HWMCA_TYPE_NULL
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.14.0.*

Object type

Get: This returns the type of object the object identifier represents.
- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_CF_OBJECT
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.22.0.*

Initial processing weight

Get/Set: The relative amount of shared general purpose processor resources initially allocated to the Coupling Facility object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 Coupling Facility does not represent a logical partition or the Coupling Facility does not represent a logical partition with at least one not dedicated general purposee processor.

  1 - 999 Represents the relative amount of shared general purpose processor resources initially allocated to the Coupling Facility object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared general purpose processor resources allocated to the Coupling Facility object.

  Note: The setting of this attribute is only valid for Coupling Facility objects that represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.30.0.*

Initial processing weight capped

Get/Set: Whether or not the initial processing weight for general purpose processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE (1)
  Indicates that the initial general purpose processor processing weight for the Coupling Facility object is capped. It represents the logical partition's maximum share of general purpose processor resources, regardless of the availability of excess general purpose processor resources.

  HWMCA_FALSE (0)
  Indicates that the initial general purpose processor processing weight for the Coupling Facility is not capped. It represents the share of general purpose processor resources guaranteed to a logical partition when all general purpose processor resources are in use. Otherwise, when excess general purpose processor resources are available, the logical partition can use them if necessary.

  Note: The initial general purpose processor processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the Coupling Facility does not represent a logical partition or the Coupling Facility does not represent a logical partition with at least one not dedicated general purpose processor.
  This attribute and the **Workload manager enabled** attribute are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this attribute it might be necessary to first disable the **Workload manager enabled** attribute. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and the **Workload manager enabled** attribute being disabled automatically in a single operation.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.31.0.*

Minimum processing weight

Get/Set: The minimum relative amount of shared general purpose processor resources allocated to the Coupling Facility object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 Coupling Facility does not represent a logical partition or the Coupling Facility does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the minimum relative amount of shared general purpose processor resources allocated to the Coupling Facility object.

- Data type for Set: HWMCA_TYPE_INTEGER

A value 1 - 999 and less than or equal to the Initial Processing Weight used to define the minimum relative amount of shared general purpose processor resources allocated to the Coupling Facility object.

Note: The setting of this attribute is only valid for Coupling Facility objects that represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.38.0.*

Maximum processing weight

Get/Set: The maximum relative amount of shared general purpose processor resources allocated to the Coupling Facility object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 Coupling Facility does not represent a logical partition or the Coupling Facility does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the maximum relative amount of shared general purpose processor resources allocated to the Coupling Facility object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 and greater than or equal to the Initial Processing Weight used to define the maximum relative amount of shared general purpose processor resources allocated to the Coupling Facility object. A value of zero can also be specified to indicate that there is no maximum value for the processing weight.

  Note: The setting of this attribute is only valid for Coupling Facility objects that represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.39.0.*

Current processing weight

Get: The relative amount of shared general purpose processor resources currently allocated to the Coupling Facility object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 Coupling Facility does not represent a logical partition or the Coupling Facility does not represent a logical partition with at least one not dedicated general purpose processor.

  1 - 999 Represents the relative amount of shared general purpose processor resources currently allocated to the Coupling Facility object.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.41.0.*

Current processing weight capped

Get: Whether or not the current general purpose processing weight is a limit or a target.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the current general purpose processing weight for the Coupling Facility object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the general purpose current processing weight for the Coupling Facility is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

Note: The current general purpose processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the Coupling Facility does not represent a logical partition or the Coupling Facility does not represent a logical partition with at least one not dedicated general purpose processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.42.0.*

WorkLoad manager enabled

Get/Set: Whether or not WorkLoad Manager is allowed to change processing weight related attributes.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that WorkLoad Manager is allowed to change processing weight related attributes for this CPC Image object.

  HWMCA_FALSE
  Indicates that WorkLoad Manager is not allowed to change processing weight related attributes for this CPC Image object.

  This attribute and the various capping attributes are mutually exclusive and cannot be enabled at the same time. Therefore in order to enable this attribute it might be necessary to first disable any capping attribute that is currently enabled. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and all capping attributes being disabled automatically in a single operation.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.40.0.*

Defined capacity

Get/Set: The defined capacity expressed in terms of Millions of Service Units (MSU)s per hour. MSUs is a measure of processor resource consumption. The amount of MSUs a logical partition consumes is dependent on the model, the number of logical processors available to the partition, and the amount of time the logical partition is dispatched. The defined capacity value specifies how much capacity the logical partition is to be managed to by WorkLoad Manager for the purpose of software pricing.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  0 No defined capacity is specified for this logical partition.

  1 - nnnn
  Represents the amount of defined capacity specified for this logical partition.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.43.0.*

Partition identifier

Get: The partition identifier for the Coupling Facility object.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.51.0*

Initial Internal Coupling Facility processing weight

Get/Set: The relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  1-999 Represents the relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the CPC Image object.

Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.72.0.*

Initial Internal Coupling Facility processing weight capped

Get/Set: Whether or not the initial processing weight for Internal Coupling Facility (ICF) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE (1)
  Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Internal Coupling Facility (ICF) processor resources, regardless of the availability of excess Internal Coupling Facility (ICF) processor resources.

  HWMCA_FALSE (0)
  Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the CPC Image is not capped. It represents the share of Internal Coupling Facility (ICF) processor resources guaranteed to a logical partition when all Internal Coupling Facility (ICF) processor resources are in use. Otherwise, when excess Internal Coupling Facility (ICF) processor resources are available, the logical partition can use them if necessary.

  Note: The initial Internal Coupling Facility (ICF) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  This attribute and the **Workload manager enabled** attribute are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this attribute it might be necessary to first disable the **Workload manager enabled** attribute. It is also possible to use a value of -1 when setting this attribute, which will result in this attribute being enabled and the **Workload manager enabled** attribute being disabled automatically in a single operation.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.73.0.*

Minimum Internal Coupling Facility processing weight

Get/Set: The minimum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  1-999 Represents the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.74.0.*

Maximum Internal Coupling Facility processing weight

Get/Set: The maximum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

1-999 Represents the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.75.0.*

Current Internal Coupling Facility processing weight

Get: The current relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  1-999 Represents the current relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.76.0.*

Current Internal Coupling Facility processing weight capped

Get: Whether or not the current Internal Coupling Facility (ICF) processing weight is a limit or a target.

- Data type for Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the current Internal Coupling Facility (ICF) processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the current Internal Coupling Facility (ICF) processing weight for the CPC Image is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

  Note: The current Internal Coupling Facility (ICF) processing weight capped attribute cannot be set and the value returned for a get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.77.0.*

Program Status Word (PSW) information

Get: An XML string that describes the current PSW information for each logical processor associated with the Coupling Facility object.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING

  Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.150.0

Internal Coupling Facility absolute capping type

Get/Set: The type of absolute capping to perform for Internal Coupling Facility (ICF) processors.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  0 None

  1 Absolute capping in number of Internal Coupling Facility (ICF) processors.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.223.0.*

Internal Coupling Facility absolute capping value

Get/Set: The value used for Internal Coupling Facility (ICF) absolute capping.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

  0 Absolute capping not enabled.

  1-nnnn
  Represents the number of Internal Coupling Facility (ICF) processors when capping in number of processors is enabled.

  Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.2.0.224.0.*

Coupling facility relationships

CPC (H/W image)

A coupling facility image is a member of a CPC (H/W image) there can be from ***1*** to ***n*** coupling facility images. ***N*** is determined by the Licensed Internal Code.

Coupling Facility Control Code (CFCC)

A coupling facility image is running the Coupling Facility Control Code to perform the Coupling Facility functions.

Coupling facility commands

Activate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.1 (HWMCA_ACTIVATE_COMMAND)

Deactivate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.2 (HWMCA_DEACTIVATE_COMMAND)

Send operating system command

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.3 (HWMCA_SEND_OPSYS_COMMAND)

Coupling facility notifications

Message (operating system - HWMCA_EVENT_MESSAGE)

An HWMCA_TYPE_INTEGER that specifies whether the message is a hardware or operating system message (HWMCA_HARDWARE_MESSAGE or HWMCA_OPSYS_MESSAGE).

For operating system messages:

- An HWMCA_TYPE_OCTETSTRING that specifies the new or refreshed operating system message text.
- An HWMCA_TYPE_OCTETSTRING that specifies the message identifier of the new operating system message.
- An HWMCA_TYPE_OCTETSTRING that specifies the date of the new operating system message.

- An HWMCA_TYPE_OCTETSTRING that specifies the time of the new operating system message.
- An HWMCA_TYPE_INTEGER that specifies whether the new operating system message should cause the alarm to be sounded (HWMCA_TRUE or HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies whether the new operating system message is a priority message or not (HWMCA_TRUE or HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies whether the new operating system message is a held message or not (HWMCA_TRUE or HWMCA_FALSE).
- An HWMCA_TYPE_OCTETSTRING that specifies the prompt text that should be associated with the new operating system message or an HWMCA_TYPE_NULL indicating that there is no prompt text for this new operating system message.
- An HWMCA_TYPE_OCTETSTRING that specifies the name of the operating system that generated this new operating system message or an HWMCA_TYPE_NULL indicating that there is no operating system name associated with this new operating system message.
- An HWMCA_TYPE_INTEGER that specifies whether the message is a new (HWMCA_FALSE) or refresh message (HWMCA_TRUE).

Status change (HWMCA_EVENT_STATUS_CHANGE)
- An HWMCA_TYPE_INTEGER that specifies the new status value
- An HWMCA_TYPE_INTEGER that specifies the old status value.

Object name change (HWMCA_EVENT_NAME_CHANGE)
- An HWMCA_TYPE_OCTETSTRING that specifies the new object name
- An HWMCA_TYPE_OCTETSTRING that specifies the old object name.

Object activation profile change (HWMCA_EVENT_ACTIVATE_PROF_CHANGE)
- An HWMCA_TYPE_OCTETSTRING that specifies the new activation profile name
- An HWMCA_TYPE_OCTETSTRING that specifies the old activation profile name.

Object created (HWMCA_EVENT_CREATED)
This event has no additional data. The object identifier can be used with the *HwmcaGet* to get any data required for this newly created object.

Object destruction (HWMCA_EVENT_DESTROYED)
This event has no additional data.

Object entered an exception state (HWMCA_EVENT_EXCEPTION_STATE)
- An HWMCA_TYPE_INTEGER that specifies whether the object is entering into an exception state (HWMCA_TRUE) or leaving an exception state (HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies the status value for the object.

Reset activation profile object

Reset activation profile name bindings

Reset activation profile object identifier
SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.x.x.y.z

Where ***x.x.*** equals the attribute identifier for the object, ***y*** equals a unique number for the specific instance of the CPC Object, and ***z*** equals a unique number for the specific instance of the Reset Activation Profile.

Reset activation profile attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.1.0.*.*

Object type

Get: This returns the type of object the object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER
  HWMCA_ACT_PROFILE_RESET
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.22.0.*.*

IOCDS

Get/Set:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A value of an empty string is used to indicate that the Reset Activation Profile will use the currently active IOCDS.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.27.0.*.*

Processor running time type

Get/Set: Defines whether the processor running time is dynamically determined by the system or set to a constant value for the Defined CPC object.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  0 (HWMCA_DETERMINED_SYSTEM)
  The processor running is dynamically determined by the system.

  1 (HWMCA_DETERMINED_USER)
  The processor running time is set to a constant value.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.78.0.*.*

Processor running time

Get/Set: Defines the amount of continuous time allowed for logical processors to perform jobs on shared processors for the Defined CPC object.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  A value between 1 and 100 for the user defined processor running time.

  Note: This value can only be set if the processor running time type is set to 1 (HWMCA_DETERMINED_USER). Additionally, this value will always be returned as zero if the processor running time type is set to 0 (HWMCA_DETERMINED_SYSTEM).
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.79.0.*.*

End timeslice if CPC image enters a wait state

Get/Set: Defines whether CPC Images lose their share of running time when they enter a wait state.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that a CPC Image should lose its share of running time when it enters a wait state.

  HWMCA_FALSE
  Indicates that a CPC Image should not lose its share of running time when it enters a wait state.

Note: This value can only be set if the processor running time type is set to 1 (HWMCA_DETERMINED_USER). Additionally, this value will always be returned as zero if the processor running time type is set to 0 (HWMCA_DETERMINED_SYSTEM).

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.80.0.*.*

Description

Get/Set: The description of the profile with a maximum length of 51 (including the null terminator).

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.5.0.203.0.*.*

Image activation profile object

Image activation profile name bindings

Image activation profile object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.x.x.y.z

Where ***x.x.*** equals the attribute identifier for the object, ***y*** equals a unique number for the specific instance of the CPC Object, and ***z*** equals a unique number for the specific instance of the Image Activation Profile.

Image activation profile attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.1.0.*.*

Object type

Get: This returns the type of object the object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER
  HWMCA_ACT_PROFILE_IMAGE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.22.0.*.*

IPL address

Get/Set:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A value of an empty string is used to indicate that the Image Activation Profile will use next IPL address set by HCD.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.28.0.*.*

IPL parameter

Get/Set:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A value of an empty string is used to indicate that the Image Activation Profile will use next IPL parameter set by HCD.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.29.0.*.*

Initial processing weight

Get/Set: The relative amount of shared processor resources initially allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition with at least one not dedicated central processor.

  1 - 999 Represents the relative amount of shared processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated central processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.30.0.*.*

Initial processing weight capped

Get/Set: Whether or not the initial processing weight is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the initial processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources.

  HWMCA_FALSE
  Indicates that the initial processing weight for the CPC Image is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.

  Note: The initial processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition with at least one not dedicated central processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.31.0.*.*

Minimum processing weight

Get/Set: The minimum relative amount of shared processor resources allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition with at least one not dedicated central processor.

  1 - 999 Represents the minimum relative amount of shared processor resources allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 and less than or equal to the Initial Processing Weight used to define the minimum relative amount of shared processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated central processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.38.0.*.*

Maximum processing weight

Get/Set: The maximum relative amount of shared processor resources allocated to the CPC Image object.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition with at least one not dedicated central processor.

  1 - 999 Represents the maximum relative amount of shared processor resources allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 and greater than or equal to the Initial Processing Weight used to define the maximum relative amount of shared processor resources allocated to the CPC Image object.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated central processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.39.0.*.*

WorkLoad manager enabled

Get/Set: Whether or not WorkLoad Manager is allowed to change processing weight related attributes.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that WorkLoad Manager is allowed to change processing weight related attributes for this CPC Image object.

  HWMCA_FALSE
  Indicates that WorkLoad Manager is not allowed to change processing weight related attributes for this CPC Image object.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.40.0.*.*

Defined capacity

Get/Set: The defined capacity expressed in terms of Millions of Service Units (MSU)s per hour. MSUs is a measure of processor resource consumption. The amount of MSUs a logical partition consumes is dependent on the model, the number of logical processors available to the partition, and the amount of time the logical partition is dispatched. The defined capacity value specifies how much capacity the logical partition is to be managed to by WorkLoad Manager for the purpose of software pricing.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  0 No defined capacity is specified for this logical partition.

  1 - nnnn
  Represents the amount of defined capacity specified for this logical partition.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.43.0.*.*

IPL type

Get/Set: The IPL type value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_IPLTYPE_STANDARD
  Indicates that the image activation profile is used to perform a standard load.

  HWMCA_IPLTYPE_SCSI
  Indicates that the image activation profile is used to perform a SCSI load.

  HWMCA_IPLTYPE_SCSIDUMP
  Indicates that the image activation profile is used to perform a SCSI dump.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.52.0.*.*

Worldwide port name

Get/Set: The worldwide port name value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.53.0.*.*

Boot program selector

Get/Set: The boot program selector value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.54.0.*.*

Logical unit number

Get/Set: The logical unit number value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.55.0.*.*

Boot record logical block address

Get/Set: The boot record logical block address value for the activation profile.

- Data type for get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.56.0.*.*

Operating system specific load parameters

Get/Set: The operating system specific load parameters for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.57.0.*.*

Initial Application Assist Processor processing weight

Get/Set: The relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1-999 Represents the relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Application Assist Processor (AAP) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.60.0.*.*

Initial Application Assist Processor processing weight capped

Get/Set: Whether or not the initial processing weight for Application Assist Processor (AAP) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the initial Application Assist Processor (AAP) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Application Assist Processor (AAP) processor resources, regardless of the availability of excess Application Assist Processor (AAP) processor resources.

  HWMCA_FALSE
  Indicates that the initial Application Assist Processor (AAP) processor processing weight for the

CPC Image is not capped. It represents the share of Application Assist Processor (AAP) processor resources guaranteed to a logical partition when all Application Assist Processor (AAP) processor resources are in use. Otherwise, when excess Application Assist Processor (AAP) processor resources are available, the logical partition can use them if necessary.

Note: The initial Application Assist Processor (AAP) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.61.0.*.*

Minimum Application Assist Processor processing weight

Get/Set: The minimum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1-999 Represents the minimum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Application Assist Processor (AAP) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.62.0.*.*

Maximum Application Assist Processor processing weight

Get/Set: The maximum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

  1-999 Represents the maximum relative amount of shared Application Assist Processor (AAP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Application Assist Processor (AAP) processor resources allocated to the CPC Image object. A value of zero can also be specified to indicate that there is no maximum value for the processing weight.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.63.0.*.*

Initial Integrated Facility for Linux processing weight

Get/Set: The relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

  1-999 Represents the relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.66.0.*.*

Initial Integrated Facility for Linux Processing weight capped

Get/Set: Whether or not the initial processing weight for Integrated Facility for Linux (IFL) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Integrated Facility for Linux (IFL) processor resources, regardless of the availability of excess Integrated Facility for Linux (IFL) processor resources.

  HWMCA_FALSE
  Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the CPC Image is not capped. It represents the share of Integrated Facility for Linux (IFL) processor resources guaranteed to a logical partition when all Integrated Facility for Linux (IFL) processor resources are in use. Otherwise, when excess Integrated Facility for Linux (IFL) processor resources are available, the logical partition can use them if necessary.

  Note: The initial Integrated Facility for Linux (IFL) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.67.0.*.*

Minimum Integrated Facility for Linux processing weight

Get/Set: The minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

  1-999 Represents the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.
- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.68.0.*.*

Maximum Integrated Facility for Linux processing weight

Get/Set: The maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.

1-999 Represents the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the CPC Image object. The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Application Assist Processor (AAP) processor. A value of zero can also be specified to indicate that there is no maximum value for the processing weight.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.69.0.*.*

Initial Internal Coupling Facility processing weight

Get/Set: The relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  1-999 Represents the relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.
- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.72.0.*.*

Initial Internal Coupling Facility processing weight capped

Get/Set: Whether or not the initial processing weight for Internal Coupling Facility (ICF) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Internal Coupling Facility (ICF) processor resources, regardless of the availability of excess Internal Coupling Facility (ICF) processor resources.

  HWMCA_FALSE
  Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the CPC Image is not capped. It represents the share of Internal Coupling Facility (ICF) processor resources guaranteed to a logical partition when all Internal Coupling Facility (ICF) processor resources are in use. Otherwise, when excess Internal Coupling Facility (ICF) processor resources are available, the logical partition can use them if necessary.

  Note: The initial Internal Coupling Facility (ICF) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.73.0.*.*

Minimum Internal Coupling Facility processing weight

Get/Set: The minimum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  1-999 Represents the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.74.0.*.*

Maximum Internal Coupling Facility processing weight

Get/Set: The maximum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

  1-999 Represents the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.75.0.*.*

Initial Integrated Information Processors processing weight

Get/Set: The relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.81.0.*.*

Initial Integrated Information Processors processing weight capped

Get/Set: Whether or not the initial processing weight for Integrated Information Processors (zIIP) processors is a limit or a target.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

HWMCA_TRUE
Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the CPC Image object is capped. It represents the logical partition's maximum share of Integrated Information Processors (zIIP) processor resources, regardless of the availability of excess Integrated Information Processors (zIIP) processor resources.

HWMCA_FALSE
Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the CPC Image is not capped. It represents the share of Integrated Information Processors (zIIP) processor resources guaranteed to a logical partition when all Integrated Information Processors (zIIP) processor resources are in use. Otherwise, when excess Integrated Information Processors (zIIP) processor resources are available, the logical partition can use them if necessary.

Note: The initial Integrated Information Processors (zIIP) processor processing weight capped attribute cannot be set and the value returned for a Get request is always HWMCA_FALSE when the CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.82.0.*.*

Minimum Integrated Information Processors processing weight

Get/Set: The minimum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the CPC Image object. Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.83.0.*.*

Maximum Integrated Information Processors processing weight

Get/Set: The maximum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  0 CPC Image does not represent a logical partition or the CPC Image does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

  1-999 Represents the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the CPC Image object.

- Data type for Set: HWMCA_TYPE_INTEGER

  A value 1 - 999 used to define the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the CPC Image object. A value of zero can also be specified to indicate that there is no maximum value for the processing weight.

  Note: The setting of this attribute is only valid for CPC Image objects that represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.84.0.*.*

Group profile name

Get/Set: Defines the name of the group capacity profile that is to be used for the CPC Image object activated with this profile.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.93.0.*.*

Load at activation

Get/Set: Defines if the CPC Image object activated with this profile should be loaded (IPLed) at the end of the activation.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The CPC Image object will be loaded (IPLed) at the end of the activation.

  HWMCA_FALSE
  The CPC Image object will not be loaded (IPLed) at the end of the activation.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.94.0.*.*

Central storage

Get/Set: Defines the initial amount of central storage (in megabytes) to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.95.0.*.*

Reserved central storage

Get/Set: Defines the reserved amount of central storage (in megabytes) to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.96.0.*.*

Expanded storage

Get/Set: Defines the initial amount of expanded storage (in megabytes) to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.97.0.*.*

Reserved expanded storage

Get/Set: Defines the reserved amount of expanded storage (in megabytes) to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.98.0.*.*

Number of dedicated general purpose processors

Get/Set: Defines the number of dedicated general purpose processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.99.0.*.*

Number of reserved dedicated general purpose processors

Get/Set: Defines the number of reserved dedicated general purpose processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.100.0.*.*

Number of dedicated Application Assist Processor (AAP) processors

Get/Set: Defines the number of dedicated Application Assist Processor (AAP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.101.0.*.*

Number of reserved dedicated Application Assist Processor (AAP) Processors

Get/Set: Defines the number of reserved dedicated Application Assist Processor (AAP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.102.0.*.*

Number of dedicated Integrated Facility for Linux (IFL) processors

Get/Set: Defines the number of dedicated Integrated Facility for Linux (IFL) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.103.0.*.*

Number of reserved dedicated Integrated Facility for Linux (IFL) processors

Get/Set: Defines the number of reserved dedicated Integrated Facility for Linux (IFL) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.104.0.*.*

Number of dedicated Internal Coupling Facility (ICF) processors

Get/Set: Defines the number of dedicated Internal Coupling Facility (ICF) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.105.0.*.*

Number of reserved dedicated Internal Coupling Facility (ICF) processors

Get/Set: Defines the number of reserved dedicated Internal Coupling Facility (ICF) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.106.0.*.*

Number of dedicated Integrated Information Processors (zIIP) processors

Get/Set: Defines the number of dedicated Integrated Information Processors (zIIP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.107.0.*.*

Number of reserved dedicated Integrated Information Processors (zIIP) processors

Get/Set: Defines the number of reserved dedicated Integrated Information Processors (zIIP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.108.0.*.*

Number of shared general purpose processors

Get/Set: Defines the number of shared general purpose processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.109.0.*.*

Number of reserved shared general purpose processors

Get/Set: Defines the number of reserved shared general purpose processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.110.0.*.*

Number of shared Application Assist Processor (AAP) processors

Get/Set: Defines the number of shared Application Assist Processor (AAP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.111.0.*.*

Number of reserved shared Application Assist Processor (AAP) processors

Get/Set: Defines the number of reserved shared Application Assist Processor (AAP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.112.0.*.*

Number of shared Integrated Facility for Linux (IFL) processors

Get/Set: Defines the number of shared Integrated Facility for Linux (IFL) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.113.0.*.*

Number of reserved shared Integrated Facility for Linux (IFL) processors

Get/Set: Defines the number of reserved shared Integrated Facility for Linux (IFL) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.114.0.*.*

Number of shared Internal Coupling Facility (ICF) processors

Get/Set: Defines the number of shared Internal Coupling Facility (ICF) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.115.0.*.*

Number of reserved shared Internal Coupling Facility (ICF) processors

Get/Set: Defines the number of reserved shared Internal Coupling Facility (ICF) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.116.0.*.*

Number of shared Integrated Information Processors (zIIP) processors

Get/Set: Defines the number of shared Integrated Information Processors (zIIP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.117.0.*.*

Number of reserved shared Integrated Information Processors (zIIP) processors

Get/Set: Defines the number of reserved shared Integrated Information Processors (zIIP) processors to be used for the CPC Image object activated with this profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.118.0.*.*

Basic CPU counter authorization control

Get/Set: Enables/disables the use of the basic CPU counter facility for the CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The authorization control is enabled.

  HWMCA_FALSE
  The authorization control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.168.0.*.*

Problem state CPU counter authorization control

Get/Set: Enables/disables the use of the problem state CPU counter facility for the CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The authorization control is enabled.

  HWMCA_FALSE
  The authorization control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.169.0.*.*

Crypto activity CPU counter authorization control

Get/Set: Enables/disables the use of the crypto activity CPU counter facility for the CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The authorization control is enabled.

  HWMCA_FALSE
  The authorization control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.170.0.*.*

Extended CPU counter authorization control

Get/Set: Enables/disables the use of the extended CPU counter facility for the CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The authorization control is enabled.

  HWMCA_FALSE
  The authorization control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.171.0.*.*

Coprocessor group CPU counter authorization control

Get/Set: Enables/disables the use of the coprocessor group CPU counter facility for the CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The authorization control is enabled.

  HWMCA_FALSE
  The authorization control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.172.0.*.*

Basic CPU sampling authorization control

Get/Set: Enables/disables the use of the basic CPU sampling facility for the CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The authorization control is enabled.

  HWMCA_FALSE
  The authorization control is disabled.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.173.0.*.*

Permit AES key import functions

Get/Set: Enables/disables the importing of AES keys for the associated CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The importing of AES keys is enabled.

  HWMCA_FALSE
  The importing of AES keys is disabled.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.183.0.*.*

Permit DEA key import functions

Get/Set: Enables/disables the importing of DEA keys for the associated CPC Image.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The importing of DEA keys is enabled.

  HWMCA_FALSE
  The importing of DEA keys is disabled.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.184.0.*.*

Description

Get/Set: The description of the profile with a maximum length of 51 (including the null terminator).

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.203.0.*.*

Partition Identifier

Get/Set: The partition identifier for the activation profile.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- Data type for Set: HWMCA_TYPE_INTEGER between 0 and 63, inclusive.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.51.0.*.*

Operating mode

Get/Set: The operating mode value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
  - HWMCA_ESA390_OPERATING_MODE (1)
  - HWMCA_ESA390TPF_OPERATING_MODE (2)
  - HWMCA_CF_OPERATING_MODE (3)
  - HWMCA_LINUX_OPERATING_MODE (4)
  - HWMCA_FMEX_OPERATING_MODE (5)
  - HWMCA_HMEX_OPERATING_MODE (6)

  - HWMCA_HMAS_OPERATING_MODE (7)
  - HWMCA_ZVM_OPERATING_MODE (8)
  - HWMCA_ZAWARE_OPERATING_MODE (9)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.204.0.*.*

Clock type
Get/Set: The clock type assignment for the activation profile.
- Data type for Get/Set: HWMCA_TYPE_INTEGER
  - HWMCA_CLOCK_TYPE_STANDARD (0)
  - HWMCA_CLOCK_TYPE_LPAR (1)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.205.0.*.*

Time offset days
Get/Set: The time offset days for the activation profile.
- Data type for Get/Set: HWMCA_TYPE_INTEGER (0 - 999)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.206.0.*.*

Time offset hours
Get/Set: The time offset hours for the activation profile.
- Data type for Get/Set: HWMCA_TYPE_INTEGER (0 - 23)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.207.0.*.*

Time offset minutes
Get/Set: The time offset minutes for the activation profile.
- Data type for Get/Set: HWMCA_TYPE_INTEGER (0, 15, 30, or 45)
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.208.0.*.*

Time offset increase or decrease
Get/Set: The time offset increase/decrease setting for the activation profile.
- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The local time zone is east of GMT.

  HWMCA_FALSE
  The local time zone is west of GMT.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.209.0.*.*

LICCC validation
Get/Set: Enables/disables whether or not the activation profile must conform to the current LICCC configuration.
- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The activation profile must conform to the current LICCC configuration.

  HWMCA_FALSE
  The activation profile is not required to conform to the current LICCC configuration.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.210.0.*.*

Global performance data control
Get/Set: Enables/disables the global performance data control setting for the activation profile.
- Data type for Get/Set: HWMCA_TYPE_INTEGER

HWMCA_TRUE
The global performance data control is enabled.

HWMCA_FALSE
The global performance data control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.211.0.*.*

Input/Output configuration control

Get/Set: Enables/disables the I/O configuration control setting for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

HWMCA_TRUE
The I/O configuration control is enabled.

HWMCA_FALSE
The I/O configuration control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.212.0.*.*

Cross partition authority control

Get: The cross partition authority control setting for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

HWMCA_TRUE
The cross partition authority control is enabled.

HWMCA_FALSE
The cross partition authority control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.213.0.*.*

Logical partition isolation control

Get/Set: Enables/disables the logical partition isolation control setting for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

HWMCA_TRUE
The logical partition isolation control is enabled.

HWMCA_FALSE
The logical partition isolation control is disabled.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.214.0.*.*

Absolute capping type

Get/Set: The type of absolute capping to perform.

- Data type returned on Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of processors

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.217.0.*

Absolute capping value

Get/Set: The value used for absolute capping (if enabled).

- Data type returned on Get/Set: HWMCA_TYPE_OCTETSTRING

0 None

1-nnnn
Represents the number of processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.218.0.*

Application Assist Processor absolute capping type

Get/Set: The type of absolute capping to perform for Application Assist Processor (AAP) processors.

- Data type returned on Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Application Assist Processor (AAP) processors

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.219.0.*

Application Assist Processor absolute capping value

Get/Set: The value used for Application Assist Processor (AAP) absolute capping.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

0 None

1-nnnn
Represents the number of Application Assist Processor (AAP) processors when capping in number of processors is enabled.

Note: Though this is an integer, value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.220.0.*

Integrated Facility for Linux absolute capping type

Get/Set: The type of absolute capping to perform for Integrated Facility for Linux (IFL) processors.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Integrated Facility for Linux (IFL) processors.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.221.0.*

Integrated Facility for Linux absolute capping value

Get/Set: The value used for Integrated Facility for Linux (IFL) absolute capping (if enabled).

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

0 None

1-nnnn
Represents the number of Integrated Facility for Linux (IFL) processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.222.0.*

Internal Coupling Facility absolute capping type

Get/Set: The type of absolute capping to perform for Internal Coupling Facility (ICF) processors.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Internal Coupling Facility (ICF) processors.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.223.0.*

Internal Coupling Facility absolute capping value

Get/Set: The value used for Internal Coupling Facility (ICF) absolute capping.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

0 Absolute capping not enabled.

1-nnnn
Represents the number of Internal Coupling Facility (ICF) processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.224.0.*

Integrated Information Processor absolute capping type

Get/Set: The type of absolute capping to perform for Integrated Information Processor (zIIP) processors.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

0 None

1 Absolute capping in number of Integrated Information Processor (zIIP) processors.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.225.0.*

Integrated Information Processor absolute capping value

Get/Set: The value used for Integrated Information Processor (zIIP) absolute capping.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING

0 Absolute capping not enabled.

1-nnnn
Represents the number of Integrated Information Processor (zIIP) processors when capping in number of processors is enabled.

Note: Though this is an integer value, it must be specified within an HWMCA_TYPE_OCTETSTRING data type. This was done in case future absolute capping types require fractional units.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.6.0.226.0.*

Load activation profile object

Load activation profile name bindings

Load activation profile object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.x.x.y.z

Where ***x.x.*** equals the attribute identifier for the object, ***y*** equals a unique number for the specific instance of the CPC Object, and *z* equals a unique number for the specific instance of the Load Activation Profile.

Load activation profile attributes

Name

Get:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.1.0.*.*

Object type

Get: This returns the type of object the object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER
  HWMCA_ACT_PROFILE_LOAD
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.22.0.*.*

IPL address

Get/Set:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A value of an empty string is used to indicate that the Load Activation Profile will use next IPL address set by HCD.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.28.0.*.*

IPL parameter

Get/Set:

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING

  Note: A value of an empty string is used to indicate that the Load Activation Profile will use next IPL parameter set by HCD.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.29.0.*.*

IPL type

Get/Set: The IPL type value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
  HWMCA_IPLTYPE_STANDARD
  Indicates that the image activation profile is used to perform a standard load.
  HWMCA_IPLTYPE_SCSI
  Indicates that the image activation profile is used to perform a SCSI load.
  HWMCA_IPLTYPE_SCSIDUMP
  Indicates that the image activation profile is used to perform a SCSI dump.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.52.0.*.*

Worldwide port name

Get/Set: The worldwide port name value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.53.0.*.*

Boot program selector

Get/Set: The boot program selector value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.54.0.*.*

Logical unit number

Get/Set: The logical unit number value for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.55.0.*.*

Boot record logical block address

Get/Set: The boot record logical block address value for the activation profile.

- Data type for get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.56.0.*.*

Operating system specific load parameters

Get/Set: The operating system specific load parameters for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.57.0.*.*

Store Status

Get/Set: The store status setting for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  The store status is performed before the load starts.

  HWMCA_FALSE
  The store status is not performed before the load starts.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.166.0.*.*.*.*

Load Type

Get/Set: The load type for the activation profile.

- Data type for Get/Set: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Clears main storage during the load.

  HWMCA_FALSE
  Performs the load without clearing main storage.

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.167.0.*.*.*.*

Description

Get/Set: The description of the profile with a maximum length of 51 (including the null terminator).

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.7.0.203.0.*.*

Group profile object

Group profile name bindings

Group profile object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.8.0.x.x.y.z
Where x.x. equals the attribute identifier for the object, y equals a unique number for the specific instance of the CPC Object, and z equals a unique number for the specific instance of the Group Profile.

Group profile attributes

Name

Get: This returns the name of object the group profile object identifier represents.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.8.0.1.0.*.*.*.*

Object type

Get: This returns the type of object the group profile object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER HWMCA_ACT_PROFILE_GROUP
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.8.0.22.0.*.*.*.*

Capacity

Get/Set: This returns the capacity value of object the group profile object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.8.0.92.0.*.*.*.*

Description

Get/Set: The description of the profile with a maximum length of 51 (including the null terminator).

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- Data type for Set: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.8.0.203.0.*.*

Capacity record object

Capacity record name bindings

Capacity record object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.x.x.y.z

Where x.x. equals the attribute identifier for the object, y equals a unique number for the specific instance of the Defined CPC Object, and z equals a unique number for the specific instance of the Capacity Record. Additionally, the capacity record itself can be queried using an object identifier of the form 1.3.6.1.4.1.2.6.42.9.0.y.z. When the capacity record itself is queried, it returns a data type of HWMCA_TYPE_OCTETSTRING with the data being an XML string describing all aspects of the record. Refer to Appendix F, "XML descriptions," on page 219 for details on the format of the XML that is returned.

Capacity record attributes

Object type

Get: This returns the type of object the capacity record object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER HWMCA_CAPACITY_RECORD
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.22.0.*.*

Record identifier

Get: This returns the identifier for the capacity record.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.135.0.*.*.*.*

Record type

Get: This returns a value that indicates the type of capacity record.

- Data type returned on Get: HWMCA_TYPE_INTEGER HWMCA_CAPACITY_RECORD_TYPE_CBU HWMCA_CAPACITY_RECORD_TYPE_OOCOD HWMCA_CAPACITY_RECORD_TYPE_PLANNED_EVENT HWMCA_CAPACITY_RECORD_TYPE_LOANER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.136.0.*.*.*.*

Activation status

Get: This returns an indication if any of the resources defined for the record are currently activated.

- Data type returned on Get: HWMCA_TYPE_INTEGER HWMCA_CAPACITY_RECORD_STATUS_NOT_ACTIVATED HWMCA_CAPACITY_RECORD_STATUS_REAL HWMCA_CAPACITY_RECORD_STATUS_TEST HWMCA_CAPACITY_RECORD_STATUS_CAN_BE_ACTIVATED
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.137.0.*.*.*.*

Activation date

Get: Defines the time stamp for when additional capacity for the record was activated.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.138.0.*.*.*.*

Record expiration date

Get: Defines the time stamp for when the capacity record will expire.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.139.0.*.*.*.*

Activation expiration date

Get: Defines the time stamp for when the additional capacity activated for the record will expire and no longer be active.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.140.0.*.*.*.*

Maximum real days

Get: Defines the maximum days that real additional capacity can be activated for the record. A value of -1 indicates that the number of days is unlimited.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.141.0.*.*.*.*

Maximum test days

Get: Defines the maximum days that test additional capacity can be activated for the record. A value of -1 indicates that the number of days is unlimited.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.142.0.*.*.*.*

Remaining real days

Get: Defines the remaining number of days that additional real capacity can be active for the record. A value of -1 indicates that the number of days is unlimited.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.143.0.*.*.*.*

Remaining test days

Get: Defines the remaining number of days that additional test capacity can be active for the record. A value of -1 indicates that the number of days is unlimited.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.144.0.*.*.*.*

Remaining number of real activations

Get: Defines the number of times that real additional capacity can be activated for the record. A value of -1 indicates that activation count is unlimited.

- Data type returned on Get: HWMCA_TYPE_INTEGER

- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.147.0.*.*.*.*

Remaining number of test activations

Get: Defines the number of times that test additional capacity can be activated for the record. A value of -1 indicates that activation count is unlimited.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.9.0.148.0.*.*.*.*

---

z/VM virtual machine object

Z/VM virtual machine name bindings

z/VM virtual machine object identifier

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.x.x..*

Where ***x.x.*** equals the attribute identifier for the object and an * equals a unique number for that specific instance of the z/VM virtual machine.

z/VM virtual machine attributes

Name

Get: This returns the name of the z/VM virtual machine object.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.1.0.*

Parent name

Get (CPC Image's name): This returns the name of the parent CPC Image object.

- Data type returned on Get: HWMCA_TYPE_OCTETSTRING
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.2.0.*

Status error

Get: This returns an indicator of whether the status of the z/VM virtual machine is acceptable.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object is in a state which is not an acceptable status.

  HWMCA_FALSE
  Object is in an acceptable status state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.7.0.*

Busy

Get: This returns an indicator of whether the object is currently busy performing a user initiated task.

- Data type returned on Get: HWMCA_TYPE_INTEGER

  HWMCA_TRUE
  Object in a busy state (currently performing a task).

  HWMCA_FALSE
  Object not in a busy state.
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.8.0.*

Status

Get: This returns a value representing the status of the z/VM virtual machine object.

- Data type returned on Get: HWMCA_TYPE_INTEGER
  One of the following bit values will be set to on:
  - HWMCA_STATUS_OPERATING
  - HWMCA_STATUS_NOT_ACTIVATED
  - HWMCA_STATUS_LINKNOTACTIVE
  - HWMCA_STATUS_NOT_OPERATING
  - HWMCA_STATUS_LOGOFF_TIMEOUT
  - HWMCA_STATUS_FORCED_SLEEP
  - HWMCA_STATUS_STORAGE_EXCEEDED
  - HWMCA_STATUS_UNKNOWN
  - HWMCA_STATUS_IMAGE_NOT_OPERATING
  - HWMCA_STATUS_IMAGE_NOT_ACTIVATED
  - HWMCA_STATUS_IMAGE_NOT_CAPABLE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.10.0.*

Acceptable status

Get: This returns a value that represents the status values that are to be considered acceptable for the z/VM virtual machine object.

- Data type returned on Get: HWMCA_TYPE_INTEGER
- Data type for Set: HWMCA_TYPE_INTEGER
  One or more of the following bit values will be set to on:
  - HWMCA_STATUS_OPERATING
  - HWMCA_STATUS_NOT_ACTIVATED
  - HWMCA_STATUS_LINKNOTACTIVE
  - HWMCA_STATUS_NOT_OPERATING
  - HWMCA_STATUS_LOGOFF_TIMEOUT
  - HWMCA_STATUS_FORCED_SLEEP
  - HWMCA_STATUS_STORAGE_EXCEEDED
  - HWMCA_STATUS_UNKNOWN
  - HWMCA_STATUS_IMAGE_NOT_OPERATING
  - HWMCA_STATUS_IMAGE_NOT_ACTIVATED
  - HWMCA_STATUS_IMAGE_NOT_CAPABLE
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.11.0.*

Object type

Get: This returns the type of object the z/VM virtual machine object identifier represents.

- Data type returned on Get: HWMCA_TYPE_INTEGER HWMCA_CAPACITY_RECORD
- SNMP MIB Name: 1.3.6.1.4.1.2.6.42.10.0.22.0.*.*

z/VM virtual machine commands

Activate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.1 (HWMCA_ACTIVATE_COMMAND)

Deactivate

SNMP MIB Name: 1.3.6.1.4.1.2.6.42.4.2 (HWMCA_DEACTIVATE_COMMAND)

z/VM virtual machine notifications

Status change (HWMCA_EVENT_STATUS_CHANGE)

- An HWMCA_TYPE_INTEGER that specifies the new status value.
- An HWMCA_TYPE_INTEGER that specifies the old status value.

Object created (HWMCA_EVENT_CREATED)

This event has no additional data. The object identifier can be used with the HwmcaGet to get any data required for this newly created object.

Object destruction (HWMCA_EVENT_DESTROYED)

This event has no additional data.

Object entered an exception state (HWMCA_EVENT_EXCEPTION_STATE)

- An HWMCA_TYPE_INTEGER that specifies whether the object is entering into an exception state (HWMCA_TRUE) or leaving an exception state (HWMCA_FALSE).
- An HWMCA_TYPE_INTEGER that specifies the status value for the object.

Chapter 5. REXX management functions

ACTZSNMP

ACTZSNMP is a Dynamic Link Library (DLL) package of OS/2 REXX External Functions written in the C language. The ACTZSNMP dynalink gives the REXX application equivalent function as an application written in C.

Likewise, HWMCAORX is a Dynamic Link Library (DLL) package of 32- bit Windows Object REXX External Functions written in the C language. The HWMCAORX dynalink gives the Object REXX application equivalent function as an application written in C.

REXX initialization functions

RxHwmcaLoadFuncs
To use a REXX Management Function, you must first register the function with the REXX RxFuncAdd function. The *RxHwmcaLoadFuncs* ACTZSNMP (HWMCAORX for Object REXX for Windows) function automatically loads all the other REXX functions.

RxHwmcaDefineVars
The *RxHwmcaDefineVars* function can be called to define REXX variables with the same values which exist for a C application. These variables are shown in "Constant definitions" on page 43 .

Note: An example of these functions is shown in "Data exchange APIs (REXX sample)" on page 167.

Data exchange functions

The purpose of the REXX Data Exchange Functions is to allow REXX applications, local or remote, the ability to exchange data related to the objects that the Console application manages. Specifically, this support will allow REXX applications to request the Console to:

- Query (Get/Get-Next) the attributes of objects
- Change (Set) certain attributes of objects
- Receive notification of significant events occurring to objects
- Generate enterprise-specific Simple Network Management Protocol traps for significant events occurring to objects.

The REXX Data Exchange Functions interface to the Data Exchange APIs, which use the Simple Network Management Protocol (SNMP), as the transport mechanism. The attributes of objects can be queried/changed through the underlying SNMP Set, Get, Get-Next requests, while event notification is accomplished through the use of the enterprise-specific SNMP Trap message.

The REXX External Functions provide the REXX programmer the same capability that exists for the Data Exchange and Commands APIs.

Specifically, the set of REXX Data Exchange Functions consists of:

RxHwmcaInitialize
Used to perform some initialization tasks necessary for the remainder of the REXX Data Exchange Functions set and the REXX Data Command Functions.

RxHwmcaGet
Used to perform a query or Get request for a specified object or object attribute.

RxHwmcaGetNext
Used to perform a query-next or Get-next request for an object or object attributes that occurs next in the lexical sequence of objects managed by the Console.

RxHwmcaSet
Used to perform a change or Set request for a specified object attribute.

RxHwmcaWaitEvent
Used to wait for a specified period of time (or forever) for an event notification.

RxHwmcaTerminate
Used to perform any cleanup tasks required by any of the other APIs in the set.

RxHwmcaBuildId
A convenience routine that can be used to construct an object identifier for any object supported by the Console.

RxHwmcaBuildAttributeId
A convenience routine that can be used to construct an attribute object identifier for any object supported by the Console, based on the object identifier of the object itself.

Refer to the following pages for detailed information about these functions.

RxHwmcaInitialize

Used to perform any initialization tasks required in order for the remainder of the functions to operate correctly. The parameters required for this function are:

INITVAR
Is REQUIRED to be present and MUST be coded as a stem variable. This variable defines all the information that is required for the Data Exchange APIs to perform the initialization request. The following tail parts of the stem variable must be initialized before the *RxHwmcaInitialize* function is called:

INITVAR.TARGET
Must contain the host name or internet address for the target Data Exchange APIs.

INITVAR.COMMUNITY
The community name that is to be used for SNMP request made to the target Console. (Refer to Chapter 6, “Configuring for the data exchange APIs,” on page 191 for more information regarding the community name used in SNMP requests.)

INITVAR.EVENTMASK
If you are going to be using the *RxHwmcaWaitEvent* call, the EVENTMASK tail should contain one or more of the events defined in “HwmcaInitialize” on page 5.

Note: Care should be used when trying to use the same **INITVAR** stem variable for *RxHwmcaWaitEvent* calls in addition to the rest of the APIs in the set. Events associated with a particular **INITVAR** stem variable will be queued until retrieved with *RxHwmcaWaitEvent* or until another API, such as *RxHwmcaGet*, is called. Therefore, making calls, such as *RxHwmcaGet*, will cause any queued events to be discarded and lost.

When both *RxHwmcaWaitEvent* and other calls need to be made, an application should perform two *RxHwmcaInitialize* calls using two distinct **INITVAR** stem variables. The application can then use one of the **INITVAR** stem variables for only *RxHwmcaWaitEvent* calls and the other **INITVAR** stem variable for the other API calls.

INITVAR.RESERVED
A reserved field and must be set to zero for the *RxHwmcaInitialize* function if the **HWMCA_QUALIFIER_SPECIFIED** event mask flag is not specified n the **INITVAR.EVENTMASK** field. If the **HWMCA_QUALIFIER_SPECIFIED** event mask flag is specified, then this field should contain the name of a stem variable, such as

'**QUALDATA.**', that provides additional event qualification information. This stem variable should be specified in the following manner.

QUALDATA.0
Contains the number of event qualification information provided in the event qualification stem variable.

QUALDATA.n.MASK
This field should be set to the event mask flag that is being qualified. Only one event mask flag should be specified in this field. For example, **HWMCA_EVENT_OPSYS_MESSAGE** should be specified when qualifying operating system message event notifications.

QUALDATA.n.TYPE
This field is used to indicate the type of event qualification information being provided. The following event qualification types are currently supported.

HWMCA_QUALIFIER_TYPE_NAME
This value is used to indicate that the event qualification data is the null terminated name of the managed object, which is specified in the **QUALDATA.n.DATA** variable. Event qualification information that specifies this event qualification type can be used to limit event notifications for the specified event mask to those associated with a managed object with the specified name.

After a successful call to the *RxHwmcaInitialize* function this field should not be altered in any way. If the same stem variable is reused for another *RxHwmcaInitialize* call after the *RxHwmcaTerminate* call has been made, this field must be reset appropriately.

TIMEOUT
Used to specify the amount of time that the REXX application wants to wait for the *RxHwmcaInitialize* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *RxHwmcaInitialize* call returns a return code value to the REXX application. This return code lets the REXX application know if the initialization request was successfully delivered and processed by the Hardware Management Console Application. A value defined by variable **HWMCA_DE_NO_ERROR** indicates successful completion.

The stem variable defined for the *RxHwmcaInitialize* call should be left alone and other information will be added by the *RxHwmcaInitialize* function. It is important that this information be left intact and accessible, since it must be passed as a parameter on almost all of the calls.

RxHwmcaGet

Used to retrieve data associated with a specific object attribute. The parameters required for this call are:

INITVAR
The stem variable that was used on the *RxHwmcaInitialize* call.

OBJECTID
The object ID variable for which the data is to be retrieved. Refer to Chapter 4, "Console application managed objects," on page 75 for more information about the object identifiers that the Console manages.

OUTPUT
Defines the stem variable which will contain the actual information returned by *RxHwmcaGet* API. This parameter MUST be a stem variable and the information returned will be as follows:

OUTPUT.0
Contains the number of occurrences of the *TYPE* and *DATA* tail variables.

OUTPUT.n.TYPE
Contains a value which defines the type of data contained in the *DATA* tail variable. Possible values are:

HWMCA_TYPE_INTEGER
Represents a number value.

HWMCA_TYPE_OCTETSTRING
Represents a string value.

HWMCA_TYPE_NULL
Used to denote that no value is present.

HWMCA_TYPE_IPADDRESS
Represents a 32- bit Internet address in host byte order.

OUTPUT.n.DATA
Contains the actual data of the above type.

TIMEOUT
Used to specify the amount of time that the REXX application wants to wait for the *RxHwmcaGet* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *RxHwmcaGet* call returns a return code value to the REXX application. This return code lets the REXX application know if the get request was successfully delivered and processed by the Console application. A value defined by variable **HWMCA_DE_NO_ERROR** indicates successful completion.

RxHwmcaGetNext

Used to retrieve the data associated with the object attribute that occurs next in the lexical sequence of objects, based on a specified object identifier. The parameters specified for the call are identical to those specified for the *RxHwmcaGet* call with two subtle differences.

1. The meaning of the **OBJECTID** variable is used as the base for the Get-Next operation, as opposed to having its object data retrieved.
2. Two pairs of *TYPE* and *DATA* variables will be returned in the output stem variable. The first is the object identifier string for the object whose data is being returned and the second is for the data itself.

RxHwmcaSet

The *RxHwmcaSet* call is used to change or set the data associated with a specific object attribute. The parameters specified for the call are:

INITVAR
The stem variable that was used on the *RxHwmcaInitialize* call.

OBJECTID
Object identifier variable for which the data is to be set. Refer to Chapter 4, “Console application managed objects,” on page 75 for more information about the object identifiers that the Console manages.

DATATYPE
Type of data represented by the Data parameter. Possible values are represented by the variables **HWMCA_TYPE_INTEGER** and **HWMCA_TYPE_OCTETSTRING**.

DATA Actual data that will be set in the object defined by **OBJECTID**. Refer to Chapter 4, “Console application managed objects,” on page 75 for more information about the object identifiers that the Console manages.

TIMEOUT
Used to specify the amount of time that the REXX application wants to wait for the *RxHwmcaSet* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

RxHwmcaWaitEvent

Used to wait for event notification for objects managed by the Console Application. The REXX application specifies the events that it wants to receive through the use of the *EVENTMASK* tail variable in the **INITVAR** variable. The parameters specified for this call are:

INITVAR
The stem variable that was used on the *RxHwmcaInitialize* call.

OUTPUT
Defines the stem variable which will contain the actual information returned by *RxHwmcaWaitEvent* function. This parameter MUST be a stem variable and the information returned will be as follows:

OUTPUT.0
Contains the number of occurrences of the *TYPE* and *DATA* tail variables.

OUTPUT.n.TYPE
Contains a value which defines the type of data contained in the *DATA* tail variable. Possible values are:

HWMCA_TYPE_INTEGER
Represents a number value.

HWMCA_TYPE_OCTESTRING
Represents a string value.

HWMCA_TYPE_NULL
Used to denote that no value is present.

HWMCA_TYPE_IPADDRESS
Represents a 32- bit internet address in host byte order.

OUTPUT.n.DATA
Contains the actual data of the above type.

TIMEOUT
Used to specify the amount of time that the REXX application wants to wait for the *RxHwmcaWaitEvent* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *RxHwmcaWaitEvent* function returns a return code value to the REXX application. This return code lets the REXX application know if any errors occurred while waiting for the event notification. A value of **HWMCA_DE_NO_ERROR** indicates successful completion. A value of **HWMCA_DE_TIMEOUT** indicates that no event notifications were present in the specified timeout period.

On successful completion of the *RxHwmcaWaitEvent* function, the stem variable **OUTPUT** is populated with a series of one or more occurrences of the *TYPE* and *DATA* tail variables.

- An **OUTPUT.1.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.1.DATA** that specifies the object identifier of the object that the event notification pertains to,
- An **OUTPUT.2.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.2.DATA** that specifies the event notification type for this event, and
- Any additional data for the event notification type, as specified below.

The additional data for each of the event notification types are:

HWMCA_EVENT_COMMAND_RESPONSE

Used to notify the REXX application of completion information for a command that has been initiated through the use of the REXX Command function.

The additional data for this event consists of six occurrences of the **OUTPUT** stem variable **TYPE\DATA** pair that describe the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.3.DATA** that specifies the object identifier of the command completed attribute of the target object for which this command response event has been generated.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.4.DATA** that specifies the object identifier of the command for which this command response event has been generated.
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the command return code attribute of the target object for which this command response event has been generated.
4. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.6.DATA** that specifies the return code value to be used to determine the success or failure of the command request that is associated with this command response event.
5. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.7.DATA** that specifies the object identifier of the last command response attribute of the target object for which this command response event has been generated.
6. An **OUTPUT.8.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.8.DATA** that specifies whether or not this is the last **HWMCA_EVENT_COMMAND_RESPONSE** event that will be issued for this command. A **DATA** value of **HWMCA_TRUE** indicates this event as the last, while a value of **HWMCA_FALSE** indicates that more **HWMCA_EVENT_COMMAND_RESPONSE** events will be forthcoming.
7. An **OUTPUT.9.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.9.DATA** that specifies the name of the object that is associated with this command response event.
8. An **OUTPUT.10.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.10.DATA** that specifies the command correlator.

   Note: This field will only be present if the command was invoked with a correlator specified.

HWMCA_EVENT_MESSAGE

Used to notify the REXX application that an object managed by the Console has a new or refreshed message. This event is generated only for the base objects and not for copies of objects within user defined groups.

This event is returned to the application when any combination of the following values is used in the *EVENTMASK* tail of the **INITVAR** parameter of the *RxHwmcaInitialize* call:
- HWMCA_EVENT_MESSAGE
- HWMCA_EVENT_HARDWARE_MESSAGE
- HWMCA_EVENT_OPSYS_MESSAGE

If the **HWMCA_EVENT_MESSAGE** value is specified in the *EVENTMASK* tail of the **INITVAR** parameter, then the application will be notified of both hardware and operating system message events.

If only the **HWMCA_EVENT_HARDWARE_MESSAGE** or **HWMCA_EVENT_OPSYS_MESSAGE** value is specified in the *EVENTMASK* tail of the **INITVAR** parameter, then the application will be notified only of hardware or operating system message events, respectively.

In addition, the **HWMCA_EVENT_NO_REFRESH_MESSAGE** value can be specified in conjunction with the above values to control whether or not the application should be notified of **HWMCA_EVENT_MESSAGE** events for refreshed messages. If the **HWMCA_EVENT_NO_REFRESH_MESSAGE** value is specified in the *EVENTMASK* field of the **INITVAR** parameter, then the application will not be notified of **HWMCA_EVENT_MESSAGE** events for refreshed messages.

The additional data for this event can take on two different formats. The format being received can be determined through examining the **OUTPUT.4.TYPE/DATA** pair. The remaining object identifier/value pairs for each of the two formats follows:

1. An **OUTPUT.3.TYPE:** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.3.DATA** that specifies the object identifier of the message type attribute of the object for which this message event has been generated.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.4.DATA** that specifies whether the message is a hardware or operating system message (**HWMCA_HARDWARE_MESSAGE** or **HWMCA_OPSYS_MESSAGE**)

   The remaining **OUTPUT.n.TYPE/DATA** for hardware messages (**HWMCA_HARDWARE_MESSAGE**) is:

   a. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the message text attribute of the object for which this message event has been generated.

   b. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_OCTETSTRING** and a **OUTPUT.6.DATA** that specifies the new or refreshed hardware message text.

   c. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.7.DATA** that specifies the object identifier of the message refresh attribute of the object for which this message event has been generated.

   d. An **OUTPUT.8.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.8.DATA** that specifies whether the message is a new (**HWMCA_FALSE**) or refresh message (**HWMCA_TRUE**).

   e. An **OUTPUT.9.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.9.DATA** that specifies the time stamp of the new or refreshed hardware message.

   f. An **OUTPUT.10.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.10.DATA** that specifies the names of the CPC Image object(s) associated with the object that generated the new or refreshed hardware message. This **HWMCA_TYPE_OCTETSTRING** is a null terminated, blank delimited list of the CPC Image name(s).

      When receiving this event from a Support Element Console, this value contains the name(s) of the CPC Images that are running on the CPC that the Support Element Console is controlling.

      When receiving this event from a Hardware Management Console, this value:

      - Contains no CPC Image names for hardware messages for the Hardware Management Console itself
      - Contains no CPC Image names for Optical Network related hardware messages
      - Contains the name(s) of the CPC Images that are running on the CPC that the hardware message pertains to.

   g. An **OUTPUT.11.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.11.DATA** that specifies the name of the object that is associated with this event.

   The remaining **OUTPUT.n.TYPE/DATA** for operating system messages (**HWMCA_OPSYS_MESSAGE**) are:

   a. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the message text attribute of the object for which this message event has been generated.

   b. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.6.DATA** that specifies the new or refreshed operating system message text.

      Note: If the operating system message text contains multiple lines, then each additional line is delimited from the next line with the character sequence of a carriage return (\r) and a new line (\n).

   c. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.7.DATA** that specifies the object identifier of the message identifier attribute of the object for which this message event has been generated.

d. An **OUTPUT.8.TYPE** of HWMCA_TYPE_OCTETSTRING and an OUTPUT.8.DATA that specifies the message identifier of the new operating system message.

e. An **OUTPUT.9.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.9.DATA** that specifies the object identifier of the message date attribute of the object for which this message event has been generated.

f. An **OUTPUT.10.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.10.DATA** that specifies the date of the new operating system message.

g. An **OUTPUT.11.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.11.DATA** that specifies the object identifier of the message time attribute of the object for which this message event has been generated

h. An **OUTPUT.12.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.12.DATA** that specifies the time of the new operating system message.

i. An **OUTPUT.13.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.13.DATA** that specifies the object identifier of the message alarm attribute of the object for which this message event has been generated.

j. An **OUTPUT.14.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.14.DATA** that specifies whether the new operating system message should cause the alarm to be sounded (**HWMCA_TRUE** or **HWMCA_FALSE**).

k. An **OUTPUT.15.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.15.DATA** that specifies the object identifier of the message priority attribute of the object for which this message event has been generated.

l. An **OUTPUT.16.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.16.DATA** that specifies whether the new operating system message is a priority message or not (**HWMCA_TRUE** or **HWMCA_FALSE**).

m. An **OUTPUT.17.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.17.DATA** that specifies the object identifier of the message held attribute of the object for which this message event has been generated.

n. An **OUTPUT.18.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.18.DATA** that specifies whether the new operating system message is a held message or not (**HWMCA_TRUE** or **HWMCA_FALSE**).

o. An **OUTPUT.19.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.19.DATA** that specifies the object identifier of the message prompt text attribute of the object for which this message event has been generated.

p. An **OUTPUT.20.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.20.DATA** that specifies the prompt text that should be associated with the new operating system message or an **HWMCA_TYPE_NULL** indicating that there is no prompt text for this new operating system message.

q. An **OUTPUT.21.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.21.DATA** that specifies the object identifier of the message operating system name attribute of the object for which this message event has been generated.

r. An **OUTPUT.22.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.22.DATA** that specifies the operating system name that should be associated with the new operating system message or an **HWMCA_TYPE_NULL** indicating that there is no operating system name for this new operating system message.

s. An **OUTPUT.23.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.23.DATA** that specifies the object identifier of the message refresh attribute of the object for which this message event has been generated.

t. An **OUTPUT.24.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.24.DATA** that specifies whether the message is a new (**HWMCA_FALSE**) or refresh message (**HWMCA_TRUE**).

u. An **OUTPUT.25.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.25.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_STATUS_CHANGE

Used to notify the REXX application that an object managed by the Console has changed status. This event is generated only for the base objects and not for copies of objects within user defined groups.

The additional data for this event consists of four **OUTPUT** stem variable **TYPE/DATA** pairs that describe the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.3.DATA** that specifies the object identifier of the status attribute of the object for which this status change event has been generated.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.4.DATA** that specifies the new status value, and
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the status attribute of the object for which this status change event has been generated.
4. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.6.DATA** that specifies the old status value.
5. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.7.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_NAME_CHANGE

Used to notify the REXX application that an object managed by the Console has had a name change. This event notification can be useful when a REXX application retains the object identifiers for objects it is interested in, since the name of an object is used to build the unique portion of the object identifier. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of four **OUTPUT** stem variable **TYPE/DATA** pairs that describe the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.3.DATA** that specifies the object identifier of the name attribute of the object for which this name change event has been generated.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.4.DATA** that specifies the new object name, and
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the name attribute of the object for which this name change event has been generated.
4. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.6.DATA** that specifies the old object name.

HWMCA_EVENT_ACTIVATE_PROF_CHANGE

Used to notify the REXX application that an object managed by the Console has changed which activation profile is associated with it.

The additional data for this event consists of four **OUTPUT** stem variable **TYPE/DATA** pairs that describe the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.3.DATA** that specifies the object identifier of the activation profile attribute of the object for which this activation profile change event has been generated.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.4.DATA** that specifies the new activation profile name, and
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the activation profile attribute of the object for which this activation profile change event has been generated.

4. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.6.DATA** that specifies the old activation profile name.
5. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.7.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_CREATED

Used to notify the REXX application that a new object managed by the Console has been defined or instantiated.

The additional data for this event consists of a single **OUTPUT** stem variable **TYPE/DATA** pair that has an **OUTPUT.3.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.3.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_DESTROYED

Used to notify the REXX application that an object managed by the Console has been undefined.

The additional data for this event consists of a single **OUTPUT** stem variable **TYPE/DATA** pair that has an **OUTPUT.3.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.3.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_EXCEPTION_STATE

Used to notify the REXX application that an object managed by the Console has either entered into or out of an exception state. An object is considered in an exception state when its status is not considered acceptable as defined by the object's acceptable status attribute. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of four **OUTPUT** stem variable **TYPE/DATA** pairs that describe the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.3.DATA** that specifies the object identifier of the status error attribute of the object for which this exception state event has been generated.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.4.DATA** that specifies whether the object is entering into an exception state (**HWMCA_TRUE**) or leaving an exception state (**HWMCA_FALSE**).
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OBJECTID** and an **OUTPUT.5.DATA** that specifies the object identifier of the status attribute of the object for which this exception state event has been generated.
4. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.6.DATA** that specifies the status value for the object.
5. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.7.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_ENDED

Used to notify the REXX application that the Console application is ending.

The additional data for this event consists of a single **OUTPUT** stem variable **TYPE/DATA** pair that has an **OUTPUT.3.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.3.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_HARDWARE_MESSAGE_DELETE

Used to notify the REXX application that a hardware message associated with an object managed by the Console application or the Console application itself has been deleted. This event is generated only for the base objects and not for copies of objects within user-defined groups.

The additional data for this event consists of five **OUTPUT** stem variable **TYPE/DATA** pairs that describe the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.3.DATA** that specifies that the message being deleted is a hardware message (HWMCA_HARDWARE_MESSAGE).
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.4.DATA** that specifies the message text for the hardware message being deleted.
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_INTEGER** and an **OUTPUT.5.DATA** that is always set to HWMCA_FALSE for this event.
4. An **OUTPUT.6.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.6.DATA** that specifies the time stamp of the hardware message being deleted.
5. An **OUTPUT.7.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.7.DATA** that specifies the names of the CPC Image object(s) associated with the object for which the hardware message is being deleted. This **HWMCA_TYPE_OCTETSTRING** is a null terminated, blank delimited list of the CPC Image name(s).

   When receiving this event from a Support Element Console, this value contains the name(s) of the CPC Images that are running on the CPC that the Support Element Console is controlling.

   When receiving this event from a Hardware Management Console, this value:
   - Contains no CPC Image names for hardware messages for the Hardware Management Console itself
   - Contains no CPC Image names for Optical Network related hardware messages
   - Contains the name(s) of the CPC Images that are running on the CPC that the hardware message pertains to.
6. An **OUTPUT.8.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.8.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_SECURITY_EVENT

Used to notify the REXX application that a security event has been logged.

The additional data for this event consists of three OUTPUT stem variable TYPE/DATA pairs that describes the following:

1. An **OUTPUT.3.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.3.DATA** that specifies the time stamp of the security log.
2. An **OUTPUT.4.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.4.DATA** that specifies the text of the security log.
3. An **OUTPUT.5.TYPE** of **HWMCA_TYPE_OCTETSTRING** and an **OUTPUT.5.DATA** that specifies the name of the object that is associated with this event.

HWMCA_EVENT_CAPACITY_CHANGE

Used to notify the REXX application that the processing capacity for a Defined CPC object has changed in some manner. The additional data for this event consists of two OUTPUT stem variable TYPE/DATA pairs that describe the following:

1. An OUTPUT.3.TYPE of HWMCA_TYPE_INTEGER and an OUTPUT.3.DATA that specifies the type of capacity change that occurred, using one of the following constants:
   - HWMCA_CAPACITY_FENCED_BOOK A processor book has been fenced and is not longer usable.
   - HWMCA_CAPACITY_DEFECTIVE_PROCESSOR A processor has become defective.
   - HWMCA_CAPACITY_CONCURRENT_BOOK_REPLACE A concurrent processor book replacement has been performed.
   - HWMCA_CAPACITY_CONCURRENT_BOOK_ADD A concurrent processor book addition has been performed.
   - HWMCA_CAPACITY_CHECK_STOP A processor has gone into a check stopped state.
   - HWMCA_CAPACITY_CHANGES_ALLOWED A user has configured the APIs to be allowed to perform capacity changes.

- HWMCA_CAPACITY_CHANGES_NOT_ALLOWED A user has configured the APIs to no longer be allowed to perform capacity changes.

2. An OUTPUT.4.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.4.DATA that specifies the name of the object that the event pertains to (in this case a Defined CPC object).

HWMCA_EVENT_CAPACITY_RECORD_CHANGE

Used to notify the REXX application that a change has occurred to a temporary capacity record. The additional data for this event consists of three OUTPUT stem variable TYPE/DATA pairs that describe the following:

1. An OUTPUT.3.TYPE of HWMCA_TYPE_INTEGER and an OUTPUT.3.DATA that specifies the type of capacity record change that occurred, using one of the following constants:
   - HWMCA_CAPACITY_RECORD_ADD The capacity record has been added to the machine.
   - HWMCA_CAPACITY_RECORD_DELTA The capacity record has been modified.
   - HWMCA_CAPACITY_RECORD_DELETE The capacity record has been deleted.
   - HWMCA_CAPACITY_RECORD_ACCOUNTING
   - HWMCA_CAPACITY_ACTIVATION_LEVEL The capacity record has changed it's level of activation (either more resources from this record have been added or removed from the machine).
   - HWMCA_CAPACITY_PRIORITY_PENDING Additional capacity has been added for the capacity record, with priority, but not enough resources were available to allow for all the capacity specified to be put into effect. As resources become available they will be added for this record in order to completely satisfy the original request for additional capacity.
   - HWMCA_CAPACITY_RECORD_OTHERThe capacity record has changed in some other manner.
2. An OUTPUT.4.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.4.DATA for the temporary capacity record that has changed.
3. An OUTPUT.5.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.5.DATA that specifies the name of the object that the event pertains to (in this case a Defined CPC object).

HWMCA_EVENT_DISABLED_WAIT

Used to notify the REXX application that a CPC Image object has entered a disabled wait state. The additional data for this event consists of six OUTPUT stem variable TYPE/DATA pairs that describe the following:

1. An OUTPUT.3.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.3.DATA for the name of the Defined CPC that is associated with the CPC Image that entered a disabled wait state.
2. An OUTPUT.4.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.4.DATA for the disabled wait PSW value.
3. An OUTPUT.5.TYPE of HWMCA_TYPE_INTEGER and an OUTPUT.5.DATA for the partition identifier of the CPC Image that entered a disabled wait state.
4. An OUTPUT.6.TYPE of HWMCA_TYPE_INTEGER and an OUTPUT.6.DATA for number of the processor that entered a disabled wait state.
5. An OUTPUT.7.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.7.DATA for the serial number of the Defined CPC that is associated with the CPC Image that entered a disabled wait state.
6. An OUTPUT.8.TYPE of HWMCA_TYPE_OCTETSTRING and an OUTPUT.8.DATA that specifies the name of the object that the event pertains to (in this case a CPC Image object).

TIMEOUT

Used to specify the amount of time that the REXX application waits for the *RxHwmcaWaitEvent* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

RxHwmcaTerminate

Used to perform any cleanup tasks required by any of the other REXX Data Exchange and Command functions. The parameters required for this function are:

INITVAR
This parameter is the stem variable that was used on the *RxHwmcaInitialize* call.

TIMEOUT
This parameter is used to specify the amount of time that the REXX application wants to wait for the *RxHwmcaTerminate* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

The *RxHwmcaTerminate* call returns a return code value to the REXX application. This return code lets the REXX application know if the termination request was successfully delivered and processed by the Console application. A value defined by variable **HWMCA_DE_NO_ERROR** indicates successful completion.

Once the *RxHwmcaTerminate* has successfully been called, the stem variable **INITVAR**, can be used for other purposes.

RxHwmcaBuildId

A convenience routine provided to aid the application program in constructing an object identifier for any object supported by the Console. The arguments specified for this API are:

BUFFER
A variable where the built object identifier string is to be placed.

PREFIX
The prefix string to be used for the object identifier to be built. Any of the valid prefixes defined by the *RxHwmcaDefineVars* call can be used, such as:
- HWMCA_CONSOLE_ID
- HWMCA_CFG_CPC_GROUP_ID
- HWMCA_CFG_CPC_ID
- HWMCA_CPC_IMAGE_GROUP_ID
- HWMCA_CPC_IMAGE_ID
- HWMCA_GROUPS_GROUP_ID
- HWMCA_GROUPS_OBJECT_ID
- HWMCA_COMMAND_PREFIX
- HWMCA_ACT_RESET_OBJECT_ID
- HWMCA_ACT_IMAGE_OBJECT_ID
- HWMCA_ACT_LOAD_OBJECT_ID
- HWMCA_ACT_GROUP_OBJECT_ID
- HWMCA_CAPACITY_RECORD_OBJECT_ID
- HWMCA_CFG_VM_GROUP_ID
- HWMCA_VM_OBJECT_ID

ATTRIBUTE
The attribute suffix string to be used for the object identifier to be built. This can be omitted when building an identifier for an object itself, as opposed to an attribute object identifier.

GROUPNAME
The group name to be used for building the object identifier. This can be omitted when building an object identifier for a predefined group or an object contained in a predefined group.

OBJECTNAME
The object name to be used for building the object identifier. This can be omitted when building an object identifier for a group object.

Note: Refer to "Console application object identifier conventions" on page 75 for more information about the conventions used for the object identifiers for objects managed by the Console.

RxHwmcaBuildAttributeId

A convenience routine provided to aid the REXX programmer in constructing an attribute object identifier for any object supported by the Console, based on the object identifier of the object itself. The parameters specified for this API are:

BUFFER
A variable where the built object identifier string is to be placed.

OBJECTID
The object identifier of the object for which the attribute identifier is to be built.

ATTRIBUTE
The attribute suffix string to be used for the object identifier to be built.

Note: Refer to "Console application object identifier conventions" on page 75 for more information about the conventions used for the object identifiers for objects managed by the Console.

Commands API

RxHwmcaCommand

Used to perform a command against a specific object managed by the Console Application. The parameters specified for the call are:

INITVAR
The stem variable that was used on the RxHwmcaInitialize call.

OBJECTID
The object identifier variable for which the data is to be retrieved.

COMMANDID
A variable containing the command that is to be executed. Valid values for this argument are:

- HWMCA_ACTIVATE_COMMAND
- HWMCA_DEACTIVATE_COMMAND
- HWMCA_SEND_OPSYS_COMMAND
- HWMCA_RESETNORMAL_COMMAND
- HWMCA_START_COMMAND
- HWMCA_STOP_COMMAND
- HWMCA_PSWRESTART_COMMAND
- HWMCA_LOAD_COMMAND
- HWMCA_HW_MESSAGE_REFRESH_COMMAND
- HWMCA_RESETCLEAR_COMMAND
- HWMCA_HW_MESSAGE_DELETE_COMMAND
- HWMCA_ACTIVATE_CBU_COMMAND
- HWMCA_UNDO_CBU_COMMAND
- HWMCA_IMPORT_PROFILE_COMMAND
- HWMCA_EXPORT_PROFILE_COMMAND
- HWMCA_RESERVE_COMMAND
- HWMCA_EXTERNAL_INTERRUPT_COMMAND
- HWMCA_SCSI_LOAD_COMMAND
- HWMCA_SCSI_DUMP_COMMAND
- HWMCA_SHUTDOWN_RESTART_COMMAND
- HWMCA_ACTIVATE_OOCOD_COMMAND
- HWMCA_UNDO_OOCOD_COMMAND
- HWMCA_ADD_CAPACITY_COMMAND
- HWMCA_REMOVE_CAPACITY_COMMAND
- HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND

- HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND
- HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_ COMMAND
- HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_COMMAND
- HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_ COMMAND

CMDINPUT

Defines the stem variable that contains the actual command used to represent the arguments to be passed to the specified command.

CMDINPUT.0

Contains the number of occurrences of the *TYPE* and *DATA* tail variables.

CMDINPUT.n.TYPE

Contains a value which defines the type of data contained in the *DATA* tail variable.

CMDINPUT.n.DATA

Contains the actual data of the above type.

The acceptable and required arguments for each command are as follows:

HWMCA_ACTIVATE_COMMAND

No arguments are required, but the following arguments can optionally be specified:

Activation profile name

Name of the activation profile to be used for the Activate command. The default is to use the profile name specified in the Activation profile name attribute for the specified object.

Force indicator

An indicator used to request conditional processing of the Activate command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Either one or both of these arguments can be specified, meaning the **CMDINPUT.0** can be 0 - 2; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument, such that the default will be used, the *TYPE* tail variable should be set to **HWMCA_TYPE_NULL** and the *DATA* tail variable should be set to the null string (for example, "").

The default for any argument can be overridden by specifying the *TYPE* and *DATA* tail variables as follows:

Activation profile name

CMDINPUT.1.TYPE

Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.1.DATA

Should contain the Activation profile name.

Force indicator

CMDINPUT.2.TYPE

Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.2.DATA

Should contain HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally.

HWMCA_DEACTIVATE_COMMAND
No arguments are required, but optionally a Force indicator can be specified. If this argument is not specified, then the default is to unconditionally perform the command. This implies that **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally.

HWMCA_RESETNORMAL_COMMAND
No arguments are required, but optionally a Force indicator can be specified. If this argument is not specified, then the default is to unconditionally perform the command. This implies that **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally.

HWMCA_RESETCLEAR_COMMAND
No arguments are required, but optionally a Force indicator can be specified. If this argument is not specified, then the default is to unconditionally perform the command. This implies that **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMINPUT.1.DATA** variable should contain the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally.

HWMCA_START_COMMAND
No arguments are accepted or required.

HWMCA_STOP_COMMAND
No arguments are accepted or required.

HWMCA_PSWRESTART_COMMAND
No arguments are accepted or required.

HWMCA_SEND_OPSYS_COMMAND
This command requires two arguments. The first is an indication of whether this is a priority operating system command and the second is the text of the operating system command. This implies that the **CMDINPUT.0** variable should contain a 2. The first **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain **HWMCA_TRUE** for priority operating system commands or **HWMCA_FALSE** for non-priority operating system commands. The second **CMDINPUT.2.TYPE** variable should be set to **HWMCA_TYPE_OCTETSTRING** and the **CMDINPUT.2.DATA** variable should contain the operating system command itself.

HWMCA_LOAD_COMMAND
No arguments are required, but the following arguments can optionally be specified:

Load address
Hexadecimal address to be used when performing the Load. The default will be to use the Load address last used when a Load was performed for the object.

Load parameter
Parameter string to be used when performing the Load. The default will be to use the Load parameter last used when a Load was performed for the object.

Clear indicator
Whether or not memory should be cleared before performing the Load. The default is to clear memory before performing the Load.

Timeout
Amount of time (in seconds) to wait for the Load to complete. The default timeout is 60 seconds.

Store status indicator
Whether or not status should be stored before performing the Load. The default is not to store status before performing the Load.

Force indicator
An indicator used to request conditional processing of the Load command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Any number of arguments can be specified, meaning the **CMDINPUT.0** can be 0 - 6; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument such that the default will be used, the *TYPE* tail variable should be set to **HWMCA_TYPE_NULL** and the *DATA* tail variable should be set to the null string (for example, "").

The default for any argument can be overridden by specifying the *TYPE* and *DATA* tail variables as follows:

Load address

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.1.DATA
Should contain the address string to be used when performing the Load. This string must consist of 4 or less hexadecimal characters.

Load parameter

CMDINPUT.2.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.2.DATA
Should contain the parameter string to be used when performing the Load. This string must be less than or equal to 8 characters in length.

Clear indicator

CMDINPUT.3.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.3.DATA
Should contain the value HWMCA_TRUE for memory to be cleared before performing the Load or HWMCA_FALSE to bypass the clearing of memory before performing the Load.

Timeout

CMDINPUT.4.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.4.DATA
Should contain the timeout value that is to be used when performing the Load. This value must be 60 - 600 seconds.

Store status indicator

CMDINPUT.5.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.5.DATA
Should contain the value HWMCA_TRUE for status to be stored before performing the Load or HWMCA_FALSE to bypass the storing of status before performing the Load.

Force indicator

CMDINPUT.6.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.6.DATA
Should contain the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally.

HWMCA_HW_MESSAGE_REFRESH_COMMAND
No arguments are accepted or required.

HWMCA_HW_MESSAGE_DELETE_COMMAND
This command requires one argument, which is the time stamp of the hardware message. This implies that the **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_OCTETSTRING** and the **CMDINPUT.1.DATA** variable should contain the hardware message time stamp string itself.

HWMCA_ACTIVATE_CBU_COMMAND
This command requires one argument, which is an indicator of whether a real or test CBU activation should be performed. This implies that the **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain the value HWMCA_TRUE for a real CBU activation or HWMCA_FALSE for a test CBU activation. A second, optional, parameter for the password used to validate the CBU activation can be specified with a CMDINPUT.2.TYPE set to HWMCA_TYPE_OCTETSTRING and the CMDINPUT.2.DATA variable set to the desired password. If not specified, the password will be obtained automatically from the IBM support system.

HWMCA_UNDO_CBU_COMMAND
No arguments are accepted or required.

HWMCA_IMPORT_PROFILE_COMMAND
This command requires one argument, which is the profile area to be imported. This implies that the **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain an integer value greater than or equal to 1 or less than or equal to 4, indicating the profile area to be imported.

HWMCA_EXPORT_PROFILE_COMMAND
This command requires one argument, which is the profile area to be exported. This implies that the **CMDINPUT.0** variable should contain a 1. The

CMDINPUT.1.TYPE variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain an integer value greater than or equal to 1 or less than or equal to 4, indicating the profile area to be exported.

HWMCA_RESERVE_COMMAND

Note: This command is available only on a Support Element console.

This command requires two arguments. The first is the request/release indicator and the second is the name of the application requesting the reserve (exclusive control). This implies that the **CMDINPUT.0** variable should contain a 2. The first **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA** variable should contain the value HWMCA_TRUE when requesting the reserve or HWMCA_FALSE when releasing the reserve. The second **CMDINPUT.2.TYPE** variable should be set to **HWMCA_TYPE_OCTETSTRING** and the **CMDINPUT.2.DATA** variable should contain the application name. This length of the application name must be less than or equal to 8.

HWMCA_EXTERNAL_INTERRUPT_COMMAND

This command requires one argument, which is the number of the processor that is the target of the external interrupt command. This implies that the **CMDINPUT.0** variable should contain a 1. The **CMDINPUT.1.TYPE** variable should be set to **HWMCA_TYPE_INTEGER** and the **CMDINPUT.1.DATA**. variable should contain the processor number. This number is between 0 and the maximum number of processors for the target CPC image object.

HWMCA_SCSI_LOAD_COMMAND

No arguments are required, but the following arguments can optionally be specified:

Load address

Hexadecimal address to be used when performing the SCSI Load. The default will be to use the Load address last used when a SCSI Load was performed for the object.

Load parameter

Parameter string to be used when performing the SCSI Load. The default will be to use the Load parameter last used when a SCSI Load was performed for the object.

Worldwide port name

The worldwide port name (WWPN) to be used for the SCSI Load. The default will be to use the worldwide port name last used when a SCSI Load was performed for the object.

Logical unit number

The logical unit number (LUN) to be used for the SCSI Load. The default will be to use the logical unit number last used when a SCSI Load was performed for the object.

Boot Program Selector

The boot program selector to be used for the SCSI Load. The default will be to use the boot program selector last used when a SCSI Load was performed for the object.

Operating system specific load parameters

The operating system specific load parameters to be used for the SCSI Load. The default will be to use the operating system specific load parameters last used when a SCSI Load was performed for the object.

Boot record logical block address
The boot record logical block address to be used for the SCSI Load. The default will be to use the boot record logical block address last used when a SCSI Load was performed for the object.

Force indicator
An indicator used to request conditional processing of the SCSI Load command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Any number of arguments can be specified, meaning the CMDINPUT.0 can be 0 - 8; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument such that the default will be used, the **TYPE** tail variable should be set to **HWMCA_TYPE_NULL** and the **DATA** tail variable should be set to the null string (for example, ""). The default for any argument can be overridden by specifying the **TYPE** and **DATA** tail variables as follows:

Load address

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.1.DATA
Should contain the address string to be used when performing the SCSI Load. This string must consist of 4 or less hexadecimal characters.

Load parameter

CMDINPUT.2.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.2.DATA
Should contain the parameter string to be used when performing the SCSI Load. This string must have a length of eight characters or less.

Worldwide port name

CMDINPUT.3.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.3.DATA
Should contain the worldwide port name string to be used when performing the SCSI Load. This string must consist of 16 or less hexadecimal characters.

Logical unit number

CMDINPUT.4.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.4.DATA
Should contain the logical unit number string to be used when performing the SCSI Load. This string must consist of 16 or less hexadecimal characters.

Disk Partition Identifier

CMDINPUT.5.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.5.DATA
Should contain the boot program selector value, which can be in the range 0 - 30, inclusive.

Operating system specific load parameters

CMDINPUT.6.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.6.DATA
Should contain the operating system specific parameters string to be used when performing the SCSI Load. This string must be 256 characters or less.

Boot record logical block address

CMDINPUT.7.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.7.DATA
Should contain the boot record logical block address string to be used when performing the SCSI Load. This string must consist of 16 or less hexadecimal characters.

Force indicator

CMDINPUT.8.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.8.DATA
Should contain the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

HWMCA_SCSI_DUMP_COMMAND
No arguments are required, but the following arguments can optionally be specified:

Load address
Hexadecimal address to be used when performing the SCSI Dump. The default will be to use the Load address last used when a SCSI Dump was performed for the object.

Load parameter
Parameter string to be used when performing the SCSI Dump. The default will be to use the Load parameter last used when a SCSI Dump was performed for the object.

Worldwide port name
The worldwide port name (WWPN) to be used for the SCSI Dump. The default will be to use the worldwide port name last used when a SCSI Dump was performed for the object.

Logical unit number
The logical unit number (LUN) to be used for the SCSI Dump. The default will be to use the logical unit number last used when a SCSI Dump was performed for the object.

Boot Program Selector
The boot program selector to be used for the SCSI Dump. The default will be to use the boot program selector last used when a SCSI Dump was performed for the object.

Operating system specific load parameters
The operating system specific load parameters to be used for the SCSI Dump. The default will be to use the operating system specific load parameters last used when a SCSI Dump was performed for the object.

Boot record logical block address
The boot record logical block address to be used for the SCSI Dump. The default will be to use the boot record logical block address last used when a SCSI Dump was performed for the object.

Force indicator
An indicator used to request conditional processing of the SCSI Dump command depending on the state of the target object. The default is to unconditionally perform the command (that is, FORCE=TRUE) no matter what the state of the target object is.

Any number of arguments can be specified, meaning the CMDINPUT.0 can be 0 - 8; however, they must be specified in the order shown by the preceding list. If an argument is not specified, then the default for that argument is used. In order to specify an argument such that the default will be used, the **TYPE** tail variable should be set to **HWMCA_TYPE_NULL** and the **DATA** tail variable should be set to the null string (for example, ""). The default for any argument can be overridden by specifying the **TYPE** and **DATA** tail variables as follows:

Load address

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.1.DATA
Should contain the address string to be used when performing the SCSI Dump. This string must consist of 4 or less hexadecimal characters.

Load parameter

CMDINPUT.2.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.2.DATA
Should contain the parameter string to be used when performing the SCSI Dump. This string must have a length of 8 characters or less.

Worldwide port name

CMDINPUT.3.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.3.DATA
Should contain the worldwide port name string to be used when performing the SCSI Dump. This string must consist of 16 or less hexadecimal characters.

Logical unit number

CMDINPUT.4.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.4.DATA
Should contain the logical unit number string to be used when performing the SCSI Dump. This string must consist of 16 or less hexadecimal characters.

Disk Partition Identifier

CMDINPUT.5.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.5.DATA
Should contain the boot program selector value, which can be in the range 0 - 30, inclusive.

Operating system specific load parameters

CMDINPUT.6.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.6.DATA
Should contain the operating system specific parameters string to be used when performing the SCSI Dump. This string must be 256 characters or less.

Boot record logical block address

CMDINPUT.7.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.7.DATA
Should contain the boot record logical block address string to be used when performing the SCSI Dump. This string must consist of 16 or less hexadecimal characters.

Force indicator

CMDINPUT.8.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.8.DATA
Should contain the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

HWMCA_SHUTDOWN_RESTART_COMMAND
This command requires one argument, which is an indicator of the type of shutdown or restart to be performed. This implies that the CMDINPUT.0 variable should contain a 1. The CMDINPUT.1.TYPE variable should be set to HWMCA_TYPE_INTEGER and the CMDINPUT.1.DATA variable should contain one of the following values.

HWMCA_RESTART_APPLICATION
Used to indicate the Console application is to be restarted.

HWMCA_RESTART_CONSOLE
Used to indicate the Console is to be restarted.

HWMCA_SHUTDOWN_CONSOLE
Used to indicate the Console is to be shutdown/powered off.

HWMCA_RESTART_APPLICATION_ALTERNATE
Used to indicate the Alternate Support Element Console application is to be restarted. This option is only valid for the Support Element Console.

HWMCA_RESTART_CONSOLE_ALTERNATE
Used to indicate the Alternate Support Element Console is to be restarted. This option is only valid for the Support Element Console.

HWMCA_SHUTDOWN_CONSOLE_ALTERNATE
Used to indicate the Alternate Support Element Console is to be shutdown/powered off. This option is only valid for the Support Element Console.

HWMCA_ACTIVATE_OOCOD_COMMAND
This command requires one argument, which is the order number of the On/Off Capacity on Demand (On/Off CoD) record to be activated. This implies that the CMDINPUT.0 variable should contain a 1. The CMDINPUT.1.TYPE variable should be set to HWMCA_TYPE_OCTETSTRING and the CMDINPUT.1.DATA variable should contain the On/Off CoD order number to be activated.

HWMCA_UNDO_OOCOD_COMMAND
No arguments are accepted or required.

HWMCA_ADD_CAPACITY_COMMAND
This command requires one argument, which is an XML string describing the parameters to be used for capacity addition. This implies that the CMDINPUT.0 variable should contain a 1. The CMDINPUT.1.TYPE variable should be set to HWMCA_TYPE_OCTETSTRING and the CMDINPUT.1.DATA variable should contain XML string for these parameters.

Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

HWMCA_REMOVE_CAPACITY_COMMAND
This command requires one argument, which is an XML string describing the parameters to be used for capacity removal. This implies that the CMDINPUT.0 variable should contain a 1. The CMDINPUT.1.TYPE variable should be set to HWMCA_TYPE_OCTETSTRING and the CMDINPUT.1.DATA variable should contain XML string for these parameters.

Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

HWMCA_SYSPLEX_TIME_SWAP_CTS_COMMAND
This command requires the following argument:

STP ID

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING

CMDINPUT.1.DATA
Should contain a string representing the current STP identifier for the Defined CPC object.

HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND
This command requires the following arguments:

STP ID

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.1.DATA
Should contain a string representing the current STP identifier for the Defined CPC object.

Force Indicator

CMDINPUT.2.TYPE
Should be set to HWMCA_TYPE_INTEGER.

CMDINPUT.2.DATA
A pointer to a field containing the value HWMCA_TRUE for the command to be performed unconditionally or HWMCA_FALSE for the command to be performed conditionally based on the state of the target object.

STP Configuration XML

CMDINPUT.3.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING.

CMDINPUT.3.DATA
Should be an XML fragment describing the configuration for the STP-only CTN.

Note: Refer to Appendix F, "XML descriptions," on page 219 for a detailed description of this XML data.

HWMCA_SYSPLEX_TIME_CHANGE_STP_ONLY_CTN_ COMMAND
This command requires the following argument:

STP ID

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING

CMDINPUT.1.DATA
Should contain a string representing the desired STP identifier for the Defined CPC object and all CPCs that are members of the same STP-only CTN

HWMCA_SYSPLEX_TIME_JOIN_STP_ONLY_CTN_ COMMAND
This command requires the following argument:

STP ID

CMDINPUT.1.TYPE
Should be set to HWMCA_TYPE_OCTETSTRING

CMDINPUT.1.DATA
Should contain a string representing the desired STP identifier for the Defined CPC object.

HWMCA_SYSPLEX_TIME_LEAVE_STP_ONLY_CTN_ COMMAND
No arguments are accepted or required.

TIMEOUT
Used to specify the amount of time that the REXX application wants to wait for the *RxHwmcaCommand* to complete. This value is specified in milliseconds and the variable **HWMCA_INFINITE_WAIT** can be used to cause the application to wait forever.

Data exchange APIs (REXX sample)

This section shows an example REXX command file using the Console Data Exchange APIs and Commands API. The most up to date copy of this code is available on Resource Link at *http://www.ibm.com/servers/resourcelink*. Click **Services**, and then Click **API**.

```
/*********************************************************************/
/* Rexx command file used to illustrate the use of the Hardware     */
/* Management Console APIs.  This sample will allow the user to see */
/* the objects that can be managed from the Hardware Management     */
/* Console, as well as perform tasks against these objects.         */
/*********************************************************************/
trace 'o';

/*********************************************************************/
/* Number of seconds that this Rexx sample will wait for API calls  */
/* to complete.  This may need to be changed for remote networks    */
/* that require more time to return the responses.                  */
/*********************************************************************/
api_timeout_secs = 30;
api_timeout = api_timeout_secs * 1000;

/*********************************************************************/
/* Parse the provided arguments.  No arguments are required, since  */
/* we will prompt the user for them.  However, they can be passed   */
/* as follows:                                                       */
/*    Argument #1 - target HMC's hostname or internet address       */
/*    Argument #2 - target HMC's SNMP community name for API request*/
/*********************************************************************/
parse arg INITBLK.TARGET INITBLK.COMMUNITY .;
'@echo off'

error = 0;
/*********************************************************************/
/* Load the OS/2 Rexx Utility functions DLL.                         */
/*********************************************************************/
if RxFuncQuery('SysLoadFuncs') then do
   if rxfuncadd( 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs' ) then do
      say 'Error trying to add OS/2 Rexx utility functions.';
      error = 98;
   end  /* Do */
end  /* Do */
/*********************************************************************/
/* Load the Hwmca Rexx API interface function DLL.                   */
/*********************************************************************/
if RxFuncQuery('RxHwmcaLoadFuncs') then do
   if rxfuncadd( 'RxHwmcaLoadFuncs', 'ACTZSNMP', 'RxHwmcaLoadFuncs' ) then do
      say 'Error trying to add the Hardware Management Console API Rexx functions.';
      error = 99;
   end  /* Do */
end  /* Do */
if error == 0 then do
   call SysLoadFuncs               /* Load Rexx utility functions  */
   call RxHwmcaLoadFuncs;          /* Load HMC API functions       */
   call RxHwmcaDefineVars;         /* Define HMC API variables     */
   /*****************************************************************/
   /* Prompt the user for the HMC hostname or internet address.     */
   /*****************************************************************/
   if INITBLK.TARGET = '' then do
      say ' ';
      say 'Please enter target Hardware Management Console hostname or internet address.';
```

```
      pull INITBLK.TARGET .;
   end  /* Do */
   /*******************************************************************/
   /* Prompt the user for the HMC community name to use.              */
   /*******************************************************************/
   if INITBLK.COMMUNITY = '' then do
      say ' ';
      say 'Please enter community name for target Hardware Management Console.';
      parse pull INITBLK.COMMUNITY .;
   end  /* Do */
   /*******************************************************************/
   /* This sample uses the same Initialization block, INITBLK, for  */
   /* all RxHwmca API calls.                                          */
   /*******************************************************************/
   INITBLK.EVENTMASK = HWMCA_EVENT_COMMAND_RESPONSE;
   /*******************************************************************/
   /* Initialize ourselves with the HMC and tell it that we are only*/
   /* interested in command response events.                          */
   /*******************************************************************/
   rc = RxHwmcaInitialize('INITBLK.',api_timeout);
   if rc == HWMCA_DE_NO_ERROR then do
      /****************************************************************/
      /* Get the size of the screen so we know how much room we     */
      /* have for outputting information.                            */
      /****************************************************************/
      parse value SysTextScreenSize() with screen_rows screen_cols;
      /****************************************************************/
      /* We are now successfully initialized with the HMC.  First,  */
      /* lets get the name of the HMC.                               */
      /****************************************************************/
      call get_hmc_name;
      nest = 0;
      bailout = 0;
      /****************************************************************/
      /* Now we need to request the list of groups and present this */
      /* to the user.                                                */
      /****************************************************************/
      if result <> ''then call show_contents HWMCA_CONSOLE_ID 'Groups';
      /****************************************************************/
      /* Terminate our session with the HMC, so that it does not    */
      /* try and send us any more events.                            */
      /****************************************************************/
      rc = RxHwmcaTerminate('INITBLK.',api_timeout);
      call SysCls;
   end /* do */
   else do
      say 'Error' rc 'on RxHwmcaInitialize call.';
   end /* do */
end  /* Do */
exit

/*********************************************************************/
/* Subroutine: get_hmc_name                                          */
/*                                                                   */
/* This subroutine will request name attribute for the HMC.          */
/*********************************************************************/
get_hmc_name:

hmc_name = get_name(HWMCA_CONSOLE_ID);
return hmc_name;
```

```
/*********************************************************************/
/* Subroutine: show_contents                                         */
/*                                                                   */
/* This subroutine will get the contents attribute for the passed    */
/* in object and display the results to the user.  Note that         */
/* returned contents can be groups themselves or objects, such as    */
/* CPCs or CPC Images.                                               */
/*                                                                   */
/* Note: We expose a lot of the HMC API variables that we defined    */
/*       earlier by calling the RxHwmcaDefineVars function.          */
/*********************************************************************/
show_contents: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                                HWMCA_NAME_SUFFIX hmc_name screen_cols,
                                HWMCA_GROUP_CONTENTS_SUFFIX,
                                HWMCA_CONSOLE_ID nest bailout,
                                HWMCA_OBJECT_TYPE_SUFFIX,
                                HWMCA_STATUS_SUFFIX,
                                HWMCA_STATUS_ERROR_SUFFIX,
                                HWMCA_EXPECTED_STATUS_SUFFIX,
                                HWMCA_ACTIVATION_PROFILE_SUFFIX,
                                HWMCA_LAST_ACT_PROFILE_SUFFIX,
                                HWMCA_IP_ADDRESS_SUFFIX,
                                HWMCA_SNA_ADDRESS_SUFFIX,
                                HWMCA_MODEL_SUFFIX,
                                HWMCA_TYPE_SUFFIX,
                                HWMCA_MACHINE_SERIAL_SUFFIX,
                                HWMCA_CPC_SERIAL_SUFFIX,
                                HWMCA_CPC_ID_SUFFIX,
                                HWMCA_OPSYS_NAME_SUFFIX,
                                HWMCA_SYSPLEX_NAME_SUFFIX,
                                HWMCA_ACT_RESET_LIST_SUFFIX,
                                HWMCA_ACT_IMAGE_LIST_SUFFIX,
                                HWMCA_ACT_LOAD_LIST_SUFFIX,
                                HWMCA_ACT_PROFILE_IOCDS_SUFFIX,
                                HWMCA_ACT_PROFILE_IPLADDR_SUFFIX,
                                HWMCA_ACT_PROFILE_IPLPARM_SUFFIX,
                                HWMCA_CPC_OBJECT,
                                HWMCA_CPC_IMAGE_OBJECT,
                                HWMCA_CF_OBJECT,
                                HWMCA_INFINITE_WAIT,
                                HWMCA_TYPE_INTEGER,
                                HWMCA_TYPE_OCTETSTRING,
                                HWMCA_TRUE,
                                HWMCA_FALSE,
                                HWMCA_DE_TIMEOUT,
                                HWMCA_EVENT_COMMAND_RESPONSE,
                                HWMCA_ACTIVATE_COMMAND,
                                HWMCA_DEACTIVATE_COMMAND,
                                HWMCA_SEND_OPSYS_COMMAND,
                                HWMCA_RESETNORMAL_COMMAND,
                                HWMCA_RESETCLEAR_COMMAND,
                                HWMCA_START_COMMAND,
                                HWMCA_STOP_COMMAND,
                                HWMCA_LOAD_COMMAND,
                                HWMCA_PSWRESTART_COMMAND,
                                HWMCA_STATUS_OPERATING,
                                HWMCA_STATUS_NOT_OPERATING,
                                HWMCA_STATUS_NO_POWER,
                                HWMCA_STATUS_NOT_ACTIVATED,
                                HWMCA_STATUS_EXCEPTIONS,
                                HWMCA_STATUS_STATUS_CHECK,
                                HWMCA_STATUS_SERVICE,
                                HWMCA_STATUS_LINKNOTACTIVE,
                                HWMCA_STATUS_POWERSAVE;
```

```
parse arg object_id view_name;

/*********************************************************************/
/* Setup the string equivalents for the status values that the HMC  */
/* APIs will return.                                                 */
/*********************************************************************/
status. = 'Error getting the status attribute.';
status.HWMCA_STATUS_OPERATING     = 'Operating';
status.HWMCA_STATUS_NOT_OPERATING = 'Not operating';
status.HWMCA_STATUS_NO_POWER      = 'No power';
status.HWMCA_STATUS_NOT_ACTIVATED = 'Not activated';
status.HWMCA_STATUS_EXCEPTIONS    = 'Exceptions';
status.HWMCA_STATUS_STATUS_CHECK  = 'Status check';
status.HWMCA_STATUS_SERVICE       = 'Service';
status.HWMCA_STATUS_LINKNOTACTIVE = 'Communications not active';
status.HWMCA_STATUS_POWERSAVE     = 'Power save';
/*********************************************************************/
/* Setup the string equivalents for the status error values that    */
/* the HMC APIs will return.                                         */
/*********************************************************************/
status_error. = '';
status_error.HWMCA_TRUE  = 'Contains object(s) in unacceptable states';
status_error.HWMCA_FALSE = 'All objects in acceptable states';
errmsg = '';
nest = nest + 1;
rc = HWMCA_DE_NO_ERROR;
/*********************************************************************/
/* Loop until the user selects RETURN or EXIT, or until an error    */
/* occurred.  We will refresh the data each time the loop is taken. */
/*********************************************************************/
do while rc == HWMCA_DE_NO_ERROR & bailout == 0
   call SysCls;
   /*****************************************************************/
   /* Build the object identifier for the object contents.          */
   /*****************************************************************/
   rc = RxHwmcaBuildAttributeId('ATTRID',object_id,HWMCA_GROUP_CONTENTS_SUFFIX);
   if rc <> HWMCA_DE_NO_ERROR then do
      rc = RxHwmcaBuildId('ATTRID',object_id,HWMCA_GROUP_CONTENTS_SUFFIX);
   end /* do */
   if rc == HWMCA_DE_NO_ERROR then do
      /**************************************************************/
      /* Get the object contents.                                   */
      /**************************************************************/
      rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
      if rc == HWMCA_DE_NO_ERROR then do
         objects.id. = '';   objects.name. = '';
         objects.type. = ''; objects.status = '';
         objects.0 = 0;
         cpcs_found = 0;
         /***********************************************************/
         /* Loop through all of the objects returned and get the    */
         /* name, type, and status or status error attributes. This */
         /* is the information that we will present to the user.    */
         /***********************************************************/
         do i = 1 to words(OUTPUT.1.DATA) while rc == 0;
            objects.0 = objects.0 + 1;
            objects.id.i = word(OUTPUT.1.DATA,i);
            /********************************************************/
            /* Get the object's name attribute.                     */
            /********************************************************/
```

```
   objects.name.i = get_name(objects.id.i);
   if objects.name.i == '' then do
      say 'Error getting the object name attribute.';
   end /* do */
   /*********************************************************/
   /* Get the object's type attribute.                      */
   /*********************************************************/
   objects.type.i = get_type(objects.id.i);
   select
      when objects.type.i == '' then do
         say 'Error getting the object type attribute.';
         get_status = 0;
      end /* do */
      when objects.type.i == HWMCA_CPC_OBJECT then do
         cpcs_found = 1;
         get_status = 1;
      end  /* Do */
      when objects.type.i == HWMCA_CF_OBJECT |,
           objects.type.i == HWMCA_CPC_IMAGE_OBJECT then do
         objects.name.i = translate(objects.name.i,':',' ');
         get_status = 1;
      end /* do */
      otherwise get_status = 2;
   end  /* select */
   if get_status == 2 then do
      /*****************************************************/
      /* Get the object's status error attribute.          */
      /*****************************************************/
      x = get_status_error(objects.id.i);
      objects.status.i = status_error.x;
   end  /* Do */
   else do
      if get_status == 1 then do
         /**************************************************/
         /* Get the object's status attribute.             */
         /**************************************************/
         x = get_status(objects.id.i);
         objects.status.i = status.x;
      end  /* Do */
   end  /* Do */
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*********************************************************/
   /* Everything is still ok, so lets build the screen for */
   /* the user.First, lets display the title and some text */
   /* that tells the user what to do.                       */
   /*********************************************************/
   say center(hmc_name||' - '||view_name||' Work Area',screen_cols);
   say 'Please type a number to select a target and press the specified';
   say 'function key to perform the desired task.';
   say;
   /*********************************************************/
   /* Now lets display each object with its name and       */
   /* status values.                                        */
   /*********************************************************/
   do i = 1 to objects.0 while rc == 0;
      say right(i,3)||'. '||left(objects.name.i,17)||' - '||objects.status.i;
   end /* do */
   say;
```

```
/*****************************************************/
/* Let's determine which set of functions key        */
/* definitions to use depending on whether or not we */
/* are looking at groups, CPCs, or CPC Images.       */
/*****************************************************/
profile. = '';
command. = '';
command.2  = 'REFRESH';
command.3  = 'EXIT';
command.12 = 'RETURN';
if nest > 1 then do
   /**************************************************/
   /* We are looking at "real" objects, not just a list */
   /* of group objects.  The Activate and Deactivate */
   /* tasks are valid for all types of "real" objects.  */
   /**************************************************/
   command.5 = HWMCA_ACTIVATE_COMMAND;
   command.6 = HWMCA_DEACTIVATE_COMMAND;
   command.7  = 'DETAILS';
   if cpcs_found == 1 then do
      /*************************************************/
      /* It is a list of CPC objects, so the only tasks */
      /* the user can do is Activate, Deactivate and    */
      /* Details.                                      */
      /*************************************************/
      say 'F1=        F2=Refresh    F3=Exit       F4 =          F5 =Activate F6=Deactivate';
      say 'F7=Details F8=Reset Prof F9=Image Prof F10=Load Prof F11=         F12=Return';
      command.8  = 'PROFILES';
      command.9  = 'PROFILES';
      command.10 = 'PROFILES';
      profile.8  = 'RESET';
      profile.9  = 'IMAGE';
      profile.10 = 'LOAD';
   end  /* Do */
   else do
      /*************************************************/
      /* It is a list of CPC Image and/or CF objects,   */
      /* so let's allow the user to do almost anything. */
      /*************************************************/
      say 'F1=Load     F2=Refresh  F3=Exit   F4 =OpSys Cmd F5 =Activate     F6 =Deactivate';
      say 'F7=Details  F8=Reset    F9=Start  F10=Stop      F11=PSW Restart  F12=Return';
      command.1  = HWMCA_LOAD_COMMAND;
      command.4  = HWMCA_SEND_OPSYS_COMMAND;
      command.8  = HWMCA_RESETNORMAL_COMMAND;
      command.9  = HWMCA_START_COMMAND;
      command.10 = HWMCA_STOP_COMMAND;
      command.11 = HWMCA_PSWRESTART_COMMAND;
   end  /* Do */
end /* do */
else do
   /**************************************************/
   /* We are looking at a list of groups, so we will    */
   /* only let the user open the group to see its       */
   /* contents.                                      */
   /**************************************************/
   command.7  = 'OPEN';
   say    'F1=         F2=Refresh  F3=Exit   F4 =          F5 =             F6 =';
   say    'F7=Open     F8=         F9=       F10=          F11=             F12=Return';
end /* do */
```

```
/*******************************************************/
/* If there is an error message, then display it and   */
/* beep!                                               */
/*******************************************************/
say errmsg;
if errmsg <> '' then do
   errmsg = '';
   call beep 523,250;
end  /* Do */
call charout , '====> ';
/*******************************************************/
/* Allow the user to type a number(s) to select the    */
/* target for the request and check for function keys  */
/* to see what task to perform.                        */
/*******************************************************/
fkey = ''; request = '';
do while fkey == ''
   key = SysGetKey('NOECHO');
   keynum = c2d(key);
   select
      when keynum == 0 then do          /* Function key */
         key = SysGetKey('NOECHO');
         keynum = c2d(key);
         if keynum >= 133 then fkey = keynum-122;
         else fkey = keynum-58;
      end  /* Do */
      when keynum == 13 then fkey = 2;  /* Enter key    */
      when keynum == 27 then fkey = 3;  /* Esc key      */
      otherwise do
         request = request||key;
         call charout ,key;
      end /* do */
   end  /* select */
end /* do */
/*******************************************************/
/* One of the functions keys or Enter has been pressed */
/* by the user, so let figure out what to do.          */
/*******************************************************/
select
   when command.fkey == 'REFRESH' then nop;  /* Refresh */
   when command.fkey == 'EXIT' then do       /* Exit    */
      bailout = 1;
      leave;
   end  /* Do */
   when command.fkey == 'RETURN' then leave; /* Return  */
   /*****************************************************/
   /* The user select to open the contents for a group. */
   /* Check to make sure that they entered the number of*/
   /* the group to open and then call show_contents to  */
   /* display the contents of the group.                */
   /*****************************************************/
   when command.fkey == 'OPEN' then do
      if datatype(request) == 'NUM' then do
         if request >= 1 & request <= objects.0 then do
            call show_contents objects.id.request,
                               objects.name.request;
         end  /* Do */
         else do
            errmsg = 'Input number is out of range.';
         end  /* Do */
      end  /* Do */
```

```
   else do
      errmsg = 'Input is not a valid number.';
   end  /* Do */
end  /* Do */
/*****************************************************/
/* The user select to display the details for an     */
/* object.  Check to make sure that they entered the */
/* number of the object and then call show_details   */
/* to display the details for the object.            */
/*****************************************************/
when command.fkey == 'DETAILS' then do
   if datatype(request) == 'NUM' then do
      if request >= 1 & request <= objects.0 then do
         call show_details objects.id.request,
                           objects.type.request,
                           objects.name.request,
                           objects.status.request,
                           '('||view_name||')';
      end  /* Do */
      else do
         errmsg = 'Input number is out of range.';
      end  /* Do */
   end  /* Do */
   else do
      errmsg = 'Input is not a valid number.';
   end  /* Do */
end  /* Do */
/*****************************************************/
/* The user select to display a list of activation   */
/* profiles.  Check to make sure that they entered   */
/* the number of the object and then call            */
/* show_profiles to see the list of profiles.        */
/*****************************************************/
when command.fkey == 'PROFILES' then do
   if datatype(request) == 'NUM' then do
      if request >= 1 & request <= objects.0 then do
         if profile.fkey <> '' then do
            call show_profiles profile.fkey,
                               objects.id.request,
                               objects.type.request,
                               objects.name.request,
                               objects.status.request,
                               '('||view_name||')';
         end /* do */
         else do
            errmsg = 'Invalid profile type specified.';
         end /* do */
      end  /* Do */
      else do
         errmsg = 'Input number is out of range.';
      end  /* Do */
   end  /* Do */
   else do
      errmsg = 'Input is not a valid number.';
   end  /* Do */
end  /* Do */
/*****************************************************/
/* Make sure the user pressed a defined function key.*/
/* If they did, then after making sure they entered  */
/* a valid number for the target object, call        */
/* perform_command to execute the specified task.    */
/*****************************************************/
```

```
               otherwise do
                  if command.fkey <> '' then do
                     if datatype(request) == 'NUM' then do
                        if request >= 1 & request <= objects.0 then do
                           call perform_command command.fkey,
                                                objects.id.request,
                                                objects.type.request,
                                                objects.name.request;
                           errmsg = result;
                        end  /* Do */
                        else do
                           errmsg = 'Input number is out of range.';
                        end  /* Do */
                     end  /* Do */
                     else do
                        errmsg = 'Input is not a valid number.';
                     end  /* Do */
                  end  /* Do */
                  else do
                     errmsg = 'Undefined function key pressed.';
                  end  /* Do */
               end  /* Do */
            end  /* select */
         end /* do */
      end /* do */
      else do
         say 'Error' rc 'on RxHwmcaGet for the object contents attribute.';
      end /* do */
   end /* do */
   else do
      say 'Error' rc 'on RxHwmcaBuildId for the object contents attribute.';
   end /* do */
end /* do */
nest = nest - 1;

return rc;
/*********************************************************************/
/* Subroutine: show_profiles                                         */
/*                                                                   */
/* This subroutine will request display a list of activation         */
/* profiles for a CPC object.                                        */
/*                                                                   */
/* Note: We expose a lot of the HMC API variables that we defined    */
/*       earlier by calling the RxHwmcaDefineVars function.          */
/*********************************************************************/
show_profiles: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                                hmc_name screen_cols bailout,
                                HWMCA_NAME_SUFFIX,
                                HWMCA_ACT_RESET_LIST_SUFFIX,
                                HWMCA_ACT_IMAGE_LIST_SUFFIX,
                                HWMCA_ACT_LOAD_LIST_SUFFIX,
                                HWMCA_ACT_PROFILE_IOCDS_SUFFIX,
                                HWMCA_ACT_PROFILE_IPLADDR_SUFFIX,
                                HWMCA_ACT_PROFILE_IPLPARM_SUFFIX,

parse arg proftype object_id object_type object_name object_status '(' group ')' .;

/*********************************************************************/
/* Setup an array of the suffix values for the attributes that we    */
/* need to get for the details display.  Note that this array is     */
/* different for CPCs and CPC Image/CF objects.                      */
/*********************************************************************/
```

```
profiletype.='';
profiletype.1 = 'Reset';
profiletype.2 = 'Image';
profiletype.3 = 'Load';
suffix.='';
suffix.0   = 3;
suffix.1   = HWMCA_ACT_RESET_LIST_SUFFIX;
suffix.1.0 = 2;
suffix.1.1 = HWMCA_NAME_SUFFIX;
suffix.1.2 = HWMCA_ACT_PROFILE_IOCDS_SUFFIX;
suffix.2   = HWMCA_ACT_IMAGE_LIST_SUFFIX;
suffix.2.0 = 3;
suffix.2.1 = HWMCA_NAME_SUFFIX;
suffix.2.2 = HWMCA_ACT_PROFILE_IPLADDR_SUFFIX;
suffix.2.3 = HWMCA_ACT_PROFILE_IPLPARM_SUFFIX;
suffix.3   = HWMCA_ACT_LOAD_LIST_SUFFIX;
suffix.3.0 = 3;
suffix.3.1 = HWMCA_NAME_SUFFIX;
suffix.3.2 = HWMCA_ACT_PROFILE_IPLADDR_SUFFIX;
suffix.3.3 = HWMCA_ACT_PROFILE_IPLPARM_SUFFIX;
rc = HWMCA_DE_NO_ERROR;
/*********************************************************************/
/* Loop until the user selects RETURN or EXIT, or until an error     */
/* occurred.  We will refresh the data each time the loop is taken. */
/*********************************************************************/
do while rc == HWMCA_DE_NO_ERROR & bailout == 0
   call SysCls;
   list. = '';
   field. = '';
   select
      when proftype == 'RESET' then i = 1;
      when proftype == 'IMAGE' then i = 2;
      when proftype == 'LOAD' then i = 3;
      otherwise i = 0;
   end  /* select */
   if i <> 0 then do
      /*****************************************************************/
      /* Call get_attribute to get each of the attributes need for the */
      /* profile list display.                                         */
      /*****************************************************************/
      /* do i = 1 to suffix.0; */
         list.i = get_attribute(object_id suffix.i);
         do j = 1 to words(list.i);
            profile_id = word(list.i,j);
            do k = 1 to suffix.i.0
               field.i.j.k = get_attribute(profile_id suffix.i.k);
            end /* do */
         end /* do */
      /* end */ /* do */
      /*****************************************************************/
      /* Display the details title.                                    */
      /*****************************************************************/
      say center(hmc_name||' - '||object_name||' '||profiletype.i||' Profile List',screen_cols);
      /*****************************************************************/
      /* Now build the lines of activation profile information that    */
      /* are to be displayed.                                          */
      /*****************************************************************/
      /* do i = 1 to suffix.0; */
         do j = 1 to words(list.i);
            select
               when profiletype.i == 'Reset' then do
```

```
            if field.i.j.2 == '' then field.i.j.2 = 'Use Active IOCDS';
            say left(field.i.j.1,16)||,
                ' - IOCDS: '||field.i.j.2
         end /* do */
         when (profiletype.i == 'Image') | (profiletype.i == 'Load') then do
            if field.i.j.2 == '' then field.i.j.2 = 'Dynamic';
            else field.i.j.2 = right(field.i.j.2,4,'0')||'   ';
            if field.i.j.3 == '' then field.i.j.3 = 'Dynamic';
            say left(field.i.j.1,16)||,
                ' - Load address: '||field.i.j.2||,
                ' Load parameter: ['||field.i.j.3||']';
         end /* do */
         otherwise nop;
      end  /* select */
   end /* do */
/* end */ /* do */
/*********************************************************************/
/* Display the valid function keys...nothing much allowed here   */
/* except for return/exit or refresh.                            */
/*********************************************************************/
say;
say    'F1=         F2=Refresh  F3=Exit   F4 =          F5 =             F6 =';
say    'F7=         F8=         F9=       F10=          F11=             F12=Return';
say;
command. = '';
command.2  = 'REFRESH';
command.3  = 'EXIT';
command.12 = 'RETURN';
call charout , '====> ';
/*********************************************************************/
/* Allow the user to press Enter or a function key.              */
/*********************************************************************/
fkey = ''; request = '';
do while fkey == ''
   key = SysGetKey('NOECHO');
   keynum = c2d(key);
   select
      when keynum == 0 then do          /* Function key */
         key = SysGetKey('NOECHO');
         keynum = c2d(key);
         if keynum >= 133 then fkey = keynum-122;
         else fkey = keynum-58;
      end  /* Do */
      when keynum == 13 then fkey = 2;  /* Enter key    */
      when keynum == 27 then fkey = 3;  /* Esc key      */
      otherwise do
         request = request||key;
         call charout ,key;
      end /* do */
   end  /* select */
end /* do */
/*********************************************************************/
/* One of the functions keys or Enter has been pressed  by the   */
/* user, so let figure out what to do.                           */
/*********************************************************************/
select
   when command.fkey == 'REFRESH' then do      /* Refresh  */
      nop;
   end  /* Do */
   when command.fkey == 'EXIT' then do         /* Exit     */
      bailout = 1;
      leave;
```

```
         end  /* Do */
         when command.fkey == 'RETURN' then leave;   /* Return   */
         otherwise nop;
      end  /* select */
   end /* do */
   else do
      leave;
   end /* do */
end  /* Do */

return rc;

/*********************************************************************/
/* Subroutine: show_details                                          */
/*                                                                   */
/* This subroutine will request display the details for a CPC        */
/* object, CPC Image object, or a CF object.                         */
/*                                                                   */
/* Note: We expose a lot of the HMC API variables that we defined    */
/*       earlier by calling the RxHwmcaDefineVars function.          */
/*********************************************************************/
show_details: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                               hmc_name screen_cols bailout,
                               status.,
                               HWMCA_CPC_OBJECT,
                               HWMCA_CPC_IMAGE_OBJECT,
                               HWMCA_CF_OBJECT,
                               HWMCA_EXPECTED_STATUS_SUFFIX,
                               HWMCA_ACTIVATION_PROFILE_SUFFIX,
                               HWMCA_LAST_ACT_PROFILE_SUFFIX,
                               HWMCA_IP_ADDRESS_SUFFIX,
                               HWMCA_SNA_ADDRESS_SUFFIX,
                               HWMCA_STATUS_SUFFIX,
                               HWMCA_MODEL_SUFFIX,
                               HWMCA_TYPE_SUFFIX,
                               HWMCA_MACHINE_SERIAL_SUFFIX,
                               HWMCA_CPC_SERIAL_SUFFIX,
                               HWMCA_CPC_ID_SUFFIX,
                               HWMCA_OPSYS_NAME_SUFFIX,
                               HWMCA_SYSPLEX_NAME_SUFFIX,
                               HWMCA_STATUS_OPERATING,
                               HWMCA_STATUS_NOT_OPERATING,
                               HWMCA_STATUS_NO_POWER,
                               HWMCA_STATUS_NOT_ACTIVATED,
                               HWMCA_STATUS_EXCEPTIONS,
                               HWMCA_STATUS_STATUS_CHECK,
                               HWMCA_STATUS_SERVICE,
                               HWMCA_STATUS_LINKNOTACTIVE,
                               HWMCA_STATUS_POWERSAVE;

parse arg object_id object_type object_name object_status '(' group ')' .;

/*********************************************************************/
/* Setup an array of the suffix values for the attributes that we    */
/* need to get for the details display.  Note that this array is     */
/* different for CPCs and CPC Image/CF objects.                      */
/*********************************************************************/
suffix.='';
suffix.1 = HWMCA_EXPECTED_STATUS_SUFFIX;
suffix.2 = HWMCA_ACTIVATION_PROFILE_SUFFIX;
suffix.3 = HWMCA_LAST_ACT_PROFILE_SUFFIX;
```

```
if object_type == HWMCA_CPC_OBJECT then do
   suffix.4  = HWMCA_IP_ADDRESS_SUFFIX;
   suffix.5  = HWMCA_SNA_ADDRESS_SUFFIX;
   suffix.6  = HWMCA_MODEL_SUFFIX;
   suffix.7  = HWMCA_TYPE_SUFFIX;
   suffix.8  = HWMCA_MACHINE_SERIAL_SUFFIX;
   suffix.9  = HWMCA_CPC_SERIAL_SUFFIX;
   suffix.10 = HWMCA_CPC_ID_SUFFIX;
   suffix.0 = 10;
end  /* Do */
else do
   suffix.4 = HWMCA_OPSYS_NAME_SUFFIX;
   suffix.5 = HWMCA_SYSPLEX_NAME_SUFFIX;
   suffix.0 = 5;
end  /* Do */
rc = HWMCA_DE_NO_ERROR;
box_width = (screen_cols-6);
/*********************************************************************/
/* Loop until the user selects RETURN or EXIT, or until an error     */
/* occurred.  We will refresh the data each time the loop is taken. */
/*********************************************************************/
do while rc == HWMCA_DE_NO_ERROR & bailout == 0
   call SysCls;
   field. = '';
   /******************************************************************/
   /* Call get_attribute to get each of the attributes need for the */
   /* details display.                                               */
   /******************************************************************/
   do i = 1 to suffix.0
      field.i = get_attribute(object_id suffix.i);
   end /* do */
   /******************************************************************/
   /* Display the details title.                                     */
   /******************************************************************/
   say center(hmc_name||' - '||object_name||' Details',screen_cols);
   /******************************************************************/
   /* Display the instance information set of attributes, note that */
   /* this set is different for CPCs and CPC Image/CF objects.      */
   /******************************************************************/
   say left(' ┌─Instance information',screen_cols-3,'─')||'┐ ';
   if object_type == HWMCA_CPC_OBJECT then do
      say left(' │ Status: 'left(object_status,(box_width/2)-8)||,
               ' Activation profile: 'field.2,screen_cols-3)||'│ ';
      say left(' │ Group:  'left(group,(box_width/2)-8)||,
               ' Last used profile:  'field.3,screen_cols-3)||'│ ';
   end  /* Do */
   else do
      say left(' │ Status:      ' left(object_status,(box_width/2)-15)||,
               ' Activation profile: 'field.2,screen_cols-3)||'│ ';
      say left(' │ Group:       ' left(group,(box_width/2)-15)||,
               ' Last used profile:  'field.3,screen_cols-3)||'│ ';
      say left(' │ SysPlex name:' left(field.5,(box_width/2)-15)||,
               ' Operating System:   'field.4,screen_cols-3)||'│ ';
   end  /* Do */
   say left(' └',screen_cols-3,'─')||'┘';
   /******************************************************************/
   /* Display the acceptable status settings.  Note the set of      */
   /* acceptable status values is different for CPCs and CPC Image  */
   /* or CF objects.                                                 */
   /******************************************************************/
```

```
say left(' ┌─Acceptable status',screen_cols-3,'─')||'┐ ';
accstatus. = ' ';
accstatus.0 = 9;
accstatus.1.value = x2b(right(d2x(HWMCA_STATUS_OPERATING),3,'0'));
accstatus.2.value = x2b(right(d2x(HWMCA_STATUS_NOT_OPERATING),3,'0'));
accstatus.3.value = x2b(right(d2x(HWMCA_STATUS_NO_POWER),3,'0'));
accstatus.4.value = x2b(right(d2x(HWMCA_STATUS_NOT_ACTIVATED),3,'0'));
accstatus.5.value = x2b(right(d2x(HWMCA_STATUS_EXCEPTIONS),3,'0'));
accstatus.6.value = x2b(right(d2x(HWMCA_STATUS_STATUS_CHECK),3,'0'));
accstatus.7.value = x2b(right(d2x(HWMCA_STATUS_SERVICE),3,'0'));
accstatus.8.value = x2b(right(d2x(HWMCA_STATUS_LINKNOTACTIVE),3,'0'));
accstatus.9.value = x2b(right(d2x(HWMCA_STATUS_POWERSAVE),3,'0'));
field.1 = x2b(right(d2x(field.1),3,'0'));
do i = 1 to accstatus.0
   if (bitand(field.1,accstatus.i.value) == accstatus.i.value) then do
      accstatus.i.check = 'x';
   end  /* Do */
end /* do */
if object_type == HWMCA_CPC_OBJECT then do
   say left(' │ 'left(accstatus.1.check' Operating',box_width/2)||,
       left(accstatus.9.check' Power save',box_width/2),screen_cols-3)||'│ ';
   say left(' │ 'left(accstatus.2.check' Not operating',box_width/2)||,
       left(accstatus.5.check' Exceptions',box_width/2),screen_cols-3)||'│ ';
   say left(' │ 'left(accstatus.3.check' No power',box_width/2)||,
       left(accstatus.6.check' Status check',box_width/2),screen_cols-3)||'│ ';
   say left(' │ 'left(accstatus.8.check' Communications not active',box_width/2)||,
       left(accstatus.7.check' Service',box_width/2),screen_cols-3)||'│ ';
end  /* Do */
else do
   say left(' │ 'left(accstatus.1.check' Operating',box_width/2)||,
       left(accstatus.9.check' Power save',box_width/2),screen_cols-3)||'│ ';
   say left(' │ 'left(accstatus.4.check' Not activated',box_width/2)||,
       left(accstatus.5.check' Exceptions',box_width/2),screen_cols-3)||'│ ';
   say left(' │ 'left(accstatus.2.check' Not operating',box_width/2)||,
       left(accstatus.6.check' Status check',box_width/2),screen_cols-3)||'│ ';
end  /* Do */
say left(' └',screen_cols-3,'─')||'┘';
/*****************************************************************/
/* If it is a CPC object, then show the product information set  */
/* of attributes.                                                */
/*****************************************************************/
if object_type == HWMCA_CPC_OBJECT then do
   say left(' ┌─Product information',screen_cols-3,'─')||'┐' ';
   type_model = right(field.7,6,'0')||' - '||field.6;
   say left(' │ Machine type - model: 'left(type_model,(box_width/2)-23)||,
            ' SNA address:      'field.5,screen_cols-3)||'│ ';
   ipaddr = field.4;
   ip.4 = ipaddr // 256;
   ipaddr = ipaddr % 256;
   ip.3 = ipaddr // 256;
   ipaddr = ipaddr % 256;
   ip.2 = ipaddr // 256;
   ipaddr = ipaddr % 256;
   ip.1 = ipaddr;
   ipaddr = ip.1'.'ip.2'.'ip.3'.'ip.4;
   mach_serial = substr(field.8,4,2)||' - '||substr(field.8,6);
   say left(' │ Machine serial:       'left(mach_serial,(box_width/2)-23)||,
            ' Internet address: 'ipaddr,screen_cols-3)||'│ ';
   mach_seq = right(substr(field.8,6),12,'0');
   plant = substr(field.8,4,2);
   say left(' │ Machine sequence:     'left(mach_seq,(box_width/2)-23)||,
            ' Plant of man.:    'plant,screen_cols-3)||'│ ';
```

```
      cpc_id = right(field.10,2,'0');
      say left(' │ CPC serial:           'left(field.9,(box_width/2)-23)||,
               ' CPC identifier:   'cpc_id,screen_cols-3)||'│ ';
      say left(' └'',screen_cols-3,'─')┘'||'┘';
   end  /* Do */
   /*****************************************************************/
   /* Display the valid function keys...nothing much allowed here   */
   /* except for return/exit or refresh.                            */
   /*****************************************************************/
   say;
   say    'F1=         F2=Refresh  F3=Exit   F4 =          F5 =             F6 =';
   say    'F7=         F8=         F9=       F10=          F11=             F12=Return';
   say;
   command. = '';
   command.2  = 'REFRESH';
   command.3  = 'EXIT';
   command.12 = 'RETURN';
   call charout , '====> ';
   /*****************************************************************/
   /* Allow the user to press Enter or a function key.              */
   /*****************************************************************/
   fkey = ''; request = '';
   do while fkey == ''
      key = SysGetKey('NOECHO');
      keynum = c2d(key);
      select
         when keynum == 0 then do          /* Function key */
            key = SysGetKey('NOECHO');
            keynum = c2d(key);
            if keynum >= 133 then fkey = keynum-122;
            else fkey = keynum-58;
         end  /* Do */
         when keynum == 13 then fkey = 2;  /* Enter key    */
         when keynum == 27 then fkey = 3;  /* Esc key      */
         otherwise do
            request = request||key;
            call charout ,key;
         end /* do */
      end  /* select */
   end /* do */
   /*****************************************************************/
   /* One of the functions keys or Enter has been pressed by the    */
   /* user, so let figure out what to do.                           */
   /*****************************************************************/
   select
      when command.fkey == 'REFRESH' then do      /* Refresh  */
         /*************************************************************/
         /* Refresh the object's status attribute value.              */
         /*************************************************************/
         x = get_status(object_id);
         object_status = status.x;
      end  /* Do */
      when command.fkey == 'EXIT' then do         /* Exit     */
         bailout = 1;
         leave;
      end  /* Do */
      when command.fkey == 'RETURN' then leave;   /* Return   */
      otherwise nop;
   end  /* select */
end  /* Do */

return rc;
```

```
/*********************************************************************/
/* Subroutine: get_name                                              */
/*                                                                   */
/* This subroutine will perform a Get request for the name attribute*/
/* for the specified object.                                         */
/*********************************************************************/
get_name: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                           HWMCA_NAME_SUFFIX;
parse arg object_id .;

name = '';
/*********************************************************************/
/* Build the object identifier for the object name attribute.       */
/*********************************************************************/
rc = RxHwmcaBuildAttributeId('ATTRID',object_id,HWMCA_NAME_SUFFIX);
if rc <> HWMCA_DE_NO_ERROR then do
   rc = RxHwmcaBuildId('ATTRID',object_id,HWMCA_NAME_SUFFIX);
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*****************************************************************/
   /* Get the object name attribute.                                */
   /*****************************************************************/
   rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
   if rc == HWMCA_DE_NO_ERROR then do
      /**************************************************************/
      /* Remove any newline characters from the name.               */
      /**************************************************************/
      name = translate(OUTPUT.1.DATA,' ','0A'x);
   end /* do */
   else do
      say 'Error' rc 'on RxHwmcaGet for name attribute.';
   end /* do */
end /* do */
else do
   say 'Error' rc 'on RxHwmcaBuildId for name attribute.';
end /* do */

return name;

/*********************************************************************/
/* Subroutine: get_status                                            */
/*                                                                   */
/* This subroutine will perform a Get request for the status         */
/* attribute for the specified object.                               */
/*********************************************************************/
get_status: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                             HWMCA_STATUS_SUFFIX;
parse arg object_id .;

status = '';
/*********************************************************************/
/* Build the object identifier for the object status attribute.      */
/*********************************************************************/
rc = RxHwmcaBuildAttributeId('ATTRID',object_id,HWMCA_STATUS_SUFFIX);
if rc <> HWMCA_DE_NO_ERROR then do
   rc = RxHwmcaBuildId('ATTRID',object_id,HWMCA_STATUS_SUFFIX);
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*****************************************************************/
   /* Get the object status attribute.                              */
   /*****************************************************************/
```

```
   rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
   if rc == HWMCA_DE_NO_ERROR then do
      status = OUTPUT.1.DATA;
   end  /* Do */
   else do
      say 'Error' rc 'on RxHwmcaGet for status attribute.';
   end /* do */
end /* do */
else do
   say 'Error' rc 'on RxHwmcaBuildId for status attribute.';
end /* do */

return status;

/*********************************************************************/
/* Subroutine: get_status_error                                      */
/*                                                                   */
/* This subroutine will perform a Get request for the status error  */
/* attribute for the specified object.                               */
/*********************************************************************/
get_status_error: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                                   HWMCA_STATUS_ERROR_SUFFIX;
parse arg object_id .;

status_error = '';
/*********************************************************************/
/*Build the object identifier for the object status error attribute.*/
/*********************************************************************/
rc = RxHwmcaBuildAttributeId('ATTRID',object_id,HWMCA_STATUS_ERROR_SUFFIX);
if rc <> HWMCA_DE_NO_ERROR then do
   rc = RxHwmcaBuildId('ATTRID',object_id,HWMCA_STATUS_ERROR_SUFFIX);
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*******************************************************************/
   /* Get the object status error attribute.                          */
   /*******************************************************************/
   rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
   if rc == HWMCA_DE_NO_ERROR then do
      status_error = OUTPUT.1.DATA;
   end  /* Do */
   else do
      say 'Error' rc 'on RxHwmcaGet for status error attribute.';
   end /* do */
end /* do */
else do
   say 'Error' rc 'on RxHwmcaBuildId for status error attribute.';
end /* do */

return status_error;

/*********************************************************************/
/* Subroutine: get_type                                              */
/*                                                                   */
/* This subroutine will perform a Get request for the type attribute*/
/* for the specified object.                                         */
/*********************************************************************/
get_type: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                           HWMCA_OBJECT_TYPE_SUFFIX;
parse arg object_id .;

type = '';
```

```
/*********************************************************************/
/* Build the object identifier for the object type attribute.       */
/*********************************************************************/
rc = RxHwmcaBuildAttributeId('ATTRID',object_id,HWMCA_OBJECT_TYPE_SUFFIX);
if rc <> HWMCA_DE_NO_ERROR then do
   rc = RxHwmcaBuildId('ATTRID',object_id,HWMCA_OBJECT_TYPE_SUFFIX);
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*****************************************************************/
   /* Get the object type attribute.                                */
   /*****************************************************************/
   rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
   if rc == HWMCA_DE_NO_ERROR then do
      type = OUTPUT.1.DATA;
   end /* do */
   else do
      say 'Error' rc 'on RxHwmcaGet for type attribute.';
   end /* do */
end /* do */
else do
   say 'Error' rc 'on RxHwmcaBuildId for type attribute.';
end /* do */

return type;

/*********************************************************************/
/* Subroutine: get_profile                                           */
/*                                                                   */
/* This subroutine will perform a Get request for the activation     */
/* profile attribute for the specified object.                       */
/*********************************************************************/
get_profile: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                              HWMCA_ACTIVATION_PROFILE_SUFFIX;
parse arg object_id .;

profile = '';
/*********************************************************************/
/* Build the object identifier for the object type attribute.       */
/*********************************************************************/
rc = RxHwmcaBuildAttributeId('ATTRID',object_id,HWMCA_ACTIVATION_PROFILE_SUFFIX);
if rc <> HWMCA_DE_NO_ERROR then do
   rc = RxHwmcaBuildId('ATTRID',object_id,HWMCA_ACTIVATION_PROFILE_SUFFIX);
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*****************************************************************/
   /* Get the activation profile attribute.                         */
   /*****************************************************************/
   rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
   if rc == HWMCA_DE_NO_ERROR then do
      profile = OUTPUT.1.DATA;
   end /* do */
   else do
      say 'Error' rc 'on RxHwmcaGet for activation profile attribute.';
   end /* do */
end /* do */
else do
   say 'Error' rc 'on RxHwmcaBuildId for activation profile attribute.';
end /* do */

return profile;
```

```
/*********************************************************************/
/* Subroutine: get_attribute                                         */
/*                                                                   */
/* This subroutine will perform a Get request for any attribute for */
/* the specified object.                                             */
/*********************************************************************/
get_attribute: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR;
parse arg object_id suffix .;

attr = '';
/*********************************************************************/
/* Build the object identifier for the object attribute.             */
/*********************************************************************/
rc = RxHwmcaBuildAttributeId('ATTRID',object_id,suffix);
if rc <> HWMCA_DE_NO_ERROR then do
   rc = RxHwmcaBuildId('ATTRID',object_id,suffix);
end /* do */
if rc == HWMCA_DE_NO_ERROR then do
   /*****************************************************************/
   /* Get the object attribute.                                     */
   /*****************************************************************/
   rc = RxHwmcaGet('INITBLK.',ATTRID,OUTPUT.,api_timeout)
   if rc == HWMCA_DE_NO_ERROR then do
      attr = OUTPUT.1.DATA;
   end /* do */
   else do
      say 'Error' rc 'on RxHwmcaGet for object attribute.';
   end /* do */
end /* do */
else do
   say 'Error' rc 'on RxHwmcaBuildId for object attribute.';
end /* do */

return attr;

/*********************************************************************/
/* Subroutine: perform_command                                       */
/*                                                                   */
/* This subroutine will ask the user for confirmation and then       */
/* perform the specified command.                                    */
/*                                                                   */
/* Note: We expose a lot of the HMC API variables that we defined    */
/*       earlier by calling the RxHwmcaDefineVars function.          */
/*********************************************************************/
perform_command: procedure expose api_timeout INITBLK. HWMCA_DE_NO_ERROR,
                                  HWMCA_NAME_SUFFIX hmc_name screen_cols,
                                  HWMCA_GROUP_CONTENTS_SUFFIX,
                                  HWMCA_CONSOLE_ID nest bailout,
                                  HWMCA_OBJECT_TYPE_SUFFIX,
                                  HWMCA_ACTIVATION_PROFILE_SUFFIX,
                                  HWMCA_CPC_OBJECT,
                                  HWMCA_INFINITE_WAIT,
                                  HWMCA_TYPE_INTEGER,
                                  HWMCA_TYPE_OCTETSTRING,
                                  HWMCA_TRUE,
                                  HWMCA_FALSE,
                                  HWMCA_DE_TIMEOUT,
                                  HWMCA_EVENT_COMMAND_RESPONSE,
                                  HWMCA_ACTIVATE_COMMAND,
                                  HWMCA_DEACTIVATE_COMMAND,
```

```
                                  HWMCA_SEND_OPSYS_COMMAND,
                                  HWMCA_RESETNORMAL_COMMAND,
                                  HWMCA_RESETCLEAR_COMMAND,
                                  HWMCA_START_COMMAND,
                                  HWMCA_STOP_COMMAND,
                                  HWMCA_LOAD_COMMAND,
                                  HWMCA_PSWRESTART_COMMAND;
parse arg command_id object_id object_type object_name;

return_msg = '';
CMDINPUT. = '';
CMDINPUT.0 = 0;
/*********************************************************************/
/* Determine the name of the command, so that we can use it when    */
/* displaying information to the user.                               */
/*********************************************************************/
select
   when command_id = HWMCA_ACTIVATE_COMMAND then do
      cmd_text = 'Activate';
   end  /* Do */
   when command_id = HWMCA_DEACTIVATE_COMMAND then do
      cmd_text = 'Deactivate';
   end  /* Do */
   when command_id = HWMCA_SEND_OPSYS_COMMAND then do
      cmd_text = 'Operating System Command';
   end  /* Do */
   when command_id = HWMCA_RESETNORMAL_COMMAND then do
      cmd_text = 'Reset';
   end  /* Do */
   when command_id = HWMCA_START_COMMAND then do
      cmd_text = 'Start';
   end  /* Do */
   when command_id = HWMCA_STOP_COMMAND then do
      cmd_text = 'Stop';
   end  /* Do */
   when command_id = HWMCA_PSWRESTART_COMMAND then do
      cmd_text = 'PSW Restart';
   end  /* Do */
   when command_id = HWMCA_LOAD_COMMAND then do
      cmd_text = 'Load';
   end  /* Do */
   otherwise do
      cmd_text = '';
      return_msg = 'Unknown command requested.';
   end  /* Do */
end  /* select */
if cmd_text <> '' then do
   /*****************************************************************/
   /* We have a command that we understand, so now ask the user if  */
   /* they are sure that they want to do this.                      */
   /*****************************************************************/
   call SysCls;
   say center(hmc_name||' - '||cmd_text||' Confirmation',screen_cols);
   say;
   say 'Target:' object_name;
   say;
   say 'Are you sure you want perform the' cmd_text 'task?';
   say "(Press 'Y' for Yes or 'N' for No)";
   say;
   call charout , '====> ';
   key = SysGetKey('ECHO');
   say;
   if key == 'Y' | key == 'y' then do
```

```
/************************************************************/
/* The user said go ahead, so let's issue the command.      */
/************************************************************/
select
   /*********************************************************/
   /* First, let's get the activation profile associated with */
   /* the object, so we can ask the user if they want to use  */
   /* that one or override it with another one.               */
   /*********************************************************/
   when command_id = HWMCA_ACTIVATE_COMMAND then do
      profile = get_profile(object_id);
      if profile <> '' then do
         say;
         say 'The activation profile currently associated with' object_name 'is:' profile'.';
      end  /* Do */
      say 'Please specify a new activation profile to be used or simply press Enter to';
      say 'accept the default activation profile.';
      call charout , '====> ';
      parse pull CMDINPUT.1.DATA .;
      if CMDINPUT.1.DATA  <> '' then do
         CMDINPUT.1.TYPE = HWMCA_TYPE_OCTETSTRING;
         CMDINPUT.0 = 1;
      end  /* Do */
   end  /* Do */
   /*********************************************************/
   /* Before issuing the request, we need to prompt the user  */
   /* for the command text and whether or not the command     */
   /* should be a priority command.                           */
   /*********************************************************/
   when command_id = HWMCA_SEND_OPSYS_COMMAND then do
      say;
      say 'Please enter the operating system command text.';
      call charout , '====> ';
      parse pull CMDINPUT.2.DATA
      CMDINPUT.2.TYPE= HWMCA_TYPE_OCTETSTRING;
      say;
      say "Should this command be issued as a priority command?";
      say "(Press 'Y' for Yes or 'N' for No)";
      call charout , '====> ';
      key = SysGetKey('ECHO');
      say;
      if key == 'Y' | key == 'y' then CMDINPUT.1.DATA = HWMCA_TRUE;
      else CMDINPUT.1.DATA = HWMCA_FALSE;
      CMDINPUT.1.TYPE= HWMCA_TYPE_INTEGER;
      CMDINPUT.0 = 2;
   end  /* Do */
   when command_id = HWMCA_LOAD_COMMAND then do
      say;
      say 'Please enter the Load address to be used.';
      call charout , '====> ';
      parse pull CMDINPUT.1.DATA
      CMDINPUT.1.TYPE= HWMCA_TYPE_OCTETSTRING;
      say;
      say 'Please enter the Load parameter to be used.';
      call charout , '====> ';
      parse pull CMDINPUT.2.DATA
      CMDINPUT.2.TYPE= HWMCA_TYPE_OCTETSTRING;
      say;
      say "Should memory be cleared before performing the Load?";
      say "(Press 'Y' for Yes or 'N' for No)";
```

```
      call charout , '====> ';
      key = SysGetKey('ECHO');
      say;
      if key == 'Y' | key == 'y' then CMDINPUT.3.DATA = HWMCA_TRUE;
      else CMDINPUT.3.DATA = HWMCA_FALSE;
      CMDINPUT.3.TYPE= HWMCA_TYPE_INTEGER;
      say;
      say 'Please enter the timeout value to use when performing the Load.';
      say "(Must be between 60 and 600 seconds.)";
      call charout , '====> ';
      parse pull CMDINPUT.4.DATA
      CMDINPUT.4.TYPE= HWMCA_TYPE_INTEGER;
      say;
      say "Should status be stored before performing the Load?";
      say "(Press 'Y' for Yes or 'N' for No)";
      call charout , '====> ';
      key = SysGetKey('ECHO');
      say;
      if key == 'Y' | key == 'y' then CMDINPUT.5.DATA = HWMCA_TRUE;
      else CMDINPUT.5.DATA = HWMCA_FALSE;
      CMDINPUT.5.TYPE= HWMCA_TYPE_INTEGER;
      CMDINPUT.0 = 5;
   end  /* Do */
   when command_id = HWMCA_RESETNORMAL_COMMAND then do
      say;
      say "Should memory be cleared as a part of the Reset?";
      say "(Press 'Y' for Yes or 'N' for No)";
      call charout , '====> ';
      key = SysGetKey('ECHO');
      say;
      if key == 'Y' | key == 'y' then command_id = HWMCA_RESETCLEAR_COMMAND;
   end  /* Do */
   otherwise nop;
end  /* select */
/*************************************************************/
/* Now we are all set, so lets issue the command.            */
/*************************************************************/
say;
say 'Issuing the' cmd_text 'command request...';
rc = RxHwmcaCommand('INITBLK.',object_id,command_id,'CMDINPUT.',api_timeout);
if rc == HWMCA_DE_NO_ERROR then do
   /*********************************************************/
   /* The request was successful, so now let's wait for the */
   /* completion response.                                  */
   /*********************************************************/
   say;
   call charout , 'Waiting for the' cmd_text 'task to complete.';
   cmd_done = 0;
   do while cmd_done == 0 & rc == HWMCA_DE_NO_ERROR
      rc = RxHwmcaWaitEvent('INITBLK.','OUTPUT.',5000);
      if rc == HWMCA_DE_NO_ERROR then do
         /*****************************************************/
         /* Let's make sure that this event is a command      */
         /* response for the same command and target for our  */
         /* request.                                          */
         /*****************************************************/
         if OUTPUT.2.DATA == HWMCA_EVENT_COMMAND_RESPONSE then do
            /* It is a command response event */
            if OUTPUT.1.DATA == object_id then do
               /* It is a response for our target object */
```

```
                     if OUTPUT.4.DATA == command_id then do
                        /* It is the command we issued */
                        cmd_done = 1;
                        if OUTPUT.6.DATA == HWMCA_DE_NO_ERROR then do
                           return_msg = cmd_text 'command completed successfully.';
                        end  /* Do */
                        else do
                           return_msg = cmd_text 'command failed with rc' OUTPUT.6.DATA'.';
                        end  /* Do */
                     end  /* Do */
                  end  /* Do */
               end  /* Do */
            end  /* Do */
            else do
               if rc == HWMCA_DE_TIMEOUT then do
                  rc = HWMCA_DE_NO_ERROR;
                  call charout , '.';
               end  /* Do */
               else do
                  return_msg = 'Error' rc 'on RxHwmcaWaitEvent call for' cmd_text 'command.';
               end  /* Do */
            end  /* Do */
         end /* do */
      end /* do */
      else do
         return_msg = 'Error' rc 'on RxHwmcaCommand call for' cmd_text 'command.';
      end /* do */
   end  /* Do */
   else do
      return_msg = 'Command request cancelled.';
   end  /* Do */
end  /* Do */

return return_msg;
```

Chapter 6. Configuring for the data exchange APIs

Before the Console APIs (Data Exchange APIs and Commands API) can be used, some configuration tasks must be performed on the Hardware Management Console or Support Element Console. These configuration tasks fall into two categories:

- SNMP configuration
- Console API configuration.

Refer to the information about the following pages for detailed steps necessary to perform these two types of configuration.

Note: Once these steps have been successfully completed, the Console APIs (Data Exchange APIs and Commands API) can be used while the Hardware Management Console or Support Element Console is up and running. The APIs will not be functional when the console is not running, even if these configuration steps have been completed.

Note: For Consoles Version 2.9.0 or later the SNMP and Console API configuration tasks have been merged into a single task named Customize API Settings.

Configuring for SNMP (for consoles earlier than version 2.9.0)

The Console uses the SNMP support provided by the SystemView® Agent for OS/2. Use the following procedure to enable the SystemView Agent for OS/2. If you need more information, refer to *SystemView Agent for OS/2 User's Guide* .

To configure the SystemView Agent for OS/2:

1. Log on to the Console in *Access Administrator* mode.
2. Start the SNMP Configuration task, which can be found under **Console Actions** in the *Views area* of the Console.
3. Add one or more entries in the **Community Name Information** box by selecting the **Communities** tab of the SNMP Configuration notebook window. After specifying the following information, select the **Add** push button to add a new community name or select the **Change** push button to change an existing community name.

   It is recommended that one entry be added for the Console itself and one additional entry for each TCP/IP host (machine) that will be making Management API requests.

   It is important that a valid entry be specified for the Console. This entry must match both the Console TCP/IP address and community name (specified in step 5 on page 193). It must also specify the use of the UDP protocol.

   Protocol
   Use this field to specify the communications protocol over which the community name is valid. This must be set to **UDP** for the community name that is to be used by this Console.

   The community name(s) that are to be used by applications using the Console APIs should also be set to **UDP**.

   Name This field should be filled in with any character string. Each community name in the list must be unique. Please note that this field is case sensitive.

   Note the community name that should be used by the Console, since it will need to be specified in step 5 on page 193.

Note the community name(s) that are to be used by applications using the Console APIs, since it will need to be specified on the *HwmcaInitialize* calls issued by those applications.

Address
If you are following the recommendation of adding an entry for the Console and each requesting application, then this field should be filled in with the TCP/IP address of the machine that will be using the community name. If you are not following this recommendation, then refer to the SystemView Agent for OS/2 documentation for further details on this field.

Note: The Console's TCP/IP address can be obtained by viewing the **Network** page of the *Hardware Management Console Settings* or *Support Element Settings* task. This task is found under **Console Actions** in the *Views area* and can only be performed by a user logged on to the Console in *Access Administrator* mode.)

Network Mask
If you are following the recommendation of adding an entry for the Console itself and each requesting application, then this field should be filled in with a character string of **255.255.255.255**. If you are not following this recommendation, then refer to the SystemView Agent for OS/2 documentation for further details on this field.

Access Type
Use this field to specify the type of access that is allowed for the community name.

The access type for the community name that is to be used by the Console can be either **read only** or **read/write**.

The access type for the community name(s) that are to be used by applications using the Console APIs *MUST* be **read/write**.

4. Update the *MIB Variables* by selecting the **MIB Variables** tab of the SNMP Configuration notebook window. (This step is optional since the SystemView Agent for OS/2 provides default values for any *MIB Variable* that is not specified.)

   Description
   Use this field to specify a meaningful name for this Console.

   Contact
   Use this field to specify the name of the contact person for this Console.

   Name Use this field to specify the TCP/IP hostname of this Console.

   Location
   Use this field to specify the physical location of this Console.
5. Select the **OK** push button to save the changed settings and close the SNMP Configuration notebook window.
6. If any of the above data was added or changed, you need to shut down and restart the Console before the changes will be put into effect. However, before doing so, continue with the configuration steps for the Console below.

Configuring the console for API (for consoles earlier than version 2.9.0)

The Console API configuration steps can be performed by using the **API** page of the *Hardware Management Console Settings* or *Support Element Settings* task. This task is found under **Console Actions** in the *Views area*.

To configure the Console for API support:

1. Log on to the Console in *Access Administrator* mode.
2. Open the **Hardware Management Console Settings** or the **Support Element Settings** task.

3. Select the **API** tab.
4. Check the **Enable the Hardware Management Console Application Program Interface** checkbox for the Hardware Management Console Application or the **Enable the Support Element Console Application Program Interface** checkbox for the Support Element.
5. Specify the **community name** to be used by the Console. Refer to step 3 on page 191 for more details about the community name and how it is specified.
6. Specify any **SNMP agent parameters** that should be used when the Console automatically starts the SystemView Agent for OS/2. Refer to the *SystemView Agent for OS/2 User's Guide* for more information regarding the SystemView Agent for OS/2 parameters.

   Note: In order for the Console to be able to communicate with the SystemView Agent for OS/2, the parameters **-transport udp** and **-dpi tcp** must be specified.
7. Specify any additional locations where enterprise-specific SNMP trap messages created by the Console should be sent in the **Event notification information** box. Entries can be added, changed, and deleted throughout with the use of the **New**, **Change**, and **Delete** push buttons respectively.

   Adding entries to the **Event notification information** box will cause the Console to send the specified event notifications to TCP/IP port 162 at the locations specified.
8. Select the **Apply** push button to save the changes.
9. If any of the above data was added or changed, then you need to stop and restart the Console before the changes will be in effect. The Console can be stopped by logging off and then selecting the **Cancel** push button on the logon window. The Hardware Management Console Application can be restarted by starting the Hardware Management Console Application icon from the desktop, while the Support Element Console can be restarted by rebooting the Support Element console.

Configuration problems

If there are configuration mistakes with either the SNMP configuration or the Console API configuration, the Console APIs will not be functional. The nature of the configuration problem can be determined by analyzing the Hardware Message that was created when the Console was started. The details of this Hardware Message will list the exact configuration problem(s) that were found, along with corrective actions.

Configuring the console for API (for consoles version 2.9.0 or later)

The Console API configuration steps can be performed by using the *Customize API Settings* task found in the *Hardware Management Console Settings* or *Support Element Settings* group of tasks. This task is found under **Console Actions** in the *Views* area.

To configure the Console for API support:

1. Log on to the Console in *Access Administrator* mode.
2. Open the **Hardware Management Console Settings** or the **Support Element Settings** group of tasks.
3. Open the **Customize API Settings** task.
4. Check the **Enable SNMP APIs**.
5. Specify any **SNMP agent parameters** desired. **Note:** No special SNMP agent parameters are required for API to work correctly.
6. Add one or more entries in the **Community Names** box by selecting **Add** push button to add a new community name or select the **Change** push button to change an existing community name. It is recommended that one entry be added for each TCP/IP host (machine) that will be making Management API requests.

   Name This field should be filled in with any character string. Each community name in the list must be unique. Note the community name(s) that are to be used by applications using the Console APIs, since it will need to be specified on the *HwmcaInitialize* calls issued by those applications.

Address

If you are following the recommendation of adding an entry for the Console and each requesting application, then this field should be filled in with the TCP/IP address of the machine that will be using the community name.

Note: This can be specified as an IPV6 address.

Network Mask/Prefix

If you are following the recommendation of adding an entry for the Console itself and each requesting application, then this field should be filled in with a character string of **255.255.255.255**. When using IPV6 addresses, the prefix for the address should be used, instead of a masked value.

Access Type

Use this field to specify the type of access that is allowed for the community name. The access type for the community name(s) that are to be used by applications using the Console APIs *MUST* be **read/write**.

7. Specify any additional locations where enterprise-specific SNMP trap messages created by the Console should be sent in the **Event notification information** box. Entries can be added, changed, and deleted throughout with the use of the **Add**, **Change**, and **Delete** push buttons respectively. Adding entries to the **Event notification information** box will cause the Console to send the specified event notifications to TCP/IP port 162 at the locations specified. 8. Select the **OK** or **Apply** push buttons to save the changes.
8. If any of the above data was added or changed, then you need to stop and restart the Console before the changes will be in effect. The Console can be stopped by using the **Shutdown** or **Restart** task found in the **Console Actions** view.

Appendix A. Building an application

The following information should be helpful when trying to build an application that uses the Console Application Programming Interfaces. All of the files necessary to build and run an API application are preloaded on Hardware Management Consoles for versions earlier than Version 2.9.0.

The most up to date copies of these build and run-time files are now available on Resource Link at *http://www.ibm.com/servers/resourcelink*. Click on **Services**, and then Click **API**.

Hardware Management Console (prior to version 2.9.0)

The Console Application Programming Interfaces build and run-time files are preloaded on the Hardware Management Console so applications executing on the Hardware Management Console can make direct use of these interfaces once they are successfully built. The Data Exchange APIs and Commands API (both the C language and Rexx interfaces) can be executed from an OS/2 workstation other than the Hardware Management Console workstation. The files that need to be moved to the other workstation for this are:

ACTZSAPI.DLL
: The OS/2 dynamic link library containing the C language Data Exchange APIs and Commands API. This file can be found in the D:\DYNALINK directory of the Hardware Management Console.

ACTZSNMP.DLL
: The OS/2 dynamic link library containing the Rexx language Data Exchange APIs and Commands API. This file can be found in the D:\DYNALINK directory of the Hardware Management Console.

The D:\TOOLKIT directory of the Hardware Management Console contains everything necessary to build an application that uses the Hardware Management Console Application Programming Interfaces. The items included in this directory are:

HWMCAAPI.H
: This C language include file contains all of the constant definitions, structure definitions, and function prototypes for the HWMCA Management APIs.

HWMCAAPI.LIB
: This library file contains all of the linkages needed in order to use the HWMCA Management APIs.

  Note: The OS/2 dynamic link library referenced by this library file can be found in the D:\DYNALINK directory of the Hardware Management Console with the name ACTZSAPI.DLL

HWMCATST.C
: This C language source file is a copy of the example Management API application found in “Data exchange APIs and commands API example” on page 62.

HWMCATST.MAK
: This OS/2 MAKE file can be used to build the example Management API application found in “Data exchange APIs and commands API example” on page 62.

HWMCATST.EXE
: This is an executable version of the Management API application found in “Data exchange APIs and commands API example” on page 62.

HWMCARX.CMD
This Rexx command file is a copy of the example Rexx Management API application found in "Data exchange APIs (REXX sample)" on page 167.

HWMCAWIN.DLL
The 32-bit Windows dynamic link library containing the C language Data Exchange APIs and Commands API.

HWMCAORX.DLL
The 32-bit Windows dynamic link library containing the Object Rexx language Data Exchange APIs and Commands API.

HWMCAWIN.LIB
This library file contains all of the linkages needed in order to use the HWMCA Management APIs on a 32-bit Windows platform.

HWMCAWIN.EXE
This is a 32-bit Windows executable version of the Management API application found in "Data exchange APIs and commands API example" on page 62.

HWMCAV1.MIB
This is a SNMP version 1 based Management Information Base (MIB) that describes the entities that can be managed by Management APIs.

HWMCAV2.MIB
This is a SNMP version 2 based Management Information Base (MIB) that describes the entities that can be managed by Management APIs.

HWMCAAIX.TAR
This file is an AIX® TAR file containing the following files:

hwmcaaix
AIX library file containing all of the linkages needed in order to use the HWMCA Management APIs on an AIX platform.

libhwmcaaix.so
Shared Object Library containing the C/C++ language HWMCA Management APIs.

HWMCAMVS.TAR
This file is an z/OS® or OS/390® OpenEdition TAR file containing the following files:

hwmcamvs
This is an OpenEdition HFS executable version of the Management API application found in "Data exchange APIs and commands API example" on page 62.

hwmcaapi.x
z/OS or OS/390 library file containing all the linkages needed in order to use the HWMCA Management APIs on an z/OS or OS/390 platform.

hwmcaapi
z/OS or OS/390 DLL containing the C/C++ language HWMCA Management APIs.

HWMCA386.TAR
This file is an Intel based Linux TAR file containing the following files:

hwmcalnx
This is an Intel Linux executable version of the Management API application found in "Data exchange APIs and commands API example" on page 62.

libhwmcalnx.so
Link to ibhwmcalnx.so.0.

libhwmcalnx.so.0
Link to libhwmcalnx.so.0.0.

libhwmcalnx.so.0.0
Intel Linux Shared Object Library containing the C/C++ language HWMCA Management APIs.

The Hardware Management Console Application Programming Interface OS/2 dynamic link libraries are built using the IBM VisualAge® C++ complier. These dynamic link libraries are written as 32- bit interfaces and should be invoked accordingly.

When building an application that uses some of the Hardware Management Console Application Programming Interfaces, make sure that the C language include files are located in a directory found in the INCLUDE environment variable, and the interface library files are located in a directory found in the LIB environment variable.

Appendix B. HWMCA_EVENT_COMMAND_RESPONSE return codes

Following is a list of HWMCA_EVENT_COMMAND_RESPONSE return codes and their descriptions. The return code values are shown as hexadecimal values with the decimal equivalent in parentheses.

0806000A (134610954) Resource unknown.

Explanation: The profile name specified in an operations command is not recognized by the receiving node.

Programmer response: Correct the configuration identifier and re-send the request.

08090000 (134807552) Mode inconsistency: The requested function cannot be performed in the present state of the receiver.

Explanation: This command is prohibited because the target is in an incompatible mode. For example, an ITIMER request is not accepted when the system is power-on reset in LPAR mode.

Programmer response: This function cannot be performed in the present state of the receiver. Retry the request after the target mode status has changed.

08090001 (134807553) Mode inconsistency: The requested function cannot be performed in the present state of the receiver.

Explanation: Acceptance of the command is prohibited because the target is in an incompatible mode. For example, an ITIMER request is not accepted when the system is power-on reset in LPAR mode.

Programmer response: None. This function cannot be performed in the present state of the receiver.

080A000A (134873098) Permission rejected: The receiver had denied an implicit or explicit request of the sender.

Explanation: A STATLEV request was rejected because it was not compatible with the status reporting values set in the receiver.

Programmer response: Correct the STATLEV value and re-send the request.

080C0005 (135004165) Procedure not supported: A procedure specified is not supported in the receiver.

Explanation: The command is not supported.

Programmer response: re-send the request using a supported command, if possible.

080C0007 (135004167) Procedure not supported: A procedure specified is not supported in the receiver.

Explanation: A request for a function is supported by the receiver, but the resource identified in the request does not support that function.

Programmer response: None. This function cannot be canceled.

08120000 (135397376) Insufficient resource: The receiver cannot act on the request because of a temporary lack of resource.

Explanation: System resources are temporarily busy.

Programmer response: re-send command if required.

08120011 (135397393) Insufficient resource: The receiver cannot act on the request because of a temporary lack of resource.

Explanation: Insufficient storage is available to the target component to satisfy the request.

Programmer response: re-send command.

08150001 (135593985) Function active: A request to activate an element or procedure was received, but the element or procedure was already active.

Explanation: Unable to perform the command because the target CPC Subset or CPC Image is operational and the force operand has not indicated the override selection.

Programmer response: Put the system in the appropriate state and re-send the command.

081A0000 (135921664) Request sequence error.

Explanation: Unable to perform the command because the target partition is in the deactivated state.

Programmer response: Activate the logical partition, then re-send the original request.

081A0009 (135921673) Request sequence error.

Explanation: Unable to perform command because power is not on.

Programmer response: Send a POWERON or ACTIVATE command, then re-send the original request.

081A000A (135921674) Request sequence error.

Explanation: Unable to perform command because power-on reset is not complete.

Programmer response: Send a POWERON or ACTIVATE command, then re-send the original request.

081A000B (135921675) Request sequence error.

Explanation: Unable to perform command because the targeted CPU is not in the stopped state.

Programmer response: Send a STOP command, then re-send the original request.

081A000E (135921678) Request sequence error.

Explanation: Unable to perform command because the interval timer is present only when the CPC Image is operating in S/370 mode.

Programmer response: None. The requested command cannot be performed when the system is power-on reset in either ESA/390 mode or LPAR mode.

081A0010 (135921680) Request sequence error.

Explanation: The request is rejected or failed because the target resource is already in the state or condition that the request would have provided.

Programmer response: None. The requested command has already been performed.

081C0005 (136052741) Request not executable: The requested function cannot be executed because of a permanent error condition in the receiver.

Explanation: A power-on request failed.

Programmer response: Verify that power is available and re-send the command.

081C0006 (136052742) Request not executable: The requested function cannot be executed because of a permanent error condition in the receiver.

Explanation: A POR(YES) or POR(IML) failed. This may be accompanied by a hardware alert.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

081C0007 (136052743) Request not executable: The requested function cannot be executed because of a permanent error condition in the receiver.

Explanation: An operating system load request (for example, LOAD) failed.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

081C000A (136052746) Request not executable: A POWEROFF request cannot be performed because of a permanent error condition in the receiver.

Explanation: A power off request failed due to an unexpected power status.

Programmer response: Reset any abnormal power conditions at the receiver, such as tripped CBs, and retry the power off command. Call for service if the problem persists.

081C00BA (136052922) Request not executable: The requested function cannot be executed because of a permanent error condition in the receiver.

Explanation: The receiver has an error resulting from a licensed internal code problem that prevents execution of the request.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

082D0001 (137166849) Busy.

Explanation: Resources needed to process the request are being used.

Programmer response: Wait for the resources to be released, then re-send the request.

08380000 (137887744) Request not executable because of resource or component state incompatibility: The request is not executable because it is not compatible with the state of a resource or component in the receiver.

Explanation: Unable to perform the command because the system is in an invalid state.

Programmer response: Put the system in a state that is compatible with the requested command and re-send the request.

0838001B (137887771) Request not executable because of resource or component state incompatibility: The request is not executable because it is not compatible with the state of a resource or component in the receiver.

Explanation: Request will not be honored because it was submitted to a node at a time when a local operator or other application reserved control of the node.

Programmer response: Request the local operator to release control (log off), or retry later.

08380037 (137887799) MVS is not receiving. The request is not executable because the MVS operating system is not able to respond because it is in an inactive or quiesced state.

Explanation: Request will not be honored because it requires that the resource operating system is in an active state.

Programmer response: Reissue the command after the operating system has been reactivated.

084F0000 (139395072) Resource not available: A requested resource is not available to service the given request.

Explanation: A resource error exits which may indicate a configuration problem or insufficient resource to execute the command.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

085B0000 (140181504) Unknown resource name: The identified resource required to complete the requested command is not known.

Explanation: The profile name specified in the AUTOACT operand of the RESET profile is not recognized by the receiving node.

Programmer response: Correct the profile name and re-send the request.

085C0000 (140247040) System exception. The node experiences an exception condition within a resident system or subsystem that inhibits further processing by the component.

Explanation: An internal error has occurred with the processing of this request. This may be accompanied by a hardware alert.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

085C0001 (140247041) System exception: The node experiences an exception condition within a resident system or subsystem that inhibits further processing by the component.

Explanation: The exception is identifiable as a system-related problem. This may be accompanied by a hardware alert.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

085C0002 (140247042) System exception: The node experiences an exception condition within a resident system or subsystem that inhibits further processing by the component.

Explanation: The exception is identified as a permanent system-related problem. This may be accompanied by a hardware alert.

Programmer response: If the code is returned for an ACTIVATE request, to complete activation, send another ACTIVATE request to complete the initial program load.

For all other requests, retry the operation. Contact the IBM Service Support System if the problem persists.

08B20002 (145883138) Data transmission failure: The data transmission between an application in the support element and an application in the processor was incomplete, causing abnormal termination of the function.

Explanation: A time-out has occurred while waiting for transmission of data between two applications.

Programmer response: Retry operation. Contact the IBM Service Support System if the problem persists.

100B0001 (269156353) Required structure absent.

Explanation: An operand required by the command was not found in the command string.

Programmer response: Enter the required operand and re-send the request.

100B0003 (269156355) Multiple occurrences of a nonrepeatable structure.

Explanation: A value that cannot be repeated was detected in the command string.

Programmer response: Change the duplicate value(s) to unique value(s) and re-send the request.

100B0006 (269156358) Length outside specified range.

Explanation: The length of the operand indicated in SDATA is outside the allowable range.

Programmer response: Correct the operand data value and re-send the request.

100B000B (269156363) Precluded combination of structures and data values present.

Explanation: One command operand or data value is in conflict with one or more other operands or data values.

Programmer response: Remove the precluded operand(s) or correct the command and re-send the request. Also check the activation profile(s) used for activation, as the error may be the result of incorrect profile data.

100B000C (269156364) Unknown or unsupported data value.

Explanation: The data value in the operand indicated by SDATA is either unknown or unsupported.

Programmer response: Correct the operand data value and re-send the request.

100B000D (269156365) Incompatible data values.

Explanation: The data value in the operand indicated by SDATA is not compatible with this or other values.

Programmer response: Correct the conflicting operand data value and re-send the request.

100B0012 (269156370) Recognized but unsupported structure.

Explanation: The operand indicated by SDATA is recognized but not supported by the target support element.

Programmer response: Remove the unsupported operand and re-send the request.

80180002 (2149056514) Resource unknown.

Explanation: The secondary OCR specified in the OCFNAME operand is not recognized.

Programmer response: Ensure that the system is power-on reset in LPAR mode and the secondary name in the OCFNAME operand matches a logical partition name in the active IOCDS.

Appendix C. API return codes

Data exchange API call return codes

Following is a list of return codes and their descriptions, which can be returned from the various Data Exchange API calls. (The decimal values are shown in parentheses).

(0) HWMCA_DE_NO_ERROR

Explanation: A Data Exchange API call has completed successfully.

Programmer response: None.

(1) HWMCA_DE_NO_SUCH_OBJECT

Explanation: A Data Exchange API call specified an object identifier that does not exist.

Programmer response: Check the specified object identifier to ensure that it is valid and that the API support is enabled and functioning correctly on the target console.

(2) HWMCA_DE_INVALID_DATA_TYPE

Explanation: A HwmcaSet Data Exchange API call specified an invalid data type.

Programmer response: Check the specified data type value to ensure that is one of the supported values and that is appropriate for the target object identifier.

(3) HWMCA_DE_INVALID_DATA_LENGTH

Explanation: Either a HwmcaSet Data Exchange API call specified a data length value that is not appropriate for the corresponding data type or is not appropriate for the target object identifier, or the result of a HwmcaGet or HwmcaGetNext is too large to be transported by the underlying transport protocol.

Programmer response: Either check to ensure that a length of zero is used for a data type of HWMCA_TYPE_NULL and that a length of 1, 2, or 4 is used for a data type of HWMCA_TYPE_INTEGER, or use an alternative approach for retrieving the desired data.

(4) HWMCA_DE_INVALID_DATA_PTR

Explanation: A HwmcaSet Data Exchange API call specified a data pointer that is not appropriate for the corresponding data type.

Programmer response: Check to ensure that a null pointer is used for a data type of HWMCA_TYPE_NULL and that a non-null pointer is used for all other data types.

(5) HWMCA_DE_INVALID_DATA_VALUE

Explanation: A HwmcaSet Data Exchange API call specified a data value that is not appropriate for the target object identifier.

Programmer response: Check to make sure that the data value is one of the allowed value for the target object identifier.

(6) HWMCA_DE_INVALID_INIT_PTR

Explanation: A Data Exchange API call specified null as the pointer to the HWMCA_INITIALIZE_T structure.

Programmer response: Make sure that this value is specified as a pointer to a valid HWMCA_INITIALIZE_T structure.

(7) HWMCA_DE_INVALID_ID_PTR

Explanation: A Data Exchange API call specified a null pointer as the object identifier parameter.

Programmer response: Make sure that this value is specified as a pointer to a valid object identifier string.

(8) HWMCA_DE_INVALID_BUF_PTR

Explanation: A Data Exchange API call specified null as the pointer to the output buffer.

Programmer response: Make sure that this value is specified as a pointer to an address of the output buffer.

(9) HWMCA_DE_INVALID_BUF_SIZE

Explanation: A Data Exchange API call specified zero as the length of the output buffer.

Programmer response: Make sure that this parameter is a non-zero value.

(10) HWMCA_DE_INVALID_DATATYPE_PTR

Explanation: A HwmcaSet Data Exchange API call specified null as the pointer to the

HWMCA_DATATYPE_T structure used to describe the data to be used for the set operation.

Programmer response: Make sure that this value is specified as a pointer to an address of a valid HWMCA_DATATYPE_T structure.

---

(11) HWMCA_DE_INVALID_TARGET

Explanation: A HwmcaInitialize Data Exchange API call specified an invalid host name or internet address for the target console.

Programmer response: . Make sure that the value pointed to by the pHost field of the HWMCA_SNMP_TARGET_T structure is internet address or hostname.

---

(12) HWMCA_DE_INVALID_EVENT_MASK

Explanation: A HwmcaInitialize Data Exchange API call specified a value in the ulEventMask field of the HWMCA_INITIALIZE_T structure that is not valid.

Programmer response: Make sure that this field only contains some combination of the valid event mask values.

---

(13) HWMCA_DE_INVALID_PARAMETER

Explanation: A Data Exchange API call specified an invalid parameter. Depending on the API call being made, one of the following problems occurred:

HwmcaInitialize

- The HWMCA_INITIALIZE_T structure used on a previous HwmcaInitialize call specifies a host name or internet address specified that is different that what was initially specified.
- The ulReserved field of HWMCA_INITIALIZE_T structure contains a non-null value.

HwmcaBuildAttributeId
The pointer to the attribute suffix string was specified as a null pointer.

HwmcaGet or HwmcaGetNext or HwmcaWaitEvent
The pointer to the value to be filled in with the number of bytes needed for the output buffer was specified as a null pointer.

---

(14) HWMCA_DE_READ_ONLY_OBJECT

Explanation: A HwmcaSet Data Exchange API call specified a target object identifier that is read only.

Programmer response: . Make sure to use a target object identifier that allows for write access.

---

(15) HWMCA_DE_SNMP_INIT_ERROR

Explanation: A HwmcaInitialize Data Exchange API call encountered an error trying to create/allocate the internal resources necessary to complete the operation. For example, memory could not be allocated successfully or TCP/IP sockets could not be created.

Programmer response: Make sure that the necessary resources are available to be used on the requesting machine.

---

(16) HWMCA_DE_INVALID_OBJECT_ID

Explanation: A Data Exchange API call was made with an invalid object identifier.

Programmer response: Check the object identifier specified on the call to make sure that it is specified correctly and is a valid object identifier.

---

(17) HWMCA_DE_REQUEST_ALLOC_ ERROR

Explanation: A Data Exchange API call encountered an error trying to allocate some temporary storage for internal use.

Programmer response: Make sure that enough memory is available on the requesting machine.

---

(18) HWMCA_DE_REQUEST_SEND_ ERROR

Explanation: A Data Exchange API encountered an error trying to send a request to the target console.

Programmer response: This is typically due to a network error of some sort.

---

(19) HWMCA_DE_TIMEOUT

Explanation: A Data Exchange API timed out while waiting for a response. For the HwmcaWaitEvent API call, this simply means that no events were received within the specified time period, so the calling application should proceed accordingly. For other Data Exchange API calls, the response was not received within the specified time period.

Programmer response: Make sure that the timeout value is large enough to allow for the request to be completed and the response to be returned.

---

(20) HWMCA_DE_REQUEST_RECV_ ERROR

Explanation: A Data Exchange API encountered an error trying to receive a response from the target console. This is typically due to a network error of some sort.

Programmer response: Investigate the possibility of a network error.

---

(21) HWMCA_DE_SNMP_ERROR

Explanation: A Data Exchange API call received a response that contained an unrecognized error status value.

Programmer response: Make sure that no errors were reported on the target console that would have resulted in incomplete or invalid data to be sent as a response.

---

(22) HWMCA_DE_INVALID_TIMEOUT

Explanation: A Data Exchange API call was made with the timeout value specified as zero.

Programmer response: Make sure that an appropriate non-zero timeout value is specified on the API call.

---

(28) HWMCA_DE_INVALID_HOST

Explanation: A HwmcaInitialize Data Exchange API call was made with a null pointer value specified as the pHost field of the HWMCA_SNMP_TARGET_T structure.

Programmer response: Make sure this field points to a valid hostname or internet address.

---

(29) HWMCA_DE_INVALID_COMMUNITY

Explanation: A HwmcaInitialize Data Exchange API call was made with a zero length string value specified as the szCommunity field of the HWMCA_SNMP_TARGET_T structure.

Programmer response: Make sure that this field contains a community name string with a length greater than zero.

---

(30) HWMCA_DE_INVALID_QUALIFIER

Explanation: A HwmcaInitialize Data Exchange API call was made that specified event qualification data, but the ulType field of the HWMCA_EVENT_QUALIFIER_T structure contained an invalid value.

Programmer response: Make sure that this field contains a valid event qualifier type value.

---

(98) HWMCA_DE_REQUIRES_QUALIFIER

Explanation: A HwmcaInitialize Data Exchange API call was made that specified an event mask indicating that requires additional event qualification information to be provided.

Programmer response: Make sure to provide the necessary event qualification information.

---

(99) HWMCA_TRANSPORT_ERROR

Explanation: A Data Exchange API call was made but an error was encountered in the transport layer that was specified to deliver the request data and return the response data.

Programmer response: Refer to information about the specific transport layer being used for more details regarding this error.

Command API call return codes

Following is a list of return codes and their descriptions, which can be returned from the various Command API call. (The decimal values are shown in parentheses).

(0) HWMCA_CMD_NO_ERROR

Explanation: A HwmcaCommand API call has completed successfully.

Programmer response: None.

(1) HWMCA_CMD_NO_SUCH_OBJECT

Explanation: A hwmcaCommand API call specified an object identifier that does not exist.

Programmer response: Check the specified object identifier to ensure that it is valid and that the API support is enabled and functioning correctly on the target console.

(2) HWMCA_CMD_INVALID_DATA_ TYPE

Explanation: A HwmcaCommand API call specified an invalid data type.

Programmer response: Check the specified data type value to ensure that is one of the supported values and that is appropriate for the target object identifier.

(3) HWMCA_CMD_INVALID_DATA_ LENGTH

Explanation: A HwmcaCommand API call specified a data length value that is not appropriate for the corresponding data type or is not appropriate for the target object identifier.

Programmer response: Check to ensure that a length of zero is used for a data type of HWMCA_TYPE_NULL and that a length of 1, 2, or 4 is used for a data type of HWMCA_TYPE_INTEGER.

(4) HWMCA_CMD_INVALID_DATA_ PTR

Explanation: A HwmcaCommand API call specified a data pointer that is not appropriate for the corresponding data type.

Programmer response: Check to ensure that a null pointer is used for a data type of HWMCA_TYPE_NULL and that a non-null pointer is used for all other data types.

(5) HWMCA_CMD_INVALID_DATA_ VALUE

Explanation: A HwmcaCommand API call specified a data value that is not appropriate for the target object identifier.

Programmer response: Check to make sure that the data value is one of the allowed value for the target object identifier.

(6) HWMCA_CMD_INVALID_INIT_ PTR

Explanation: A HwmcaCommand API call specified null as the pointer to the HWMCA_INITIALIZE_T structure.

Programmer response: Make sure that this value is specified as a pointer to a valid HWMCA_INITIALIZE_T structure.

(7) HWMCA_CMD_INVALID_ID_PTR

Explanation: A HmcaCommand API call specified a null pointer as the object identifier parameter.

Programmer response: Make sure that this value is specified as a pointer to a valid object identifier string.

(10) HWMCA_CMD_INVALID_ DATATYPE_PTR

Explanation: A HwmcaCommand API call specified null as the pointer to the HWMCA_DATATYPE_T structure used to describe the data to be used for the command parameter information.

Programmer response: Make sure that this value is specified as a pointer to an address of a valid HWMCA_DATATYPE_T structure.

(11) HWMCA_CMD_INVALID_TARGET

Explanation: A HwmcaCommand API call specified an invalid target.

Programmer response: Check the specified object identifier to ensure that it is valid and that there is not already an active command for the specified object.

(13) HWMCA_CMD_INVALID_ PARAMETER

Explanation: A HwmcaCommand API call specified an invalid parameter.

Programmer response: Make sure all the required parameters are correctly specified.

(17) HWMCA_CMD_REQUEST_ALLOC_ ERROR

Explanation: A HwmcaCommand PI call encountered an error trying to allocate some temporary storage for internal use.

Programmer response: Make sure that enough memory is available on the requesting machine.

---

(18) HWMCA_CMD_REQUEST_SEND_ERROR

Explanation: A HwmcaCommand API encountered an error trying to send a request to the target console. This is typically due to a network error of some sort.

Programmer response: Investigate the possibility of a network error.

---

(19) HWMCA_CMD_TIMEOUT

Explanation: A HwmcaCommand API timed out while waiting for a response. The response was not received within the specified time period.

Programmer response: Make sure that the timeout value is large enough to allow for the request to be completed and the response to be returned.

---

(20) HWMCA_CMD_REQUEST_RECV_ERROR

Explanation: A HwmcaCommand API encountered an error trying to receive a response from the target console. This is typically due to a network error of some sort.

Programmer response: Investigate the possibility of a network error.

---

(21) HWMCA_CMD_SNMP_ERROR

Explanation: A HwmcaCommand API call received a response that contained an unrecognized error status value.

Programmer response: Make sure that no errors were reported on the target console that would have resulted in incomplete or invalid data to be sent as a response.

---

(22) HWMCA_CMD_INVALID_TIMEOUT

Explanation: A HwmcaCommand API call was made with the timeout value specified as zero.

Programmer response: Make sure that an appropriate non-zero timeout value is specified on the API call.

---

(23) HWMCA_CMD_INVALID_CMD

Explanation: A HwmcaCommand API call was made with an invalid command object identifier.

Programmer response: Make sure that the command object identifier corresponds to a valid command for the target object.

---

(24) HWMCA_CMD_OBJECT_BUSY

Explanation: A HwmcaCommand API call was made specifying a target object identifier for an object that is currently busy performing another command.

Programmer response: . Retry the call again after the target object is no longer busy.

---

(25) HWMCA_CMD_INVALID_OBJECT

Explanation: A HwmcaCommand API call was made with a target object identifier that is not valid for the specified command.

Programmer response: Make sure that the command object identifier corresponds to an appropriate command for the target object.

---

(26) HWMCA_CMD_COMMAND_FAILED

Explanation: A HwmcaCommand API call failed due to an internal error on the target console.

Programmer response: Check the target console for details regarding the error.

---

(27) HWMCA_CMD_INITTERM_OK

Explanation: This is a value only used internally and should never be received by the calling application.

HWMCA_EVENT_COMMAND_RESPONSE return codes

The following values are returned as HWMCA_EVENT_COMMAND_RESPONSE return codes for HWMCA_ACTIVATE_CBU_COMMAND, HWMCA_UNDO_CBU_COMMAND, HWMCA_ADD_CAPACITY_COMMAND, and HWMCA_REMOVE_CAPACITY_COMMAND command requests.

(26) HWMCA_CMD_COMMAND_FAILED

Explanation: A command call failed due to an internal error on the target console and a more specific failure code was not provided by the command.

Programmer response: Check the target console for details regarding the error.

(28) HWMCA_CMD_CBU_DISRUPTIVE_ OK

Explanation: The command request was successful but requires a system IML for the changes to take effect.

Programmer response: Perform a system IML.

(29) HWMCA_CMD_CBU_PARTIAL_ HW

Explanation: The command request was successful for the available hardware, but complete success for the command request could not be achieved (probably due to defective hardware).

(30) HWMCA_CMD_CBU_NO_SPARES

Explanation: The command request was unsuccessful because the required hardware was not available.

(31) HWMCA_CMD_CBU_TEMPORARY

Explanation: The command request was successful, but there was a problem updating the system SEEPROM so the new capacity will be lost at the next system IML.

(32) HWMCA_CMD_CBU_NOT_ ENABLED

Explanation: The command request failed because the CBU feature is not enabled for the target console.

(33) HWMCA_CMD_CBU_NOT_ AUTHORIZED

Explanation: The command request failed because the target console is not authorized for the requested command.

(34) HWMCA_CMD_CBU_FAILED

Explanation: The command request failed due to an internal error.

Programmer response: Check the target console for details about the failure.

(35) HWMCA_CMD_CBU_ALREADY_ ACTIVE

Explanation: The command request failed because there is already a previous CBU request in effect.

(36) HWMCA_CMD_CBU_INPROGRESS

Explanation: The command request cannot be performed at this time because another operation is being performed at this time.

Programmer response: The command request can be retried at a later time when the currently executing operation is complete.

(37) HWMCA_CMD_CBU_CPSAP_ SPLIT_CHG

Explanation: The command request cannot be performed at this time because the current CP/SAP allocation for the machine differs from what it was originally.

Programmer response: The command request can be retried at a later time after the CP/SAP split matches the original values for the machine.

(38) HWMCA_CMD_INVALID_ MACHINE_STATE

Explanation: The command request cannot be performed at this time because the target object is currently not in an appropriate state (i.e. it is not powered on).

Programmer response: The command request can be retried at a later time when the target object is in an appropriate state.

(39) HWMCA_CMD_NO_RECORDID

Explanation: The command request was unsuccessful because the specified a capacity record with the specified identifier does not exist.

Programmer response: The command request can be retried at a later time with an identifier for an existing capacity record.

(40) HWMCA_CMD_NO_SW_ MODEL

Explanation: The command request was unsuccessful because the specified software model is invalid for the target object.

Programmer response: The command request can be retried at a later time with a software model that is valid for the target object.

---

(41) HWMCA_CMD_NOT_ENOUGH_ RESOURCES

Explanation: The command request was unsuccessful because the request specifies more resources than are currently available on the target object.

Programmer response: The command request can be retried at a later time with a number of resources that are available on the target object.

---

(42) HWMCA_CMD_NOT_ENOUGH_ ACTIVE_RESOURCES

Explanation: The command request was unsuccessful because the request specifies more resources than are currently active on the target object.

Programmer response: The command request can be retried at a later time with a number of resources less than or equal to those that are currently active on the target object.

---

(43) HWMCA_CMD_ACT_LESS_ RESOURCES

Explanation: The command request for additional resources was unsuccessful because the request specifies a net decrease in the resources for the target object.

Programmer response: The command request can be retried at a later time with an increase of resources for the target object.

---

(44) HWMCA_CMD_DEACT_MORE_ RESOURCES

Explanation: The command request for a removal of resources was unsuccessful because the request specifies a net increase in the resources for the target object.

Programmer response: The command request can be retried at a later time with a decrease of resources for the target object.

---

(45) HWMCA_CMD_ACT_TYPE_ MISMATCH

Explanation: The command request was unsuccessful because the type value specified (real or test) was not valid for the type of capacity record being used.

Programmer response: The command request can be retried at a later time with a valid type value for the capacity record.

---

(46) HWMCA_CMD_API_NOT_ ALLOWED

Explanation: The command request was unsuccessful because the target Defined CPC was configured to not allow capacity changes via the Console Application Programming Interfaces.

Programmer response: In order to be successful, capacity changes must be allowed for the Console Application Programming Interfaces via the Customized API Settings task.

---

(47) HWMCA_CMD_CDU_IN_ PROGRESS

Explanation: The command request cannot be performed at this time because a concurrent driver upgrade operation is being performed at this time.

Programmer response: The command request can be retried at a later time when the currently executing operation is complete.

---

(48) HWMCA_CMD_MIRRORING_ RUNNING

Explanation: The command request cannot be performed at this time because a support element mirror operation is being performed at this time.

Programmer response: The command request can be retried at a later time when the currently executing operation is complete.

---

(49) HWMCA_CMD_COMMUNICATIONS_ NOT_ACTIVE

Explanation: The command request cannot be performed at this time because communication with the Defined CPC object is not active.

Programmer response: The command request can be retried at a later time when there is active communications with the Defined CPC object.

---

(50) HWMCA_CMD_RECORD_EXPIRED

Explanation: The command request was unsuccessful because the capacity record being used for the operation has expired.

Programmer response: The command request can be retried at a later time with after the capacity record expiration date has been extended or a different (not expired) capacity record can be used.

---

(51) HWMCA_CMD_PARTIAL_CAPACITY

Explanation: The command request was successful, but not all the resources requested could be made available. There are two cases that can result in this return code:

1. The request was issued with priority and not enough free resources existed to completely satisfy

the request. The additional resources to completely satisfy the request are pending and will be associated with the request as soon as they are no longer being used for other purposes.

2. At the time of the request there were sufficient resources to completely satisfy the request, but during while these resources were being brought online for the request something happened to cause the number of available resources to no longer be sufficient (i.e. a defective processor was detected).

---

(52) HWMCA_CMD_INVALID_REQUEST

Explanation: The command request was unsuccessful because it is not valid to be performed at this time due to one of the following conditions.

- An EDM (Enhanced Driver Maintenance) operation is currently in progress.
- The target of the request is configured to not allow capacity change API requests.
- The targeted system is not in the correct state to perform the request.

Programmer response: The command request can be retried at a later time after the system is in the correct state to allow the operation.

---

(53) HWMCA_CMD_ALREADY_ACTIVE

Explanation: The command request was unsuccessful because there is a different capacity record that is already active.

Programmer response: The command request can be retried at a later time after the currently active capacity record is no longer active or a different (the currently active) capacity record can be used.

---

(54) HWMCA_CMD_RESERVE_HELD

Explanation: The command request was unsuccessful because a task running on the targeted system has reserved control.

Programmer response: Wait until the task is complete and retry the command request.

---

(55) HWMCA_CMD_GENERAL_XML_PARSING_ERROR

Explanation: The command request was unsuccessful because the XML specified on the command is not well formed and could not be parsed properly.

Programmer response: Retry the command request with a well formed XML document.

---

(56) HWMCA_CMD_STP_NOT_ENABLED

Explanation: The command request was unsuccessful because STP is not enabled on the target system.

Programmer response: Verify that the request is targeted toward a system that has the STP feature enabled.

---

(57) HWMCA_CMD_STP_MUST_TARGET_CTS

Explanation: The command request failed because the targeted system is not the Current Time Server for the STP-only Coordinated Timing Network (CTN) or the system specified to become the Current Time Server does not match the targeted system.

Programmer response: Verify that the request is targeted toward the system that will be the Current Time Server after the command request and resubmit the request.

---

(58) HWMCA_CMD_STP_INVALID_CONFIG_SPECIFIED

Explanation: The command request was unsuccessful because the specified STP configuration is not valid.

Programmer response: Verify that the current configuration will support the command request and that the configuration specified in the command request is valid. Retry the command request with an appropriate STP configuration.

---

(59) HWMCA_CMD_STP_WRONG_CTN

Explanation: The command request was unsuccessful because the CTN ID of the current STP configuration is not valid for the request. This is most likely the result of the configuration changing between the time that the command request was created and the time the request was processed.

Programmer response: Verify that the current configuration will support the command request and that the configuration specified in the command request is valid. Then retry the command request with an appropriate STP configuration.

---

(60) HWMCA_CMD_STP_NOT_VALID_FOR_CTS

Explanation: The command request cannot be processed on the Current Time Server. Certain actions are not allowed on the Current Time Server because the result would be disruptive to the entire STP-only CTN.

Programmer response: Verify that the request is targeted toward the appropriate system.

---

(61) HWMCA_CMD_STP_IN_ETR_MIGRATION

Explanation: The command request was unsuccessful because the CPC is a member of an STP-only CTN that is migrating back to a Mixed CTN, which uses a Sysplex Timer. STP-related commands are not allowed until this procedure is complete.

Programmer response: Determine the appropriate action after the ETR migration is complete.

(62) HWMCA_CMD_STP_NODE_NOT_FOUND_IN_SYSTEM_LIST

Explanation: The command request was unsuccessful because the specified NodeName could not be converted into a NodeID.

Programmer response: The system referenced in the NodeName tag needs to be a Defined CPC object on the HMC console. Add the object to the HMC and retry the command request.

(63) HWMCA_CMD_STP_CTNID_TAG_ERROR

Explanation: The command request was unsuccessful because the CTN ID portion of the set STP configuration XML was not correct.

Programmer response: Retry the command request after verifying that the CTN ID information specified is in the proper format.

(64) HWMCA_CMD_STP_NODE_TAG_ERROR

Explanation: The command request was unsuccessful because the Preferred Time Server, Backup Time Server, or Arbiter portion of the set STP configuration XML was not correct.

Programmer response: Retry the command request after verifying that the node information specified is in the proper format.

(65) HWMCA_CMD_STP_CONFIG_TAG_NOT_FOUND

Explanation: The command request was unsuccessful because the STPConfiguration tag was not found in the set STP configuration XML.

Programmer response: Retry the command request specifying STPConfiguration as the outermost tag of the XML document.

(66) HWMCA_CMD_STP_ACTIVE_CTS_TAG_ERROR

Explanation: The command request was unsuccessful because the CurrentTimeServer portion of the set STP configuration XML was not correct.

Programmer response: Retry the command request verifying that the CurrentTimeServer information specified is in the proper format.

(67) HWMCA_CMD_STP_INITIALIZE_INCOMPLETE

Explanation: The command request was unsuccessful because the specified STP configuration cannot be set until initialization of the STP-only CTN is complete. The time zone and leap second values need to be set.

Programmer response: Manually set the time zone and/or leap second values via the Initialize Time button on the Network Configuration tab in the System (Sysplex) Time task and retry the command request.

(68) HWMCA_CMD_STP_INVALID_STP_ID

Explanation: The command request was unsuccessful because the STP ID specified on the command was not correct.

Programmer response: Retry the command request specifying an STP ID with 1-8 valid characters.

(69) HWMCA_CMD_STP_LINKS_ERROR

Explanation: The command request was unsuccessful because the communication links between the Preferred Time Server, Backup Time Server, and/or Arbiter systems in the STP-only CTN are not active.

Programmer response: Manually check the links between systems in the STP-only CTN with the roles of Preferred Time Server, Backup Time Server, and/or Arbiter or retry the command request with force, if applicable.

(70) HWMCA_CMD_STP_REQUIRES_FORCE_TO_CONFIGURE

Explanation: The command request was unsuccessful because of the current state of the targeted system. Verification of connections between systems with key roles in the STP-only CTN is done to ensure that the configuration will function properly. Once the connections are verified, the force parameter is required to ensure that the customer is not creating an island STP-only CTN.

Programmer response: Retry the set configuration command request with force, if applicable.

Data exchange and command API (REXX version) return codes

The following return codes are specific to the REXX version of the Data Exchange and Command APIs.

(1000) HWMCA_RX_INVALID_STEM_VAR

Explanation: The name of a REXX stem variable that was specified on one of the REXX API calls did not end with a ".".

Programmer response: Make sure the stem variable name ends with a ".".

Appendix D. APIs for Java (com.ibm.hwmca.api)

The purpose of the **com.ibm.hwmca.api** package is to allow Java™ applications, local or remote, the ability to exchange data related to the objects that the Console application manages. Specifically, this support allows other applications to request the Console application to:

- Query (Get/Get-Next) the attributes of objects
- Change (Set) certain attributes of objects
- Receive notification of significant events occurring to objects
- Generate enterprise-specific Simple Network Management Protocol traps for significant events occurring to objects.

The **com.ibm.hwmca.api** package uses the Simple Network Management Protocol (SNMP) as the transport mechanism. The attributes of objects can be queried/changed through the underlying SNMP Set, Get, Get-Next requests, while event notification is accomplished through the user of the enterprise-specific SNMP Trap message. The underlying SNMP protocol is encapsulated in several APIs in order to reduce the complexities for the application programmer. The **com.ibm.hwmca.api** package is part of the **HWMCAAPI.JAR** jar file which is located in the D:\TOOLKIT directory of the Hardware Management Console for Version 2.9.0 or earlier. The most up to date copy of this file is available on Resource Link at *http://www.ibm.com/servers/resourcelink*. Click **Services**, and then click **API**.

For documentation describing this Java application, see *Application Programming Interfaces for Java*.

Appendix E. Object Attribute Availability

Except for the attributes found in Table 1, it can be assumed that each object attribute described in Chapter 4, "Console application managed objects," on page 75 is valid for any level of object. The following table defines the required level of object for each attribute.

Table 1. Level of objects required on attributes

Attribute	Availability
Console Attributes	
Version	Available on consoles version 2.9.2 or later, or Support Element console 1.8.2 with the latest level of microcode applied.
Internet Protocol (IP) Addresses	Available on consoles version 2.10.0 or later
Engineering Change (EC)/Microcode Level (MCL)	Available on consoles version 2.10.0 or later
Defined CPC attributes	
Processor running time type	Available for Defined CPCs version 1.8.2 or later.
Processor running time	Available for Defined CPCs version 1.8.2 or later.
End timeslice if CPC enters a wait state	Available for Defined CPCs version 1.8.2 or later.
On/Off Capacity on Demand (On/Off CoD) installed	Available for Defined CPCs version 2.9.0 or later.
On/Off Capacity on Demand (On/Off CoD) activated	Available for Defined CPCs version 2.9.0 or later.
On/Off Capacity on Demand (On/Off CoD) enabled	Available for Defined CPCs version 2.9.0 or later.
On/Off Capacity on Demand (On/Off CoD) activation date	Available for Defined CPCs version 2.9.0 or later.
List of group profiles	Available for Defined CPCs version 2.9.2 or later.
Temporary capacity records	Available for Defined CPCs version 2.10.0 or later.
Permanent software model	Available for Defined CPCs version 2.10.0 or later.
Permanent plus billable software model	Available for Defined CPCs version 2.10.0 or later.
Permanent plus all temporary software model	Available for Defined CPCs version 2.10.0 or later.
Permanent MSU	Available for Defined CPCs version 2.10.0 or later.
Permanent plus billable MSU	Available for Defined CPCs version 2.10.0 or later.
Permanent plus all temporary MSU	Available for Defined CPCs version 2.10.0 or later.
General purpose processors	Available for Defined CPCs version 2.10.0 or later.
Service assist processors	Available for Defined CPCs version 2.10.0 or later.
Application Assist Processor (zAAP) processors	Available for Defined CPCs version 2.10.0 or later.
Integrated Facility for Linux (IFL) processors	Available for Defined CPCs version 2.10.0 or later.
Internal Coupling Facility (ICF) processors	Available for Defined CPCs version 2.10.0 or later.
Integrated Information Processors (zIIP) processors	Available for Defined CPCs version 2.10.0 or later.
Defective processors	Available for Defined CPCs version 2.10.0 or later.
Spare processors	Available for Defined CPCs version 2.10.0 or later.
Pending processors	Available for Defined CPCs version 2.10.0 or later.
Temporary capacity change allowed	Available for Defined CPCs version 2.10.0 or later.

Table 1. Level of objects required on attributes (continued)

Attribute	Availability
Version	Available for Defined CPCs version 2.9.2 or later or version 1.8.2 with the latest level of microcode applied.
Automatic switch enabled	Available for Defined CPCs version 1.7.3 or later.
Server Time Protocol (STP) configuration	Available for Defined CPCs, with STP enabled, 2.9.2 or later or 2.9.0 and 1.8.2 with the latest available microcode.
Pending General Purpose Processors	Available for Defined CPCs version 2.10.1 or later.
Pending Service Assist Processors	Available for Defined CPCs version 2.10.1 or later.
Pending Application Assist Processor (zAAP) Processors	Available for Defined CPCs version 2.10.1 or later.
Pending Integrated Facility for Linux (IFL) Processors	Available for Defined CPCs version 2.10.1 or later.
Pending Internal Coupling Facility (ICF) Processors	Available for Defined CPCs version 2.10.1 or later.
Pending Integrated Information Processors (zIIP) Processors	Available for Defined CPCs version 2.10.1 or later.
CPC Image attributes	
Initial Application Assist Processor processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Initial Application Assist Processor processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Minimum Application Assist Processor processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Maximum Application Assist Processor processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Current Application Assist Processor processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Current Application Assist Processor processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Initial Integrated Facility for Linux processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Initial Integrated Facility for Linux processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Minimum Integrated Facility for Linux processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Maximum Integrated Facility for Linux processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Current Integrated Facility for Linux processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Current Integrated Facility for Linux processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Initial Integrated Information Processors processing weight	Available for CPC Images running on Defined CPCs version 2.9.0 or later.
Initial Integrated Information Processors processing weight capped	Available for CPC Images running on Defined CPCs version 2.9.0 or later.
Minimum Integrated Information Processors processing weight	Available for CPC Images running on Defined CPCs version 2.9.0 or later.
Maximum Integrated Information Processors processing weight	Available for CPC Images running on Defined CPCs version 2.9.0 or later.

Table 1. Level of objects required on attributes (continued)

Attribute	Availability
Current Integrated Information Processors processing weight	Available for CPC Images running on Defined CPCs version 2.9.0 or later.
Current Integrated Information Processors processing weight capped	Available for CPC Images running on Defined CPCs version 2.9.0 or later.
Program Status Word (PSW) information	Available for CPC Images running on Defined CPCs version 2.10.0 or later.
IPL Token	Available for CPC Images running on Defined CPCs version 2.10.1 or later.
Group Profile capacity	Available for CPC Images running on Defined CPCs version 2.9.2 or later.
Coupling Facility attributes	
Initial Internal Coupling Facility processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Initial Internal Coupling Facility processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Minimum Internal Coupling Facility processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Maximum Internal Coupling Facility processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Current Internal Coupling Facility processing weight	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Current Internal Coupling Facility processing weight capped	Available for CPC Images running on Defined CPCs version 1.8.2 or later.
Group Profile attributes	
Capacity	Available for Defined CPCs version 2.9.2 or later.
Image Activation Profile attributes	
Group profile name	Available for Defined CPCs version 2.9.2 or later.
Number of dedicated Application Assist Processor (zAAP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved dedicated Application Assist Processors (zAAP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of dedicated Integrated Facility for Linux (IFL) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved dedicated Integrated Facility for Linux (IFL) processors	Available for Defined CPCs version 2.9.2 or later.
Number of dedicated Internal Coupling Facility (ICF) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved dedicated Internal Coupling Facility (ICF) processors	Available for Defined CPCs version 2.9.2 or later.
Number of dedicated Integrated Information Processors (zIIP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved dedicated Integrated Information Processors (zIIP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of shared general purpose processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved shared general purpose processors	Available for Defined CPCs version 2.9.2 or later.

Table 1. Level of objects required on attributes (continued)

Attribute	Availability
Number of shared Application Assist Processor (zAAP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved shared Application Assist Processor (zAAP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of shared Integrated Facility for Linux (IFL) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved shared Integrated Facility for Linux (IFL) processors	Available for Defined CPCs version 2.9.2 or later.
Number of shared Internal Coupling Facility (ICF) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved shared Internal Coupling Facility (ICF) processors	Available for Defined CPCs version 2.9.2 or later.
Number of shared Integrated Information Processors (zIIP) processors	Available for Defined CPCs version 2.9.2 or later.
Number of reserved shared Integrated Information Processors (zIIP) processors	Available for Defined CPCs version 2.9.2 or later.
Capacity Record attributes	
All attributes	Available for Defined CPCs version 2.10.0 or later.

Appendix F. XML descriptions

XML strings are used in several places throughout this document. This appendix defines the format of this XML and provides examples of each. It is important to keep in mind that while the XML returned and provided on input must be well formed and syntactically correct, this XML is not a complete document, but rather an XML fragment as illustrated with the examples. XML is used in the following areas of the Console Application Programming Interfaces.

Add capacity command

The input parameters for the HWMCA_ADD_CAPACITY_COMMAND on page 39 is specified using XML. Following is an example of this input; refer to the XML schema at the end of this appendix for the complete syntax definition of this XML.

```
<!--
 This example XML document illustrate the markup used to perform the following
 addition of temporary capacity :
 - change the general processors to a model A99
 - add 3 AAP processors
 - indicate the activation should take not priority
 - indicate the activation is not a "test", but a "real" activation
 -->
<add>
   <recordid>12345</recordid>
   <softwaremodel>A99</softwaremodel>
   <processorinfo>
      <type>AAP</type>
      <procstep>3</procstep>
   </processorinfo>
   <priority>false</priority>
   <test>false</test>
</add>
```

Remove capacity command

The input parameters for the HWMCA_REMOVE_CAPACITY_COMMAND on page 40 is specified using XML. Following is an example of this input; refer to the XML schema at the end of this appendix for the complete syntax definition of this XML.

```
<!--
 This example XML document illustrate the markup used to perform the following
 removal of temporary capacity :
 - change the general processors to a model A99
 - remove 3 AAP processors
 -->
<remove>
   <recordid>12345</recordid>
   <softwaremodel>A99</softwaremodel>
   <processorinfo>
      <type>AAP</type>
      <procstep>3</procstep>
   </processorinfo>
</remove>
```

Capacity record query

The output of a Get operation for a Capacity Record Object is an XML string that describes the record. Following is an example of this output; refer to the XML schema at the end of this appendix for the complete syntax definition of this XML.

```
<!--
 This example XML document illustrates the markup used to describe
 a capacity record that allows for:
   - 2 additional CPs with 2 additional speed steps
   - currently 1 CP with 1 speed step are active
 -->
<record xmlns="http://www.ibm.com/hwmca/api"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="file:///C:/Documents%20and%20Settings/Administrator/My%20Documents/
                            Test%20Java%20Code/xml ebod.xsd">
   <!-- Record id -->
   <recordid>12345</recordid>
   <!-- Offering type -->
   <recordtype>OOCOD</recordtype>
   <!-- Activation status -->
   <status>Real</status>
   <!-- Activation processor information. -->
   <processorinfo>
      <!-- Processor type. -->
      <type>CP</type>
      <!-- Processor count. -->
      <procstep>+1</procstep>
      <!-- Speed count. -->
      <speedstep>+1</speedstep>
      <!-- Maximum number of processors. -->
      <max>2</max>
      <!-- Remaining processor days. (-1 means unlimited) -->
      <remainingprocdays>99</remainingprocdays>
      <!-- Remaining MSU days. (-1 means unlimited) -->
      <remainingmsudays>99</remainingmsudays>
   </processorinfo>

   <processorinfo>
      <!-- Processor type. -->
      <type>AAP</type>
      <!-- Processor count. -->
      <procstep>+1</procstep>
      <!-- Maximum number of processors. -->
      <max>2</max>
      <!-- Remaining processor days. (-1 means unlimited) -->
      <remainingprocdays>99</remainingprocdays>
      <!-- Remaining MSU days. (-1 means unlimited) -->
      <remainingmsudays>99</remainingmsudays>
   </processorinfo>
   <!-- Activation start date (UTC). -->
   <activationstart>2006-07-04T11:11:11Z</activationstart>
   <!-- Activation expiration date. -->
   <activationexpiration>2006-08-04T11:11:11Z</activationexpiration>
   <!-- Record expiration date. -->
   <recordexpiration>2006-12-31T23:59:59Z</recordexpiration>
   <!-- Maximum real activation days. (-1 means unlimited) -->
   <maxrealdays>22</maxrealdays>
   <!-- Maximum test activation days. (-1 means unlimited) -->
   <maxtestdays>-1</maxtestdays>
   <!-- Remaining real activation days. (-1 means unlimited) -->
   <remainingrealdays>15</remainingrealdays>
   <!-- Remaining test activation days. (-1 means unlimited) -->
   <remainingtestdays>-1</remainingtestdays>
   <!-- Target information. -->
```

```
<target>
   <!-- Processor count. -->
   <procstep>-1</procstep>
   <!-- Speed count. -->
   <speedstep>-1</speedstep>
   <!-- Software model. -->
   <softwaremodel>A100</softwaremodel>
   <!-- Billable MSU cost. -->
   <billablemsucost>100</billablemsucost>
   <!-- Billable MSU delta -->
   <billablemsudelta>-10</billablemsudelta>
</target>
```

```
   <target>
      <!-- Processor count. -->
      <procstep>-1</procstep>
      <!-- Software model. -->
      <softwaremodel>A104</softwaremodel>
      <!-- Billable MSU cost. -->
      <billablemsucost>104</billablemsucost>
      <!-- Billable MSU delta -->
      <billablemsudelta>-6</billablemsudelta>
   </target>
   <target>
      <!-- Speed count. -->
      <speedstep>-1</speedstep>
      <!-- Software model. -->
      <softwaremodel>A105</softwaremodel>
      <!-- Real MSU cost. -->
      <billablemsucost>105</billablemsucost>
      <!-- Billable MSU delta -->
      <billablemsudelta>-5</billablemsudelta>
   </target>
   <target>
      <!-- Speed count. -->
      <speedstep>+1</speedstep>
      <!-- Software model. -->
      <softwaremodel>A115</softwaremodel>
      <!-- Billable MSU cost. -->
      <billablemsucost>115</billablemsucost>
      <!-- Billable MSU delta -->
      <billablemsudelta>5</billablemsudelta>
   </target>
   <target>
      <!-- Processor count. -->
      <procstep>+1</procstep>
      <!-- Software model. -->
      <softwaremodel>A116</softwaremodel>
      <!-- Billable MSU cost. -->
      <billablemsucost>116</billablemsucost>
      <!-- Billable MSU delta -->
      <billablemsudelta>6</billablemsudelta>
   </target>
   <target>
      <!-- Processor count. -->
      <procstep>+1</procstep>
      <!-- Speed count. -->
      <speedstep>+1</speedstep>
      <!-- Software model. -->
      <softwaremodel>A120</softwaremodel>
      <!-- Billable MSU cost. -->
      <billablemsucost>120</billablemsucost>
      <!-- Billable MSU delta -->
      <billablemsudelta>10</billablemsudelta>
   </target>
</record>
```

Engineering Change (EC)/Microcode Level (MCL) query

The output of a Get operation for the Engineering Change (EC)/Microcode Level (MCL) attribute of a Defined CPC or Console Object is an XML string. Following is an example of this output; refer to the XML schema at the end of this appendix for the complete syntax definition of this XML.

```
<!--
 This example XML document illustrates the markup used to describe
 EC/MCL information with one EC having only retrieved MCLs.
 -->
<sysinfo>
  <ec>
    <number>A12345</number>
    <partnumber>098765432</partnumber>
    <type>Console Application</type>
    <description>Console Application code</description>
    <mcl>
      <type>retrieved</type>
      <level>009</level>
      <lastupdate>2007-01-01T11:59:00Z</lastupdate>
    </mcl>
  </ec>
</sysinfo>
```

STP configuration information

The output of a Get operation for the STP Information attribute of a Defined CPC object as well as the input required for the HWMCA_SYSPLEX_TIME_SET_STP_CONFIG_COMMAND is an XML string. Following is an example of this data; refer to the XML schema at the end of this appendix for the complete syntax definition of this XML.

```
<STPConfiguration>
  <CTNID>
    <STPID>stpTst1</STPID>
  </CTNID>
  <Preferred>
    <NodeName>T25A</NodeName>
    <NodeID>
      <Type>002094</Type>
      <Model>S18</Model>
      <Manuf>IBM</Manuf>
      <PoManuf>00</PoManuf>
      <SeqNum>00000000T25A</SeqNum>
    </NodeID>
  </Preferred>
  <Backup>
    <NodeName>POLLUX</NodeName>
  </Backup>
  <Arbiter>
    <NodeID>
      <Type>002086</Type>
      <Model>A04</Model>
      <Manuf>IBM</Manuf>
      <PoManuf>00</PoManuf>
      <SeqNum>000000000T03</SeqNum>
    </NodeID>
  </Arbiter>
  <CurrentTimeServer>Backup</CurrentTimeServer>
</STPConfiguration>
```

XML schema

Following is the XML schema used to define the syntax of the XML used as input and output of the Console Application Programming Interfaces.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.ibm.com/hwmca/api"
            xmlns="http://www.ibm.com/hwmca/api"
            elementFormDefault="qualified">
<xsd:annotation>
    <xsd:documentation>
        This is the first version of the XML schema
        used to describe the XML that can be used as
        input or returned as output from the Console
        Application Programming Interfaces.  As future
        additions are made to this schema the intent is
        that it will remain compatible with the earlier
        version of the schema.
    </xsd:documentation>
</xsd:annotation>

<!--
Temporary Capacity related XML definitions.
-->



<!--
 Used to define the type for a processor.  The currently valid values for
 this element are:
    AAP - Application Assist Processor
    IFL - Integrated Facility for Linux processor
    ICF - Internal Coupling Facility processor
    IIP - Integrated Information Processors processor
    SAP - System Assist processor
 -->
<xsd:element name="type" type="xsd:string"/>
<!--
 Used to define the identifier for a capacity record.
 -->
<xsd:element name="recordid" type="xsd:string"/>
```

```
<!--
 Used to define the number of processor steps for a specific type of
 processor compared to some base point.
 -->
<xsd:element name="procstep" type="xsd:integer"/>
<!--
 Used to define the number of processor speed steps for a specific type of
 processor compared to some base point.
 -->
<xsd:element name="speedstep" type="xsd:integer"/>
<!--
 Used to define the software model for a specific processor configuration.
 -->
<xsd:element name="softwaremodel" type="xsd:string"/>

<!--
 Used to define if a capacity request has priority.
 -->
<xsd:element name="priority" type="xsd:boolean" default="false"/>
<!--
 Used to define if a capacity request is for test or real purposes.
 -->
<xsd:element name="test" type="xsd:boolean"/>
<!--
 The "processorinfo" element define information about a processor.
 The information that can be specified with this tag includes:
    - processor type (required)
    - number of processor steps
 -->
<xsd:element name="processorinfo">
   <xsd:complexType>
      <xsd:all>
         <xsd:element ref="type"/>
         <xsd:element ref="procstep" minOccurs="0"/>
      </xsd:all>
   </xsd:complexType>
</xsd:element>


<!--
 Used to define information specific to a capacity record.
 -->
<xsd:complexType name="recordinfo">
   <xsd:sequence>
      <xsd:element ref="recordid" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="softwaremodel" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="processorinfo" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<!--
 Used to define the parameters to use for the addition of temporary
 capacity.  The processor steps/software model specified in the element are
 specified with relation to the current configuration of the system.
 -->
<xsd:complexType name="addtype">
   <xsd:complexContent>
      <xsd:extension base="recordinfo">
         <xsd:sequence>
            <xsd:element ref="priority"/>
            <xsd:element ref="test"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

```
<xsd:element name="add" type="addtype"/>
<!--
 Used to define the parameters to use for removal of temporary
 capacity.  The processor steps/software model specified in the element are
 specified with relation to the current configuration of the system.
 -->
<xsd:element name="remove" type="recordinfo"/>

<!--
 Used to define the processor information for a capacity record.
-->
<xsd:complexType name="recordprocessors">
   <xsd:all>
      <!-- Processor type. -->
      <xsd:element ref="type"/>
      <!-- Processor count. -->
      <xsd:element ref="procstep" minOccurs="0"/>
      <!-- Speed count. -->
      <xsd:element ref="speedstep" minOccurs="0"/>
      <!-- Maximum number of processors. -->
      <xsd:element name="max" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <!-- Remaining processor days. (-1 means unlimited) -->
      <xsd:element name="remainingprocdays" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <!-- Remaining MSU days. (-1 means unlimited) -->
      <xsd:element name="remainingmsudays" type="xsd:integer" minOccurs="0" maxOccurs="1"/>
   </xsd:all>
</xsd:complexType>
<!--
 Used to define information for an activation/deactivation target of a capacity record.
-->
<xsd:complexType name="target">
    <xsd:all>
       <!-- Processor count. -->
       <xsd:element ref="procstep" minOccurs="0"/>
       <!-- Speed count. -->
       <xsd:element ref="speedstep" minOccurs="0"/>
       <!-- Software model. -->
       <xsd:element ref="softwaremodel" minOccurs="1"/>
       <!-- Billable MSU cost. -->
       <xsd:element name="billablemsucost" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
       <!-- Billable MSU delta -->
       <xsd:element name="billablemsudelta" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
    </xsd:all>
</xsd:complexType>
```

```
<!--
 Used to define a capacity record.
-->
<xsd:complexType name="recordtype">
   <xsd:sequence>
      <!-- Record id -->
      <xsd:element ref="recordid" minOccurs="1" maxOccurs="1"/>
      <!-- Offering type -->
      <xsd:element name="recordtype" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <!-- Activation status -->
      <xsd:element name="status" minOccurs="1" maxOccurs="1">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Test"/>
              <xsd:enumeration value="Real"/>
              <xsd:enumeration value="None"/>
              <xsd:enumeration value="Available"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <!-- Activation processor information. -->
      <xsd:element name="processorinfo" type="recordprocessors" minOccurs="1" maxOccurs="unbounded"/>
      <!-- Activation start date. -->
      <xsd:element name="activationstart" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
      <!-- Activation expiration date. -->
      <xsd:element name="activationexpiration" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
      <!-- Record expiration date. -->
      <xsd:element name="recordexpiration" type="xsd:dateTime" minOccurs="1" maxOccurs="1"/>
      <!-- Maximum real activation days. (-1 means unlimited) -->
      <xsd:element name="maxrealdays" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <!-- Maximum test activation days. (-1 means unlimited) -->
      <xsd:element name="maxtestdays" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <!-- Remaining real activation days. (-1 means unlimited) -->
      <xsd:element name="remainingrealdays" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <!-- Remaining test activation days. (-1 means unlimited) -->
      <xsd:element name="remainingtestdays" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <!-- Target information. -->
      <xsd:element name="target" type="target" minOccurs="1" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<xsd:element name="record" type="recordtype"/>

<!--
EC/MCL Information XML definitions.
-->
<!--
Used to define MicroCode Level (MCL) information.
-->
<xsd:complexType name="mcl">
  <xsd:sequence>
    <xsd:element name="type" minOccurs="1" maxOccurs="1">
       <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="retrieved"/>
            <xsd:enumeration value="installed"/>
            <xsd:enumeration value="activated"/>
            <xsd:enumeration value="accepted"/>
            <xsd:enumeration value="removed"/>
            <xsd:enumeration value="installableconcurrent"/>
            <xsd:enumeration value="removableconcurrent"/>
          </xsd:restriction>
       </xsd:simpleType>
    </xsd:element>
    <xsd:element name="level" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="lastupdate" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>


<!--
Used to define pending action information.
-->
<xsd:simpleType name="actiontype">
   <xsd:restriction base="xsd:string">
       <xsd:enumeration value="ChannelConfig"/>
       <xsd:enumeration value="CouplingFacilityReactivation"/>
       <xsd:enumeration value="PoweronResetTracking"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="actionactivation">
   <xsd:restriction base="xsd:string">
       <xsd:enumeration value="current"/>
       <xsd:enumeration value="next"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="action">
    <xsd:attribute name="type" use="required" type="actiontype"/>
    <xsd:attribute name="activation" use="required" type="actionactivation"/>
    <xsd:attribute name="pending" type="xsd:boolean" use="required"/>
</xsd:complexType>
<xsd:complexType name="pending">
    <xsd:sequence>
        <xsd:element name="action" type="action" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

```
<!--
Used to define Engineering Change (EC) information.
-->
<xsd:complexType name="ec">
  <xsd:sequence>
    <xsd:element name="number" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="partnumber" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="mcl" type="mcl" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>


<!--
Used to define Engineering Change (EC) / MicroCode Level (MCL) information.
-->
<xsd:element name="sysinfo">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="pending" type="pending" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ec" type="ec" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!--
Program Status Word (PSW) Information XML definitions.
-->
<!--
Used to define Program Status Word (PSW) information for a single processor.
-->
<xsd:complexType name="pswinfo">
  <xsd:sequence>
    <xsd:element name="psw" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="cpid" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<!--
Used to define CPC Image Program Status Word (PSW) information.
-->
<xsd:element name="imagepswinfo">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="pswinfo" type="pswinfo" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<!--
STP related XML definitions.
-->

<!--
Used to define the STP ID.
Limited to 8 alpha-numeric (plus '_' & '-') characters.
-->
<xsd:simpleType name="STPID">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="0"/>
    <xsd:maxLength value="8"/>
    <xsd:pattern value="([0-9a-zA-Z_\-])*"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
Used to define the ETR ID.
Limited to 2 numeric characters. (0-31)
-->
<xsd:simpleType name="ETRID">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="31"/>
  </xsd:restriction>
</xsd:simpleType>


<!--
Used to define the CTN ID.

For output, if not in an STP-only or Mixed CTN, the STP ID will be empty
-->
<xsd:complexType name="CTNID">
  <xsd:all>
    <!-- Used to define the STP ID of the CTN. -->
    <xsd:element name="STPID" type="STPID" minOccurs="1" maxOccurs="1"/>
    <!-- Used to define the ETR ID of the CTN. -->
    <xsd:element name="ETRID" type="ETRID" minOccurs="0" maxOccurs="1"/>
  </xsd:all>
</xsd:complexType>

<!--
Used to define the type of processor.
Limited to 6 characters.
-->
<xsd:simpleType name="Type">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="6"/>
    <xsd:maxLength value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!--
Used to define the model of the processor.
Limited to 3 characters.
-->
<xsd:simpleType name="Model">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="3"/>
    <xsd:maxLength value="3"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
Used to define the manufacturer of the processor.
Limited to 3 characters.
-->
<xsd:simpleType name="Manuf">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="3"/>
    <xsd:maxLength value="3"/>
  </xsd:restriction>
</xsd:simpleType>


<!--
Used to define the plant of manufacturer of the processor.
Limited to 2 characters.
-->
<xsd:simpleType name="PoManuf">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="2"/>
    <xsd:maxLength value="2"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
Used to define the sequence number of the processor.
Limited to 12 characters.
-->
<xsd:simpleType name="SeqNum">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="12"/>
    <xsd:maxLength value="12"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!--
Used to define the Node ID of the processor.
-->
<xsd:complexType name="NodeID">
  <xsd:all>
    <!-- Used to define the type of the CPC. -->
    <xsd:element name="Type" type="Type" minOccurs="1" maxOccurs="1"/>
    <!-- Used to define the model of the CPC. -->
    <xsd:element name="Model" type="Model" minOccurs="1" maxOccurs="1"/>
    <!-- Used to define the manufacturer of the CPC. -->
    <xsd:element name="Manuf" type="Manuf" minOccurs="1" maxOccurs="1"/>
    <!-- Used to define the plant of manufacturer of the CPC. -->
    <xsd:element name="PoManuf" type="PoManuf" minOccurs="1" maxOccurs="1"/>
    <!-- Used to define the sequence number of the CPC. -->
    <xsd:element name="SeqNum" type="SeqNum" minOccurs="1" maxOccurs="1"/>
  </xsd:all>
</xsd:complexType>

<!--
Used to define the name of the processor.
Limited to 8 characters.
-->
<xsd:simpleType name="NodeName">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="8"/>
  </xsd:restriction>
</xsd:simpleType>


<!--
Used to define the Node of the processor.

For input, there must be at least one of these specified.
If both are specified, the NodeID will take precedence.

For output, at least one is guaranteed.
-->
<xsd:complexType name="NodeDef">
  <xsd:all>
    <!-- Used to define Node ID to identify the CPC. -->
    <xsd:element name="NodeID" type="NodeID" minOccurs="0" maxOccurs="1"/>
    <!-- Used to define Node Name to identify the CPC. -->
    <xsd:element name="NodeName" type="NodeName" minOccurs="0" maxOccurs="1"/>
  </xsd:all>
</xsd:complexType>

<!--
Used to define the Current Time Server of the CTN.
Limited to "Preferred" and "Backup".
-->
<xsd:simpleType name="CurrentTimeServer">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Preferred"/>
    <xsd:enumeration value="Backup"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!--
Used to define STP configuration information.

For input:
Only STP-only configurations are allowed to be configured.
The CTNID is not required unless changing the CTNID of the STP-only CTN.
The Preferred node definition is required.
-->
<xsd:element name="STPConfiguration">
  <xsd:complexType>
    <xsd:all>
      <!-- Used to define new CTN ID.  -->
      <xsd:element name="CTNID" type="CTNID" minOccurs="0" maxOccurs="1"/>
      <!-- Used to define new Preferred CPC.  -->
      <xsd:element name="Preferred" type="NodeDef" minOccurs="0" maxOccurs="1"/>
      <!-- Used to define new Backup CPC.  -->
      <xsd:element name="Backup" type="NodeDef" minOccurs="0" maxOccurs="1"/>
      <!-- Used to define new Arbiter CPC.  -->
      <xsd:element name="Arbiter" type="NodeDef" minOccurs="0" maxOccurs="1"/>
      <!-- Used to define which system is going to be the Current Time Server CPC.  -->
      <xsd:element name="CurrentTimeServer" type="CurrentTimeServer" minOccurs="1" maxOccurs="1"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Appendix G. Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 USA

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION " AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linux Torvalds in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Windows is a trademark or registered trademark of Microsoft Corporation.

Other product and service names might be trademarks of IBM or other companies.

Electronic emission notices

The following statements apply to this IBM product. The statement for other IBM products intended for use with this product will appear in their accompanying manuals.

Federal Communications Commission (FCC) Statement

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions contained in the installation manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Properly shielded and grounded cables and connectors must be used in order to meet FCC emission limits. IBM is not responsible for any radio or television interference caused by using other than recommended cables and connectors, by installation or use of this equipment other than as specified in

the installation manual, or by any other unauthorized changes or modifications to this equipment. Unauthorized changes or modifications could void the user's authority to operate the equipment.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Canadian Department of Communications Compliance Statement

This Class A digital apparatus complies with Canadian ICES-003.

Avis de conformlté aux normes du ministère des Communications du Canada

Cet appareil numérique de la classe A est conform à la norme NMB-003 du Canada.

European Union (EU) Electromagnetic Compatibility Directive

This product is in conformity with the protection requirements of EU Council Directive 2004/108/EC on the approximation of the laws of the Member States relating to electromagnetic compatibility. IBM cannot accept responsibility for any failure to satisfy the protection requirements resulting from a non-recommended modification of the product, including the fitting of non-IBM option cards.

This product has been tested and found to comply with the limits for Class A Information Technology Equipment according to European Standard EN 55022. The limits for Class equipment were derived for commercial and industrial environments to provide reasonable protection against interference with licensed communication equipment.

Warning: This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user may be required to take adequate measures.

European Community contact:
IBM Deutschland GmbH
Technical Regulations, Department M372
IBM-Allee 1, 71139 Ehningen, Germany
Telephone: 0049 (0) 7032 15-2941
email: lugi@de.ibm.com

EC Declaration of Conformity (In German)

Deutschsprachiger EU Hinweis: Hinweis für Geräte der Klasse A EU-Richtlinie zur Elektromagnetischen Verträglichkeit

Dieses Produkt entspricht den Schutzanforderungen der EU-Richtlinie 89/336/EWG zur Angleichung der Rechtsvorschriften über die elektromagnetische Verträglichkeit in den EU-Mitgliedsstaaten und hält die Grenzwerte der EN 55022 Klasse A ein.

Um dieses sicherzustellen, sind die Geräte wie in den Handbüchern beschrieben zu installieren und zu betreiben. Des Weiteren dürfen auch nur von der IBM empfohlene Kabel angeschlossen werden. IBM übernimmt keine Verantwortung für die Einhaltung der Schutzanforderungen, wenn das Produkt ohne Zustimmung der IBM verändert bzw. wenn Erweiterungskomponenten von Fremdherstellern ohne Empfehlung der IBM gesteckt/eingebaut werden.

EN 55022 Klasse A Geräte müssen mit folgendem Warnhinweis versehen werden:

"Warnung: Dieses ist eine Einrichtung der Klasse A. Diese Einrichtung kann im Wohnbereich Funk-Störungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen zu ergreifen und dafür aufzukommen."

Deutschland: Einhaltung des Gesetzes über die elektromagnetische Verträglichkeit von Geräten

Dieses Produkt entspricht dem “Gesetz über die elektromagnetische Verträglichkeit von Geräten (EMVG)“. Dies ist die Umsetzung der EU-Richtlinie 89/336/EWG in der Bundesrepublik Deutschland.

Zulassungsbescheinigung laut dem Deutschen Gesetz über die elektromagnetische Verträglichkeit von Geräten (EMVG) vom 18. September 1998 (bzw. der EMC EG Richtlinie 89/336) für Geräte der Klasse A.

Dieses Gerät ist berechtigt, in Übereinstimmung mit dem Deutschen EMVG das EG-Konformitätszeichen - CE - zu führen.

Verantwortlich für die Konformitätserklärung nach Paragraf 5 des EMVG ist die IBM Deutschland GmbH, 70548 Stuttgart.

Informationen in Hinsicht EMVG Paragraf 4 Abs. (1) 4:

Das Gerät erfüllt die Schutzanforderungen nach EN 55024 und EN 55022 Klasse A.

update: 2004/12/07

People's Republic of China Class A Compliance Statement

This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user may need to perform practical actions.

声　明

此为A级产品，在生活环境中，
该产品可能会造成无线电干扰。
在这种情况下，可能需要用户对其
干扰采取切实可行的措施。

Japan Class A Compliance Statement

This is a Class A product based on the standard of the VCCI Council. If this equipment is used in a domestic environment, radio interference may occur, in which case, the user may be required to take corrective actions.

この装置は，クラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。　VCCI－A

Korean Class A Compliance Statement

이 기기는 업무용(A급)으로 전자파적합등록을 한 기기이오니
판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의
지역에서 사용하는 것을 목적으로 합니다.

Taiwan Class A Compliance Statement

Warning: This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user will be required to take adequate measures.

警告使用者:
這是甲類的資訊產品，在
居住的環境中使用時，可
能會造成射頻干擾，在這
種情況下，使用者會被要
求採取某些適當的對策。

台灣IBM 產品服務聯絡方式：
台灣國際商業機器股份有限公司
台北市松仁路7號3樓
電話：0800-016-888

Glossary

This glossary includes terms and definitions from:

- The Dictionary of Computing, SC20-1699.
- The American National Standard Dictionary for Information Systems, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The Information Technology Vocabulary, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The following cross-references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

See. This refers the reader to multiple-word terms in which this term appears.

See also. This refers the reader to terms that have a related, buy not synonymous meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in the glossary.

action One of the defined tasks that an application performs. Actions modify the properties of an object or manipulate the object in some way.

active window
The window that users are currently interacting with. This is the window that receives keyboard input.

address
A value that identifies a register, a particular part of storage, a data source, or a data sink. The value is represented by one or more characters. (T)

To refer to a device or an item of data by its address. (I) (A)

The location in the storage of a computer where data is stored.

In data communication, the unique code assigned to each device or workstation connected to a network.

The identifier of a location, source, or destination.

alert A unit of information, usually indicating the loss of a system resource, passed from one machine or program to a host to signal an error.

An error message sent to the system services control point (SSCP) at the host system.

allocate
To assign a resource, such as a disk, to perform a task.

application
The use to which an information processing system is put, for example, a payroll application, an airline reservation application, a network application.

A collection of software components used to perform specific types of work on a computer.

application program
A program that is specific to the solution of an application problem. (T)

A program written for or by a user that applies to the user's work, such as a program that does inventory control or payroll.

A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

asynchronous
Pertaining to two or more processes that

do not depend upon the occurrence of specific events such as common timing signals. (T)

Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions. Contrast with *synchronous*.

bit Either of the digits 0 or 1 when used in the binary numeration system. (T) See also *byte*.

block A string of data elements recorded or transmitted as a unit. The element may be characters, words, or physical records. (T)

buffer A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (A)

To allocate and schedule the use of buffers. (A)

A portion of storage used to hold input or output data temporarily.

byte A string that consists of a number of bits, treated as a unit, and representing a character. (T)

A binary character operated upon as a unit and usually shorter than a computer word. (A)

A string that consists of a particular number of bits, usually eight, that is treated as a unit, and that represents a character.

group of eight adjacent binary digits that represent one extended binary-coded decimal interchange code (EBCDIC) character.

central processor (CP)
The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

central processor complex (CPC)
The boundaries of a system, exclusive of I/O control units and devices, that can be controlled by a single operating system. A CPC consists of main storage, one or more central processor units, time-of-day clocks, and channels, which are or can be placed in a single configuration. A CPC also includes channel subsystems, service processors, and expanded storage where installed.

change
An alteration (addition, deletion, or modification) of one or more information system components, of one of the following types: hardware (may include internal code), or software (system or application). The term *change* also refers to an SNA/File Services data object containing internal code, internal code customizing data, software, software customizing data, applications data, procedures, or documentation.

channel
A path along which signals can be sent, for example, input/output channel.

The system element that controls one channel path, whose mode of operation depends on the type of hardware to which it is attached.

command
A character string from a source external to a system that represents a request for system action.

A request from a terminal for performance of an operation or execution of a program.

A value sent on an I/O interface from a channel to a control unit that specifies the operation to be performed.

command retry
A channel and control unit procedure that causes a command to be retried without requiring an I/O interrupt.

component
Hardware or software that is part of a functional unit.

A functional part of an operating system; for example, the scheduler or supervisor.

configuration
The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (I) (A)

configure
To describe to the system the devices and optional features installed on the system.

console
A logical device used for communication between the user and the system. (A)

coupling facility
A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

coupling facility channel
A high bandwidth fiber optic channel that provides the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

CPC Image
The set of CPC resources that support a single control program.

default
Pertaining to an attribute, value, or option that is assumed when none is explicitly specified. (I)

degraded
Pertaining to a mode of operation in which the system operates with some resources not available.

element
A major part of a component (for example, the buffer control element) or a major part of a system (for example, the system control element).

enter An action that submits information to the computer for processing.

error The smallest detectable anomaly or exception that can occur in an information system. Errors may be caused by hardware, software, internal code, media, or external causes, for example, people or environmental abnormalities.

error message
An indication that an error has been detected.

ESA Enterprise Systems Architecture.

ESA/390
Enterprise Systems Architecture/390.

event An occurrence or happening.

An occurrence of significance to a task; for example, the completion of an asynchronous operation, such as an input/output operation.

exchange
To remove an item and put another in its place; for example, to remove a field-replaceable unit (FRU) and install another of the same type.

facility
An operational capability, or the means for providing such a capability. (T)

A service provided by an operating system for a particular purpose; for example, the checkpoint/restart facility.

failure
An uncorrected hardware error. Contrast with *error* and *fault*.

Note: Failures are either recoverable or not recoverable by the software or the operator. The operator is always notified when failures occur. Usually, system recovery occurs through a hardware reconfiguration. If this is not possible, recovery requires a repair of the failed hardware.

fault An accidental condition that causes a functional unit to fail to perform its required function. (I) (A) Contrast with *error* and *failure*.

feature
A particular part of an IBM product that can be ordered separately.

function key
In computer graphics, a button or switch that may be operated to send a signal to the computer program controlling the display. (T)

A key that, when pressed, performs a specified set of operations.

guest In interpretive execution mode, the interpreted or virtual machine as opposed to the real machine (the host).

hard disk
A rigid disk used in a hard disk drive.

hardware
The equipment, as opposed to the programs, of a computer system.

Hardware Management Console
A console used to monitor and control hardware such as the System/390® and zSeries 900 processors.

Hardware Management Console Application (HWMCA)
A user customized, object-oriented graphical user interface that provides a single point of control for the system's hardware elements. The HWMCA provides aggregated and individual real-time system status via colors, consolidated hardware messages support, consolidated operating system messages support, consolidated service support, and hardware commands targeted at a single system, multiple systems, or a group of systems.

hexadecimal
Pertaining to a selection, choice, or condition that has 16 possible values or states. (I)

Pertaining to a fixed-radix numeration system, with radix of 16. (I)

Pertaining to a numbering system with base of 16; valid numbers use the digits 0–9 and characters A–F, where A represents 10 and F represents 15.

host The primary or controlling computer in a multiple computer installation.

host system
The primary or controlling computer in a network.

IBM program support representative
An IBM service representative who preforms maintenance services for IBM Licensed Internal Code.

icon A pictorial representation of an object or a selection choice. Icons can represent objects that users want to work on or actions that users want to perform. A unique icon also represents the application when it is minimized.

identifier (ID)
One or more characters used to identify or name a data element and possibly to indicate certain properties of that data element. (T)

A sequence of bits or characters that identifies a program, device, or system to another program, device, or system.

initial machine load (IML)
A procedure that prepares a device for use.

initial program load (IPL)
The initialization procedure that causes an operating system to commence operation.

The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction.

The process of loading system programs and preparing a system to run jobs.

initialization
The operations required for setting a device to a starting state, before the use of a data medium, or before implementation of a process. (T)

Preparation of a system, device, or program for operation.

To set counters, switches, addresses, latches, or storage contents to zero or to other starting values at the beginning of, or at the prescribed points in, a computer program or process.

initialize
To prepare for use.

input Data to be processed.

input/output (I/O)
Pertaining to a device whose parts can perform an input process and an output process at the same time. (I)

Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process.

Pertaining to input, output, or both.

input/output configuration data set (IOCDS)
The data set that contains an I/O configuration definition built by the I/O configuration program (IOCP).

interface
A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate. The concept includes the specification of the connection of two devices having different functions. (T)

Hardware, software, or both, that links systems, programs, or devices.

logical partition
A subset of the processor hardware that is defined to support the operation of a system control program (SCP). See also *logically partitioned (LPAR) mode*.

logically partitioned (LPAR) mode
A central processor complex (CPC) power-on reset mode that enables use of the PR/SM feature and allows an operator to allocate CPC hardware resources (including central processors, central storage, expanded storage, and channel paths) among logical partitions.

loop A sequence of instructions that is to be executed iteratively (T)

A closed unidirectional signal path connecting input/output devices to a system.

message
Information sent to a user from a program or another user.

mode A method of operation.

MVS Multiple Virtual Storage.

MVS system
An MVS image together with its associated hardware, which collectively are often referred to simply as a system, or MVS system.

network
An arrangement of nodes and connecting branches. (T)

A configuration of data processing devices and software connected for information exchange.

node In a network, the point at which one or more functional units connect channels or data circuits. (I)

In a network topology, the point at the end of a branch. (T)

In SNA, an endpoint of a link, or a junction common to two or more links in a network.

operand
Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

operate
To do a defined action, such as adding or comparing, performed on one or more data items.

operating system
Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. (T)

operations command facility (OCF)
A facility of the central processor complex that accepts and processes operations management commands.

output
Data that has been processed.

panel A display of a list of available functions for selection by the operator.

parameter
A variable that is given a constant value for a specified application and that may denote the application. (I) (A)

An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted.

Data passed between programs or procedures.

partition
See *logical partition.*

password
A value used in authentication or a value used to establish membership in a set of people having specific privileges.

A unique string of characters known to the computer system, and to a user who must specify it to gain full or limited access to a system and to the information stored within it.

pointer
The symbol displayed on the screen that is moved by a pointing device, such as a mouse. It is used to point at the objects and actions users want to select.

power-on reset (POR)
A function that reinitializes all the hardware in the system and loads the internal code that enables the machine to load and run an operating system. This function is intended as a recovery function.

problem
An error condition resulting in a loss of availability of a system resource to an end user.

processing weight
A relative value, ranging from 1 to 999, assigned to a partition of a system running in logically partitioned mode. It is used to calculate the share of processing resource to be allocated to that partition.

processor
In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (T)

The functional unit that interprets and executes instructions.

The boundaries of a system, exclusive of I/O control units and devices, that can be controlled by a single operating system. A processor consists of main storage, one or more central processors, time-of-day clocks, and channels, which are, or can be, placed in a single configuration. A processor also includes channel subsystems, and expanded storage where installed.

profile
A description of the characteristics of an entity to which access is controlled.

Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

program
Sequence of instructions for a computer. A program interacts and relies on either the hardware or other programs.

program status word (PSW)
An area in storage used to indicate the sequence in which instructions are executed, and to hold and indicate the status of the computer system.

protocol
A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (I)

In SNA, the meanings of and the sequencing rules for requests and responses used for managing the network, transferring data, and synchronizing the states of network components.

A specification for the format and relative timing of information exchanged between communicating parties.

push button
A rounded-corner rectangle with text inside. Actions occur immediately when the push button is selected.

quiesce
To bring a system or a device to a halt by rejecting new requests for work.

In a VTAM® application program, for one node to stop another node from sending synchronous-flow messages.

register
A part of internal storage having a specified storage capacity and usually intended for a specific purpose. (T)

remote
Physically distant. Pertains to a computer or device that is connected to another computer or device over a communication line. Contrast with local.

return code
A code used to influence the execution of succeeding instructions. (A)

A value returned to a program to indicate the results of an operation requested by that program.

S/370 System/370 mode.

screen The physical surface of a workstation on which information is shown to users.

single point of control
The characteristic a sysplex displays when you can accomplish a given set of tasks from a single workstation, even if you need multiple IBM and vendor products to accomplish that particular set of tasks.

single system image
The characteristic a product displays when multiple images of the product can be viewed and managed as one image.

storage
A functional unit into which data can be placed, in which they can be retained, and from which they can be retrieved. (T)

The action of placing data into a storage device. (I) (A)

support element
An internal control element of a processor that assists in many of the processor operational functions.

A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

synchronous
Pertaining to two or more processes that depend o the occurrence of specific events, such as common timing signals. (T)

Occurring with a regular or predictable time relationship. Contrast with *asynchronous*.

system
Comprises the processor complex and all attached and configured I/O and communication devices.

system resource
Hardware, such as a central processor, I/O devices, channel paths, software programs, or other components that contribute to system operation.

systems network architecture (SNA)
SNA specifies how products connect and communicate with one another in a network. SNA is a design for a total data communication system, encompassing every part of the communication network from the user's application program at the central site to the terminal at a remote location possibly hundreds of miles away. SNA itself is not a system, but an architecture—a specified set of formats and protocols to guide the design of machines and programs. The purpose of SNA is to define uniform formats and protocols for data communication networks, which have traditionally been characterized by programs, devices, and communication techniques that often were not compatible.

title bar
The area at the top of each window that contains the window title and system menu icon. When appropriate, it also contains the minimize, maximize, and restore icons.

token-ring network
A ring network that allows unidirectional data transmission between data stations, by a token passing procedure, such that the transmitted data return to the transmitting station. (T)

Note: The IBM Token-Ring Network is a baseband LAN with a star-wired ring topology that passes tokens from network adapter to network adapter.

trap An unprogrammed conditional jump to a specified address that is automatically activated by hardware. A recording is made of the location from which the jump occurred. (I)

A forced Licensed Internal Code branch, usually to an error routine.

user interface
Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

variable
In programming languages, a language object that may take different values, one at a time. The values of a variable are usually restricted to a certain data type. (I)

A quantity that can assume any of a given set of values. (A)

A name used to represent a data item whose value can be changed while the program is running.

window
An area of the screen with visible boundaries through which information is displayed. A window can be smaller than or equal in size to the screen. Windows can overlap on the screen and give the appearance of one window being on top of another

A choice in the action bar of some applications. Users select it to arrange the display of several windows or to change the active window.

A choice in the action bar of multiple-document interface applications.

A choice in an action bar that allows the user to arrange the display of all open windows and to change the active window.

A choice in the action bar of multiple-document interface applications that allows a user to arrange the display of all open windows and to change the active window.

work area
An area reserved for temporary storage of data to be operated on.

workstation
A functional unit at which a user works. A workstation often has some processing capability. (T)

A terminal or microcomputer, usually one that is connected to a mainframe or network, at which a user can perform applications.

Printed in USA

SB10-7030-16