SMASH Proxy

Installation and User's Guide

Version 1.0

SMASH Proxy

Installation and User's Guide

Version 1.0

Note
Before using this information and the product it supports, read the information in "Notices," on page 113.

First Edition (June 2006)

This edition applies to Version 1, Release 1, of IBM SMASH Proxy (product number 40K1-521) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

Preface

IBM® Systems Management Architecture for Server Hardware (SMASH) Proxy provides a command line interface based on the Distributed Management Task Force (DMTF) SMASH Command Line Protocol (CLP)® specification that allows a user to discover and manage IBM BladeCenter® chassis in his network from a single management station.

About this guide

The purpose of this guide is to provide users of the SMASH Proxy:

- An overview of SMASH, its history, features, and components and its relation to the IBM SMASH Proxy product.
- A detailed overview of the SMASH Proxy, including installation, configuration, functionality, accessibility, features, and components.

Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for the SMASH Proxy

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in the IBM SMASH Proxy. These features support:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers

Note: The SMASH Proxy Information Center and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard shortcuts

The following SMASH Proxy command-line editing shortcuts are available for command entry:

Table 1. Basic command editing

Shortcut	Action
Ctrl-B	Move back one character.
Ctrl-F	Move forward one character.
Backspace	Delete the character to the left of the cursor.
Ctrl-D	Delete the character underneath the cursor.

Table 2. Cursor movement

Shortcut	Action
Ctrl-A	Move to the start of the line.
Ctrl-E	Move to the end of the line.
Ctrl-L	Clear the screen, reprinting the current line at the top.

Table 3. Deletion

Shortcut	Action
Ctrl-K	Delete text from the current cursor position to the end of the line.
Ctrl-W	Delete the whole line.

IBM and accessibility

See the IBM Accessibility Center for more information about the commitment that IBM has to accessibility.

Who should read this guide

This guide is for system programmers and users working in an IBM BladeCenter environment and using SMASH Proxy to manage all BladeCenter chassis. It is a good starting point for a basic understanding of the product.

Chapter 1. Introduction

Large service providers require large quantities of equipment from one or more vendors to meet their computing needs. Most of this equipment must be managed actively. Examples of management activities include starting, configuration, monitoring operational parameters and alarms, executing firmware upgrades, and so on. Typically, management interfaces and procedures to perform such activities are standardized across an equipment vendor's portfolio. However, standardization of management interfaces and procedures across different vendors or even across different product portfolios offered by the same vendor is rare. As a result of the crucial need for this type of industry-wide standardization, the DMTF created a Web Based Enterprise Management (WBEM - pronounced ″web-em″) architecture, based on a Common Information Model (CIM).

The CIM strives to define the manageable properties (for example, model number, serial number) and behavior (for example, reset, power on, power off) of entities, be they logical (for example, a software installation service) or physical (for example, a blade, a chassis), through standardized templates. The templates are described formally using a modeling technique and are generally accepted by the industry as being the most common representations of those manageable entities.

Having defined models for common manageable entities, a standardized method for accessing the properties (or invoking the behavior) of these entities was required in order to effectively manage them. This implied that an agreed-upon method had to be devised for packaging the request for accessing properties or invoking behaviors, transporting the request across to the managed entity, executing the request on the managed entity, collecting the results of the execution and transporting it back to the requesting entity. In essence, an architecture had to be defined for interfacing system management software with the managed entities. This architecture is the WBEM.

In the area of servers, the DMTF has been working to define common models for standalone servers, rack-mounted servers, server blades mounted on a chassis, and so on. The group within the DMTF responsible for addressing the standardization of server management is the server management workgroup. Although the WBEM architecture addressed the need for interfacing GUI-based system management programs (for example, IBM Director or HP OpenView) to the managed entities (for example, servers), a portion of the server management industry felt the need to provide a standardized command line interface (CLI) for managing servers. A standardization effort in this area, by the server management workgroup, produced the SMASH specifications.

Through the SMASH CLP, you can run server management operations from a console by keying in standardized commands and options (as opposed to working through a GUI). The standard also enables you to write scripts that can work across equipment provided by different vendors.

SMASH includes the following components:

- An architectural framework
- A specification for a CLP
- A grammar for creating unique addresses for CIM objects so that they can serve as CLP command targets

- A set of approximately 30 CIM profiles and subprofiles that model server hardware
- A set of mappings of CLP commands to profile and subprofile CIM elements

The focus of SMASH is to enable the management of server resources in a standard manner regardless of server product type, vendor, or operating system state.

Having been heavily involved in the standardization process, IBM endeavors to be among the first implementers of this standard for SMASH as it applies to BladeCenter management using the SMASH Proxy.

Chapter 2. SMASH: An overview

The information in the SMASH overview section details the general features and components of SMASH and an introduction to addressing in a SMASH environment. It includes discussion of the following topics:

- "SMASH CLP architecture overview"
- "SMASH CLP architectural model"
- "Addressing managed elements" on page 4

SMASH CLP architecture overview

The goal of the architecture is to describe server management in abstract terms regardless of server type, topology, and framework. It demonstrates that SMASH is useful in a variety of server implementations, spans the spectrum from small stand-alone servers to large partitionable servers, and encompasses topologies such as blade servers and racks as well as unique segments such as industry-standard servers, telecommunications, and mission-critical high-end servers.

SMASH CLP architectural model

The SMASH CLP architecture has three components:

- Client
- Management server
- Managed element or elements

Figure 1 on page 4 displays a model of these components.

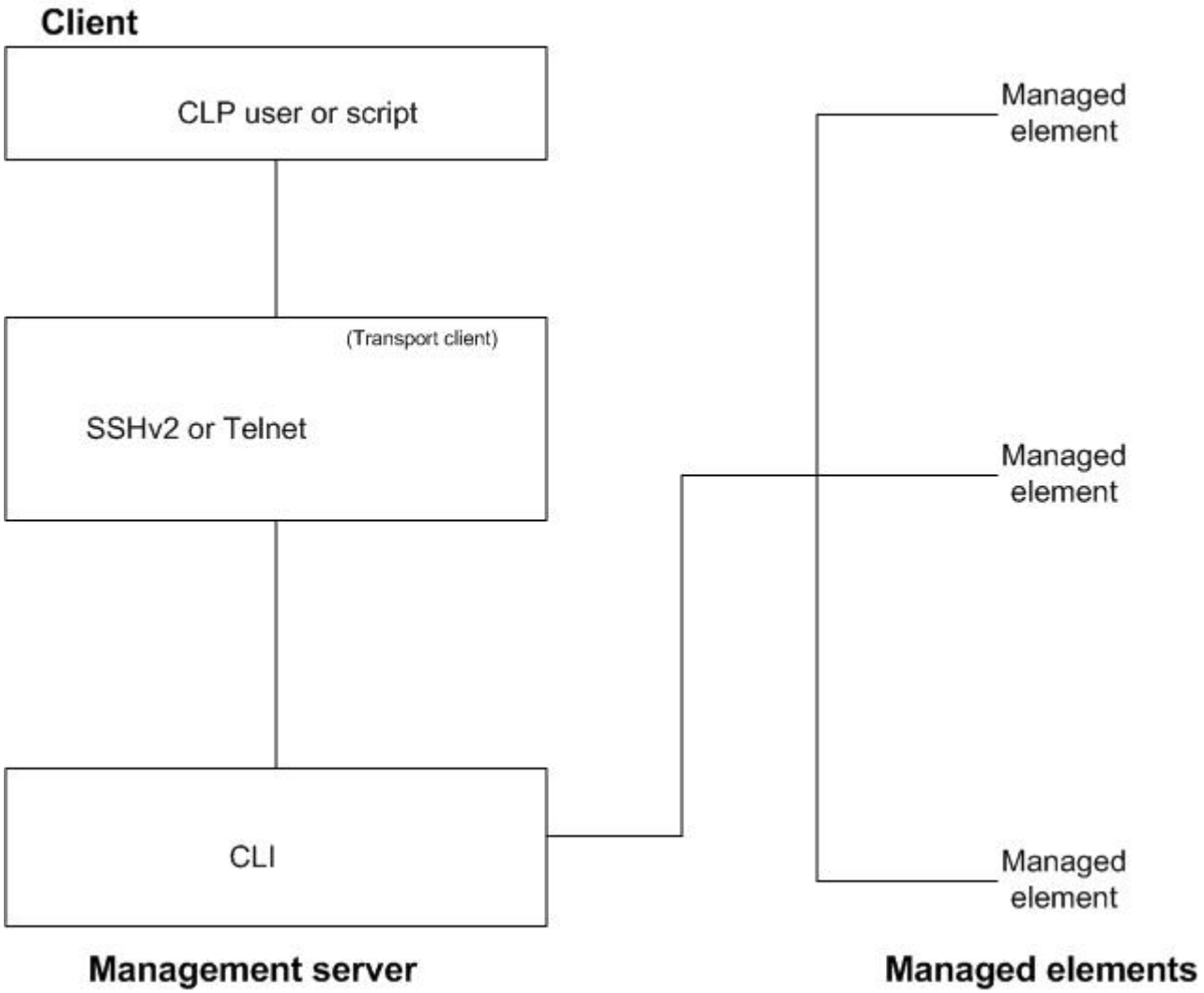

Figure 1. SMASH CLP architectural model

In the *Client* section, the *CLP user* represents the operator or program utilizing the CLI. Transport of information between the client and the *Management server* is through either Telnet or Secure Shell (SSHv2) for secure, encrypted transmissions.

The *managed elements* (*ME*s) are the objects, targets, components, resources, collections, or logical entities within a managed system that the operations manipulate, such as its blades or switches.

Addressing managed elements

In Figure 1, the CLI command string is directed towards a particular managed element specified through a hierarchy of managed elements or the directory path. This section takes a closer look at the concept of hierarchical addressing of target managed elements, *Server Management Managed Element (SM ME) addressing* , which is a user-friendly way to address objects in a managed system.

Figure 2 on page 5 demonstrates the concept of hierarchy among manageable elements. According to this figure:

- Sensors are managed logical elements that reside on blades.

- Blades are containers for sensors.
- A chassis is a container for blades and cooling fans.
- An administrative domain is a logical entity that is the access point for managing a chassis.

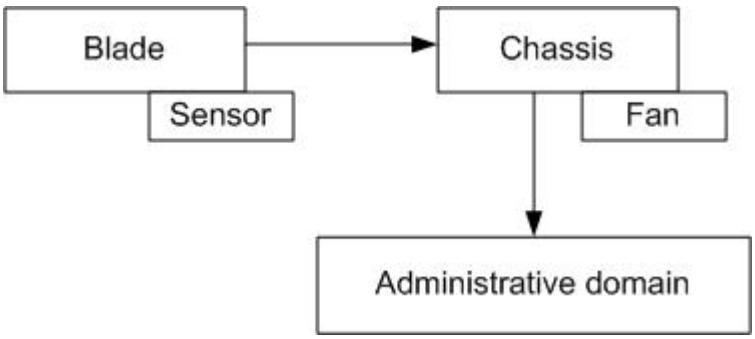

Figure 2. Example of managed element addressing

Given this hierarchy, suppose that:

- The administrative domain is specified in the CLP specification with the name `admin`.
- The physical aspect of a chassis is specified with the name `chassis`. The logical aspect of a chassis is specified with the name `modular`.
- A fan is specified with the name `fanpkg`.
- The logical aspect of a blade is specified with the name `system`.
- A temperature sensor is specified with the name `ntempsensor`.

As examples, you can specify the following SM ME addresses:

```
/admin1/chassis1/fanpkg1
/admin1/modular1/system1/ntempsensor1
```

Each of these given names (admin, chassis, modular, fanpkg, system, and ntempsensor) only specify a certain type (or class) of object. But the object names are very intuitive and, in the SMASH CLP, are termed *user-friendly class tags (UFcT)*.

However, as seen in Figure 2, CLP commands refer to a specific instance of an object type, for example, a particular blade. To differentiate between two blades, we can take the UFcT and add an instance number to the end of it. The result, `system1`, `system2`, `system3`, and so on, is called a *user-friendly instance tag (UFiT)*, in SMASH terminology.

Specifying an instance does not always make the managed element unambiguous. For example, system1 can reside in modular1 and another system1 can reside in modular6. To avoid ambiguity, you need to specify that the operation be performed on system1 residing in modular1 of admin1, for example. To specify this to a CLP program, you can write the target element as:

```
/admin1/modular1/system1
```

In SMASH CLP terminology, `/admin1/modular1/system1` is the *user-friendly instance path (UFiP)*.

In circumstances when all instances of a managed element need to be addressed, there is a short-hand notation for making such a selection: *user-friendly selection tags*

(*UFsTs*). UFsTs are constructed by concatenating the UFcT for a managed element with the character *. For example, to address all blades for a particular managed system, write the target as:

```
/admin1/modular2/system*
```

To assure there is agreement across vendors as to what to call a specific manageable element, DMTF has documented a *Server Management Managed Element Addressing Specification*. For details on accessing this document, see Chapter 7, "SMASH-related documentation," on page 109. Further discussion of addressing can be found in "Physical and logical target addressing" on page 18.

Chapter 3. Using the SMASH CLP command line

This section details the SMASH CLP command line and includes discussion of the following topics:

SMASH CLP syntax

The CLP syntax in the SMASH CLP is as follows:

```
verb options target properties
```

where

- *verb* refers to a specific command or action taking place.
- *options* are selections that affect the action, behavior, or output of the verb.
- *target* is the implicit or explicitly identified managed element at which the command is directed.
- *properties* are attributes of the target relative to running the command.

SMASH CLP verbs and OEM verb extensions

Verbs are the commands you can use to perform a management action on a specified target, such as an ME, resource, or object. You can modify verbs using verb options (see "SMASH CLP verb options" on page 9).

The SMASH CLP specifies a small number of verbs for usability reasons:

- **General:** cd, help, version, exit
- **Retrieve information:** show, dump
- **Manage:** set, create, delete, load
- **Change state:** start, stop, reset

In addition to these supported verbs, the SMASH CLP allows an original equipment manufacturer (OEM) to add support for vendor-unique commands. OEM extension name strings used for CLP command line terms must be identified by the CLP standard prefix `OEM` followed by a vendor-unique identification string. In IBM SMASH Proxy, this vendor unique string is `IICM`.

For additional information on OEM verbs, see “OEM verbs” on page 49.

SMASH CLP supported verbs and descriptions

The following table lists verbs supported in a SMASH CLP implementation and their descriptions.

Table 4. SMASH CLP verbs

Verb	Description
cd	This verb navigates through the hierarchy of addresses used to uniquely identify an ME. Specifically, this verb changes the address hierarchy or path from which the search for the specified ME starts. The relationship between addressing hierarchy and the final address of an ME is analogous to directory path and file names in file systems.
create	This verb creates a new instance of an ME. **Note:** SMASH Proxy Version 1.0 does not support the **create** verb.
delete	This verb deletes an instance of an ME.
dump	This verb dumps the binary image associated with an ME (for example, physical memory) to a specified location (the location is specified using a Uniform Resource Indicator [URI]). **Note:** SMASH Proxy Version 1.0 does not support the **dump** verb.
exit	This verb ends the associated usage session. A response is sent to the requesting client prior to session closure.
help	This verb displays help information associated with the usage of a verb or information about a target.
load	This verb fetches a binary image from a specified location (specified using a URI) and loads it on the indicated ME.
reset	This verb changes an ME to a known state. This is frequently the initial state but the verb can also be used to make the transition of the ME to any other valid state.
set	This verb sets the value of one or more of an ME’s properties. Use this verb to set either one property or a set of properties, but the operation can be performed on only one ME at a time.
show	This verb displays information about MEs. You can use it to show information about a single ME, a hierarchy of MEs, or MEs matching a property filter value.
start	This verb starts the ME.
stop	This verb stops the ME.
version	This verb shows the SMASH CLP version supported by the current implementation.

Detailed descriptions of each verb can be found in the *Server Management Command Line Protocol Specification (SM CLP), V1.0*, which you can access from Chapter 7, "SMASH-related documentation," on page 109.

The following table lists OEM verbs supported in a SMASH CLP implementation and their descriptions.

Table 5. OEM verbs

Verb	Description
oemiicmdiscover	This verb discovers BladeCenter chassis in the network for management.
oemiicmremovechassis	This verb deletes an individual chassis or all chassis in your administrative domain.
oemiicmlogin	This verb establishes new credentials for the user session.
oemiicmlogoff	This verb returns the previous set of credentials for the user.

For details on OEM verbs, see "OEM verbs" on page 49.

SMASH CLP verb options

SMASH CLP verb *options* modify the behavior of verbs or provide additional information. Options can appear immediately after the verb on the command line and must be preceded by a hyphen (**-**).

For example: `show -display targets /modular*`

where

- *-display* is the verb option.
- *targets* is the option argument.
- */modular** is the target or UFiP of the show command. *modular** indicates the UFsT for a chassis target, with *modular* being the logical UFcT name for a chassis target.

For a list of verb options and their descriptions, see "SMASH CLP supported verb options and descriptions." A specific discussion about the display option and its arguments can be found in "Display and Output verb options" on page 11.

SMASH CLP supported verb options and descriptions

This table lists verb options supported by the SMASH CLP and their descriptions.

Note: An italicized first letter in an option indicates the short form of the option. For example, you can specify the **version** option as -v.

Table 6. List of SMASH CLP supported verb options

Option	Description	Applicable verbs	Argument
-***a***ll	Instructs the verb to return all data element types subject to any filtering of categories by the `-display` option. **Note:** You must specify `-all` to see key properties, OEM properties, and OEM targets. OEM properties and targets are those which begin with `oemiicm` or `OEMIICM`.	**show**	NONE
-destination [*URI*]	Indicates the location of a destination for an image or other target data.	**dump**	URI or SM instance address.
-***d***isplay [*args*]	Selects the data that you want to display.	**show**	Multiple arguments controlling the type of information returned about a target.
-***ex***amine	Instructs the command processor to check the verb for syntactic and semantic correctness only.	All verbs	NONE
-***f***orce	Instructs the verb to ignore any warning conditions that would otherwise prevent implementation.	**Delete**, **dump**, **load**, **reset**, **set**, **show**, **start**, **stop**. Support for this option is not mandatory. **Note:** Although the `-force` option is allowed following the CLP specification on the **set, reset, start, stop, delete,** and **load** verbs, it has no effect on any verb in the IBM SMASH Proxy implementation.	NONE
-***h***elp	Displays documentation about the verb.	All verbs	NONE
-***k***eep (*m[.s]*)	Establishes a holding time for the job ID and status associated with a verb.	All verbs	Time to hold command job ID, status.

Table 6. List of SMASH CLP supported verb options (continued)

Option	Description	Applicable verbs	Argument
-***l***evel [*n*]	Instructs the command processor to run the verb for the current target plus targets contained through the specified level of depth.	**show**	Number of levels expressed as a natural number or `all`.
-***o***utput [*args*]	Controls the content and form of the verb output.	All verbs	Many arguments providing control of format, language, level of detail, order, and so on, of output data.
-source [*URI*]	Indicates the location of a source image or target.	**load**	URI or SM instance address.
-***v***ersion	Displays the version of the verb.	All verbs	NONE
-***w***ait	Instructs the command processor to hold the verb response until all spawned jobs have been completed.	All verbs except **exit** allow the `-wait` option following the CLP specification. However, `-wait` only affects the behavior of the **load** verb that runs asynchronously by default. All other SMASH Proxy verbs run synchronously.	NONE

Display and Output verb options

This section provides detailed descriptions of the frequently used -***d***`isplay` and -***o***`utput` verb options.

Display option: The formats for use of the display option are as follows:

```
-display (type)*[,(type)]
-d (type)*[,(type)]
```

The display option filters the information returned in the command results. It requires one or more arguments specifying the category of information to include in the command results.

The valid types and formats for the arguments of the display option are:

verbs Display the commands that are valid for this target.

properties[=*"*(*"*[*name* | *name*==*value*)***(*","*(*name* | *name*==*value*))*"*)*"*]**
Filter the command results such that information about an instance is returned only if the instance has all of the properties specified and the instance's property values match all of the property values specified. Only those properties specified by name, without a property value, are returned.

The properties are returned in the order indicated. If no property names are specified, SMASH returns all properties included in the command results.

Important:

1. SMASH requires parentheses only if more than one property name entry is specified.
2. Multiple *name==value* entries behave as an AND. SMASH returns results only if the target matches both property and value pairs.

targets[="("(*UFcT***][,(***UFCT***),(***UFcT***),...,(***UFcT***)")"]**

Filter the command results to only show information about the specified targets. If no UFcTs are specified, SMASH returns all targets included in the command results.

Note: SMASH requires parentheses only if more than one property name entry is specified.

associations[="("(*classname***] [,(***classname***),(***classname***),...,(***classname***)")"]**

Filter the command results to only show information about the specified associations. If no association class names are specified, SMASH returns all associations included in the command results.

Note: SMASH requires parentheses only if more than one property name entry is specified.

all Display targets, associations, verbs, and properties.

Display option examples

-> show /hdwr1/chassis1

```
-> show /hdwr1/chassis1
Success
UFiT: chassis1
    UFiP: /hdwr1/chassis1
    Properties:
        SKU: 59P6609
        SerialNumber: KPBK912
        Version: 0
        PartNumber: Not Available
        ManufactureDate: 01 January 2000 07:00:00 -300
    (20000101070000.000000-300)
        PackageType: Chassis/Frame (3)
        MultipleSystemSupport: True (1)
        Manufacturer: Not Available
        Model: 8677-2XX
        OtherIdentifyingInfo: 679AD2A1-32EC-11D8-BB2C-9B1406EFA202
    Verbs:
        Standard: help show
    UFiT: bladepkg1
    UFiT: bladepkg2
    UFiT: bladepkg4
    UFiT: bladepkg5
    UFiT: bladepkg6
    UFiT: pkg1
    UFiT: pkg2
    UFiT: pkg3
    UFiT: pkg4
    UFiT: modulepkg1
    UFiT: pwrpkg1
```

```
    UFiT: pwrpkg2
    UFiT: fanpkg1
    UFiT: fanpkg2
    UFiT: storagepkg1
```

Note: The -300 following the Hours:Minutes:Seconds (HH:MM:SS) on the `ManufactureDate` line is the minutes difference from GMT.

-> show -d verbs /hdwr1/chassis1

```
Success
UFiT: chassis1
UFiP: /hdwr1/chassis1
Verbs:
Standard: help set show
```

-> show -d targets,verbs /hdwr1/chassis1

```
Success
UFiT: chassis1
UFiP: /hdwr1/chassis1
Verbs:
Standard: help set show
UFiT: bladepkg1
UFiT: bladepkg2
UFiT: bladepkg4
UFiT: bladepkg5
UFiT: bladepkg6
UFiT: pkg1
UFiT: pkg2
UFiT: pkg3
UFiT: pkg4
UFiT: modulepkg1
UFiT: pwrpkg1
UFiT: pwrpkg2
UFiT: fanpkg1
UFiT: fanpkg2
UFiT: storagepkg1
```

-> show -d all /hdwr1/chassis1

```
Success
UFiT: chassis1
    UFiP: /hdwr1/chassis1
    Properties:
        SKU: 59P6609
        SerialNumber: KPBK912
        Version: 0
        PartNumber: Not Available
        ManufactureDate: 01 January 2000 07:00:00 -300
    (20000101070000.000000-300)
        PackageType: Chassis/Frame (3)
        MultipleSystemSupport: True (1)
        Manufacturer: Not Available
        Model: 8677-2XX
        OtherIdentifyingInfo: 679AD2A1-32EC-11D8-BB2C-9B1406EFA202
    Associations:
        MemberOfCollection <-- /hdwr1
        PackageInChassis --> /hdwr1/chassis1/bladepkg1
        PackageInChassis --> /hdwr1/chassis1/bladepkg2
        PackageInChassis --> /hdwr1/chassis1/bladepkg4
        PackageInChassis --> /hdwr1/chassis1/bladepkg5
        PackageInChassis --> /hdwr1/chassis1/bladepkg6
        PackageInChassis --> /hdwr1/chassis1/pkg1
        PackageInChassis --> /hdwr1/chassis1/pkg2
        PackageInChassis --> /hdwr1/chassis1/pkg3
        PackageInChassis --> /hdwr1/chassis1/pkg4
```

```
        PackageInChassis --> /hdwr1/chassis1/modulepkg1
        PackageInChassis --> /hdwr1/chassis1/pwrpkg1
        PackageInChassis --> /hdwr1/chassis1/pwrpkg2
        PackageInChassis --> /hdwr1/chassis1/fanpkg1
        PackageInChassis --> /hdwr1/chassis1/fanpkg2
        PackageInChassis --> /hdwr1/chassis1/storagepkg1
        SystemPackaging --> /modular1
        ConcreteDependency --> /modular1/ntempsensor1
    Verbs:
        Standard: help show
    UFiT: bladepkg1
    UFiT: bladepkg2
    UFiT: bladepkg4
    UFiT: bladepkg5
    UFiT: bladepkg6
    UFiT: pkg1
    UFiT: pkg2
    UFiT: pkg3
    UFiT: pkg4
    UFiT: modulepkg1
    UFiT: pwrpkg1
    UFiT: pwrpkg2
    UFiT: fanpkg1
    UFiT: fanpkg2
    UFiT: storagepkg1
```

Note: The -300 following the Hours:Minutes:Seconds (HH:MM:SS) on the `ManufactureDate` line is the minutes difference from GMT.

-> show -d targets,properties=(SKU,SerialNumber,Model,MultipleSystemS
upport==1,PackageType==3),associations=SystemPackaging

```
Success
UFiT: chassis1
UFiP: /hdwr1/chassis1
Properties:
SKU: 59P6609
SerialNumber: KPBK912
Model: 8677-2XX
Associations:
SystemPackaging --> /modular1
```

Output option: The formats for use of the output option are as follows:

`-output (`*arguments*`)`
`-o (`*arguments*`)`

The output option controls the format of output returned by the CLP to the client.

Table 7 lists allowable arguments for the option.

Table 7. Arguments for the output option

Argument	Value domain	Description
format=(*value*)	`text`, `keyword`, `clpxml`	Controls the structure of the output text. **Note:** The SMASH Proxy supports text and clpxml formats only. The clpxml format conforms to the SM CLP Command Response XML Schema (www.dmtf.org/apps/org/workgroup/svrmgmt/download.php/17388/dsp0224.xsd).
error, terse, verbose		Selects the level of detail included in the output.
language=(*value*)	A 3-character string identifier of language as specified in ISO 639.2; `eng` (English) is the default.	Selects the translation of the text. **Note:** The SMASH Proxy supports `eng` only.
begin, end		When multiple items are returned in the output, `begin` and `end` controls where to start and end, respectively, in the list.
order=(*value*)	`default`, `reverse`	When multiple items are returned in the output, `order` controls the order of those items.
count=(*value*)	(`integer string or all`)	When multiple items are returned in the output, `count` controls the number of items returned; the default is `all` items. Maximum value for *value* is determined by the class of the target.
number=(x-y)	[`integer string`]-[`integer string`]	Requests that a range of results be returned.

Output option examples

-> show -d verbs -o format=text record1

```
Success
UFiT: record1
    UFiP: /modular1/chassismgr1/logs1/log1/record1
    Verbs:
        Standard: delete help show
```

-> show -d verbs -o format=clpxml record1

```
[?xml version="1.0" encoding="utf-8"?]
[response xmlns="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_Response.xsd" xmlns:
xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:iicm="http://iicm.com/smas
hclp-extensions" xsi:schemaLocation="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_
Response.xsd CLPXML_Response.xsd"]
    [command]
```

```
        [inputline]show -d verbs -o format=clpxml record1[/inputline]
    [/command]
    [cmdstat]
        [status]0[/status]
        [status_tag]COMMAND COMPLETED[/status_tag]
        [job]
            [job_id]6[/job_id]
        [/job]
    [/cmdstat]
    [show]
        [target]
            [instance]
                [ufit ufct="record" instance="1"]record1[/ufit]
                [ufip]/modular1/chassismgr1/logs1/log1/record1[/ufip]
                [verbs]
                    [standardverbs]delete help show[/standardverbs]
                    [oemverbs/]
                [/verbs]
            [/instance]
        [/target]
    [/show]
[/response]
```

Note: Each instance of a left bracket ([) in the above example represents a *less than* (<) symbol and each instance of a right bracket (]) represents a *more than* (>) symbol. On screen, you see the < and > symbols, not brackets (for example, </show>). The left and right brackets are used for documentation purposes only.

-> show -display targets /modular1/chassismgr1/logs1/log1/record*

```
Success
UFiT: log1
UFiT: record1
UFiT: record2
UFiT: record3
UFiT: record4
UFiT: record5
UFiT: record6
UFiT: record7
UFiT: record8
UFiT: record9
UFiT: record10
UFiT: record11
UFiT: record12
UFiT: record13
```

-> show -display targets -output order=reverse /modular1/chassismgr1/logs1/log1/record*

```
Success
UFiT: log1
UFiT: record13
UFiT: record12
UFiT: record11
UFiT: record10
UFiT: record9
UFiT: record8
UFiT: record7
UFiT: record6
UFiT: record5
UFiT: record4
UFiT: record3
UFiT: record2
UFiT: record1
```

-> show -d targets -o begin,count=2 /modular1/chassismgr1/logs1/log1/record*

```
Success
UFiT: log1
UFiT: record1
UFiT: record2
```

-> show -d targets -o end,count=2 /modular1/chassismgr1/logs1/log1/record*

```
Success
UFiT: log1
UFiT: record12
UFiT: record13
```

-> show -display targets -output number=2-3 /modular1/chassismgr1/logs1/log1/record*

```
Success
UFiT: log1
UFiT: record2
UFiT: record3
```

SMASH CLP supported targets

A SMASH CLP *target* represents the address or path of the target of the verb. You specify the target in the hierarchical containment of an SM ME address that you derive from the concatenation of supported UFiTs into UFiPs (for details on UFiTs and UFiPs, see "Addressing managed elements" on page 4).

Most SMASH CLP verbs have a verb target, whether explicitly or implicitly identified (one exception is the **exit** verb). An explicitly identified target is a target address path that is included in the command line entered. An implicitly identified target is a target that you do not identify on the command line, but that the verb references from the session environment variable. This kind of target is called a *current default target (CDT)*.

A default target address is always in effect during a SMASH CLP session. The command processor uses it to determine the resultant target for the command. On entry to the SMASH CLP session, the CDT is always / (`/admin1`).

For example, the results of a command utilizing the `show` verb are identical for the following sequences. Each shows information about the target ME (indicated here by the SM ME address or UFiP `/modular1/switch1`) whether or not a target is explicitly defined:

```
  -> cd /modular1
  -> show switch1 (explicit)

Or
  -> cd /modular1/switch1
  -> show (implicit)

Or
  -> show /modular1/switch1 (explicit)
```

To view a list of SMASH CLP supported targets, see "SMASH Proxy nonaddressing associations and supported physical and logical targets" on page 64 and "SMASH Proxy supported targets (by UFcT) and associated command target properties" on page 72.

Physical and logical target addressing

As described in "Addressing managed elements" on page 4, SMASH uses SM ME addressing to provide a user-friendly way to accurately address managed elements or objects in a managed system. Objects are divided into physical and logical targets. *Physical targets* represent actual hardware that a user can touch, for example, a chassis, a blade, or a daughter card and *logical targets* represent software functions or entities, for example, a network configuration or an event log.

Note that a given entity can be represented both logically and physically. For example, a switch has a physical UFcT representation of *pkg* and a logical UFcT representation of *switch*.

You can address physical or logical targets, for example, a switch component on a BladeCenter enterprise chassis that requires action, as follows:

- The switch has a logical UFcT of *switch*. The enterprise chassis has a logical UFcT of *modular*.
- Because you manage a specific BladeCenter enterprise chassis and its switch, you further define the UFcTs to UFiTs that can be used in a target SM ME address within a CLP command used for managing the switch. In this case, the BladeCenter enterprise chassis unique UFiT identifier is *modular1* and the switch's unique UFiT is *switch1*.
- The resulting address to the switch would be: `/modular1/switch1`.

Addressing associations

As discussed in "Physical and logical target addressing," objects can be divided into physical and logical targets where physical targets represent actual hardware, for example, a chassis, and logical targets represent software functions or entities, for example, a network configuration.

Associations represent relationships between objects with the objects linking to each other through these associations. The *Server Management Managed Element Addressing Specification* explains that associations can be of two types: addressing and nonaddressing.

An *addressing association* means that you can use it to target an object with the following addressing format:

```
[parent object]/[object]
```

For example, admin1/hdwr1.

While a BladeCenter chassis is represented physically as a *chassis* target, and a BladeCenter blade is represented physically as a *bladepkg* target, the *containment* relationship of a blade in a chassis is represented by a `PackageInChassis` association. These associations can themselves be targets on a SMASH command line:

```
show chassis1=>PackageInChassis=>/hdwr1/chassis1/bladepkg1
```

Another example, Figure 3 on page 20, shows that the CIM_AdminDomain (UFcT is admin1) links to CIM_ConcreteCollection (UFcT is hdwr1) through the `OwningCollectionElement` association.

Further discussion of addressing associations can be found in "Addressing association diagrams."

For details on *nonaddressing associations*, see "SMASH Proxy nonaddressing associations and supported physical and logical targets" on page 64.

Addressing association diagrams

The following diagrams show all objects, UFcTs, and addressing associations supported by the SMASH Proxy. Each path from the root of a tree to one of the intermediate or terminal leaves represents a SMASH target. The root of the tree, /admin1, can be omitted from the target path. For example, in Figure 3 on page 20, the following targets are all possible:

```
/
/hdwr1
/hdwr1/chassis1
/hdwr1/chassis1/bladepkg1
/hdwr1/chassis1/bladepkg1/card1
/hdwr1/chassis1/bladepkg1/bladexpkg1
/hdwr1/chassis1/modulepkg1
/hdwr1/chassis1/fanpkg1
/hdwr1/chassis1/pwrpkg1
/hdwr1/chassis1/pkg1
/hdwr1/chassis1/storagepkg1
```

You can interpret Figure 4 on page 22 through Figure 7 on page 28 in a similar manner.

Note: There can be multiple chassis and multiple blades, power supplies, fans, inside a chassis. Thus, although the targets above are all listed with the UFiT instance number of 1, for example, chassis1, other UFiT instance numbers might also be possible depending on the chassis in your environment. For example, if you have two chassis and the second chassis has four power supplies, then /hdwr1/chassis2/pwrpkg4 is a valid target.

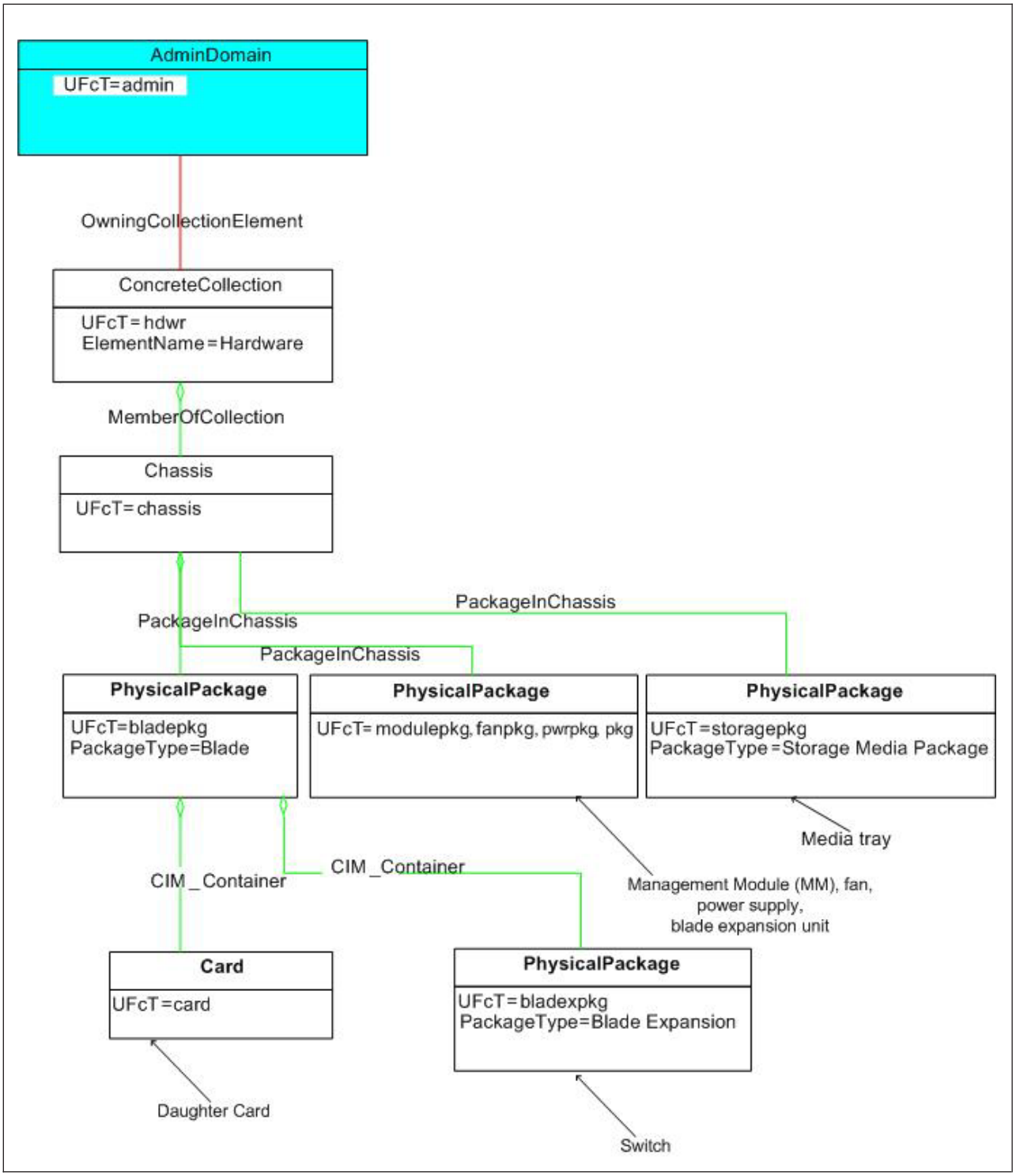

Figure 3. Physical target addressing

Sample path for this tree: To see physical data on the BladeCenter media tray, the target address might be:

```
/hdwr1/chassis1/storagepkg1
```

where:

- *hdwr* is the UFcT for the CIM Class ConcreteCollection, with the ElementName of *Hardware*.

- *chassis* is the UFcT for the CIM Class Chassis
- *storagepkg* is the UFcT for the CIM Class PhysicalPackage, with thePackageType of *Storage Media Package*.

hdwr links to chassis through the `MemberOfCollection` addressing association. *chassis* links to *storagepkg* through the `PackageInChassis` addressing association.

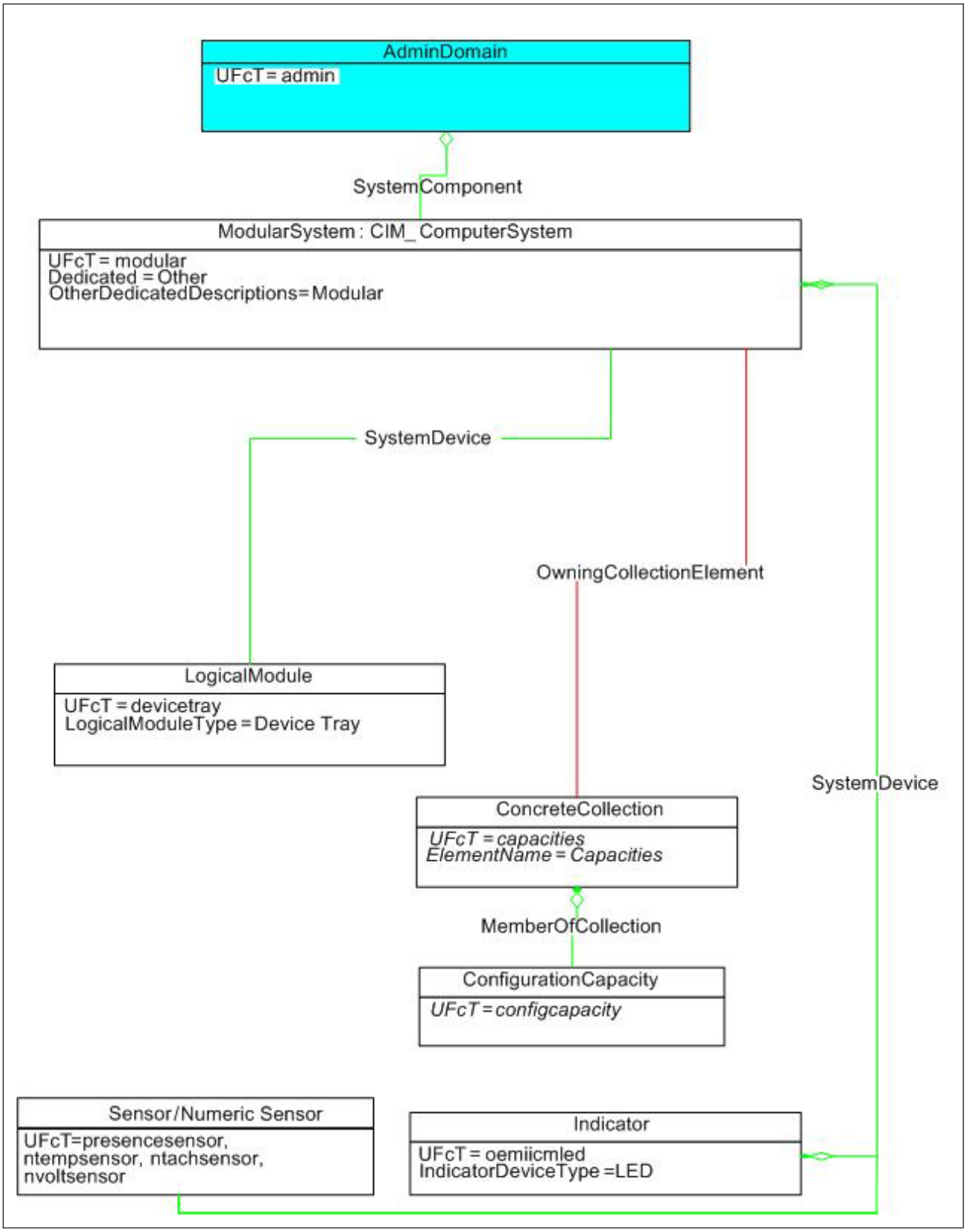

Figure 4. Logical target addressing - modular system and subcomponents

Sample path for this tree: To manage the MM device tray, the target address might be:

```
/modular1/devicetray1
```

where:

- *modular* is the UFcT for a component in the CIM Class ComputerSystem with OtherDedicatedDescriptions=*Modular*
- *devicetray* is the UFcT for a component in the CIM Class LogicalModule

modular links to *devicetray* through the `SystemDevice` addressing association.

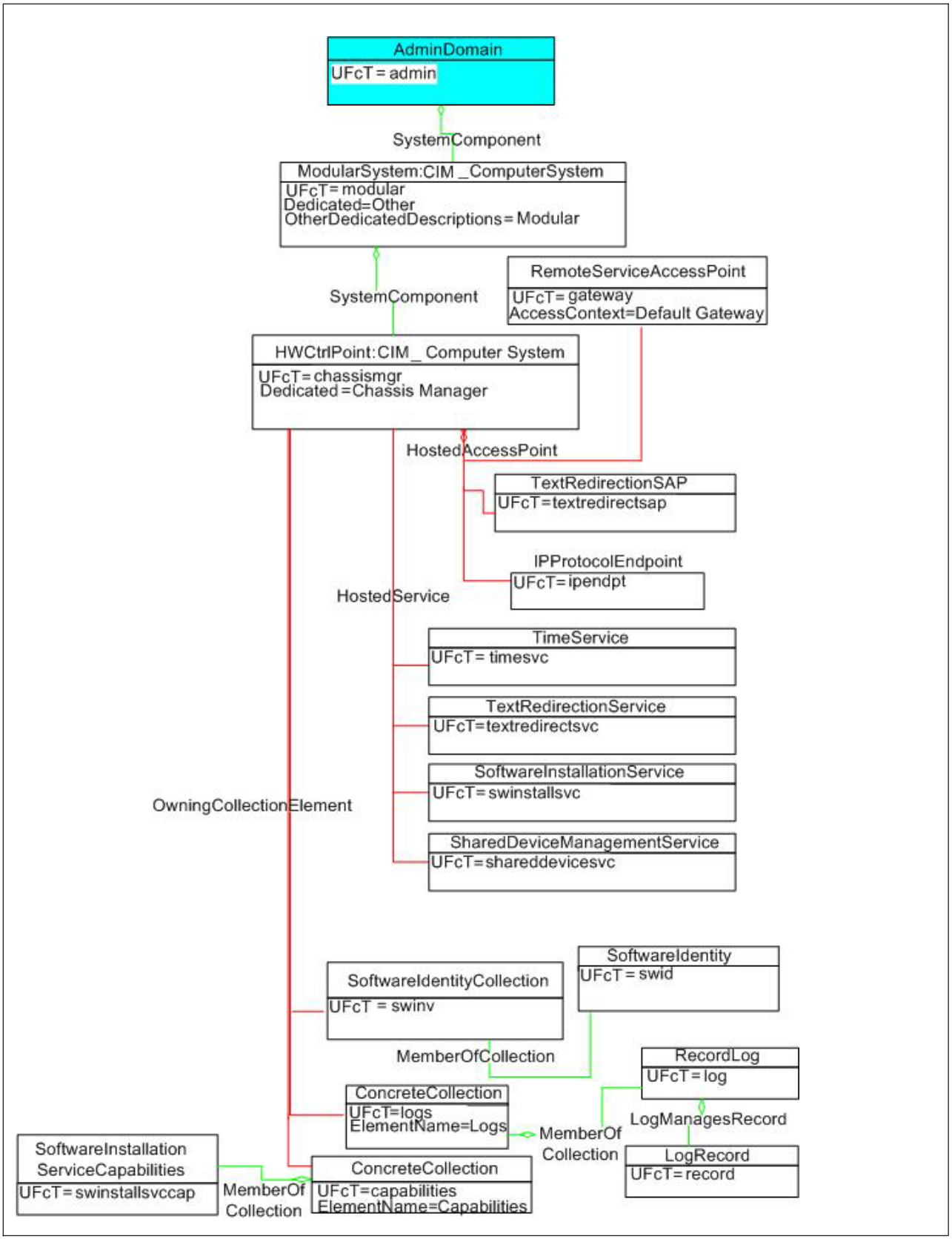

Figure 5. Logical target addressing - chassis manager (management module) and subcomponents

Sample path for this tree: To access the BladeCenter Text Redirection for blade 1, the target address might be:

```
/modular1/chassismgr1/textredirectsap1
```

where:

- *modular* is the UFcT for a component in the CIM Class ComputerSystem with OtherDedicatedDescriptions=*Modular*
- *chassismgr* is the UFcT for a component in the CIM Class ComputerSystem with Dedicated=*Chassis Manager*
- *textredirectsap* is the UFcT for the CIM Class TextRedirectionSAP

modular links to *chassismgr* through the `SystemComponent` addressing association. *chassismgr* links to *textredirectsap* through the `HostedAccessPoint` addressing association.

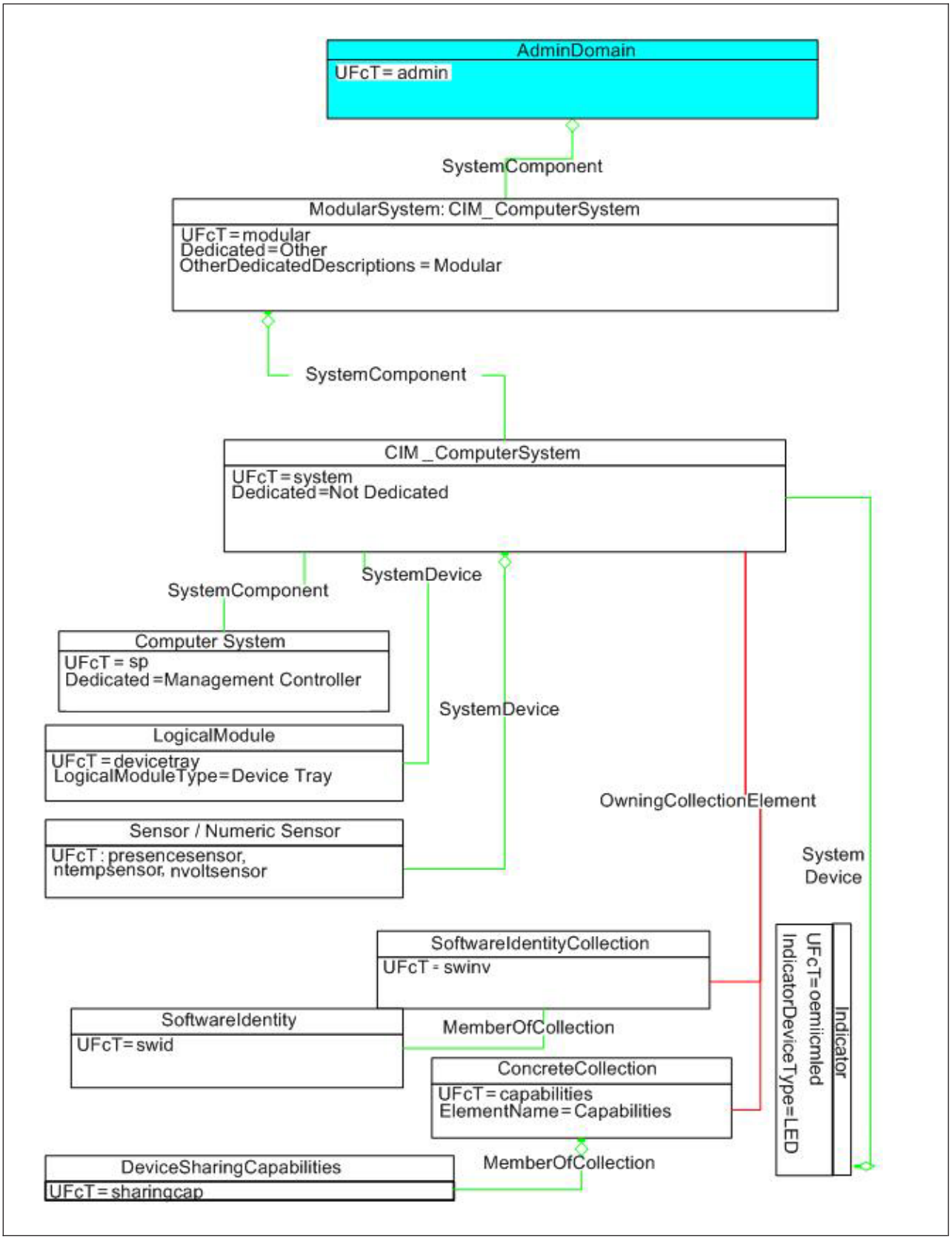

Figure 6. Logical target addressing - blade and subcomponents

Sample path for this tree: To manage a blade's expansion unit presence sensor, the target address might be:

```
/modular1/system1/presencesensor1
```

where:

- *modular* is the UFcT for a component in the the CIM Class ComputerSystem with OtherDedicatedDescriptions=*Modular*
- *system* is the UFcT for a component in the CIM Class ComputerSystem with Dedicated=*Not Dedicated*
- *presencesensor* is the UFcT for the CIM Class Sensor where SensorType=*Presence*

modular links to *system* through the `SystemComponent` addressing association.
presencesensor links to *system* through the `SystemDevice` addressing association.

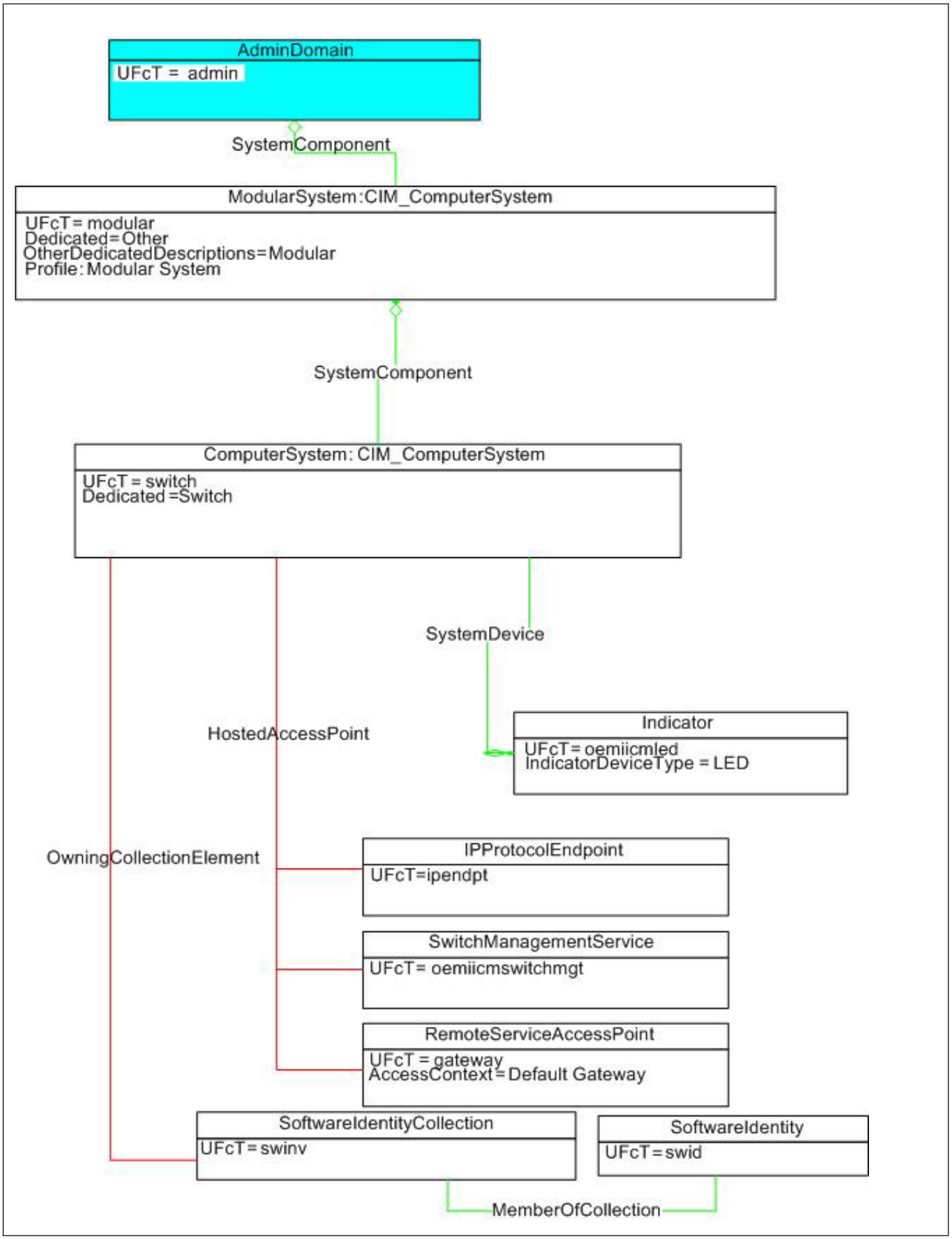

Figure 7. Logical target addressing - switch and subcomponents

Sample path for this tree: To manage the switch's IP endpoint, the target address might be:

```
/modular1/switch1/ipendpt1
```

where:

- *modular* is the UFcT for a component in CIM Class ComputerSystem with OtherDedicatedDescriptions=*Modular*
- *switch* is the UFcT for a component in CIM Class ComputerSystem where Dedicated=*Switch*
- *ipendpt* is the UFcT for the CIM Class IPProtocolEndpoint

modular links to *switch* through the `SystemComponent` addressing association. *switch* links to *ipendpt* through the `HostedAccessPoint` addressing association.

SMASH CLP profiles

Management of system elements through SMASH is described using a large number of profiles. A *profile* is an object model that describes an aspect of the system in terms of classes, associations, aggregations, and inheritances while keeping within the constraints of the CIM core model.

SMASH uses profiles to make heterogeneous systems appear similar, define required and recommended properties, and help define addressing and tags. Because addressing in SMASH is based on associations, profiles detail those required associations.

SMASH CLP supported command target properties

Command target *properties* are attributes that can contain values associated with a target that the SMASH CLP needs to process the command. Command target properties identify properties of the target class that the command retrieves or modifies.

You express property values in `property name=value`, `property name==value`, and just `property name` formats. There can be zero or more property terms per command.

For example, chassis properties include *Model, SerialNumber, SKU* and *Version*, where:

- *Model* is the vendor system model and machine type of the chassis.
- *SerialNumber* is a manufacturer-allocated number used to identify the chassis.
- *SKU* is the stock-keeping unit number for the chassis.
- *Version* is a string that indicates the version of the chassis.

For further details about command target properties, see "SMASH Proxy supported targets (by UFcT) and associated command target properties" on page 72, "SMASH Proxy command target property descriptions" on page 94, and "CIM property types" on page 102.

Command line editing

The SMASH CLP supports a rich set of command-line editing capabilities that save keystrokes for many typical sessions.

The following command-line editing directives are available for command entry:

Table 8. Command-line editing directives

Command-line editing directive	Purpose
Basic command editing	
Ctrl-B	Move back one character.
Ctrl-F	Move forward one character.
Backspace	Delete the character to the left of the cursor.
Ctrl-D	Delete the character underneath the cursor.
Cursor movement	
Ctrl-A	Move to the start of the line.
Ctrl-E	Move to the end of the line.
Ctrl-L	Clear the screen, reprinting the current line at the top.
Deletion	
Ctrl-K	Delete text from the current cursor position to the end of the line.
Ctrl-W	Delete the whole line.

Command history

You can access history items using the up or down arrow keys. Use the up arrow key to access next or later history items. Use the down arrow key to access earlier history items.

Command authority

Some commands can only be successfully run by users who are assigned a required level of authority; this authority level is the authority level of the corresponding user ID on the MM.

Chapter 4. The SMASH Proxy: An overview

The IBM SMASH CLP Proxy CLI provides direct access to BladeCenter management functions as an alternative to using the existing CLI or Web-based user interface. Using the CLP interface, you can view and manage the components by issuing commands that display the management objects, and enable you to control the power and configuration of the MM and other components in the BladeCenter unit. Unless otherwise noted, you can run all commands on the BladeCenter unit.

The SMASH CLP command line also provides access to the text-console command prompt on each blade server through a serial over local area network (LAN) (SOL) connection. Users can access the SMASH CLP interface by establishing a Telnet connection or a Secure Shell (SSH) connection to the IP address of the proxy. You can initiate connections from the client computer using standard remote communication software; no special programs are required.

For a user to manage a BladeCenter or BladeCenter T chassis with the SMASH CLP Proxy, they must be authenticated for that chassis MM and the MM must be accessible from the proxy management station.

The proxy implements a subset of the planned Server Management Workgroup SMASH profiles as follows:

- Modular System Profile
- Physical Asset Profile
- Shared Device Management Profile
- Chassis Manager Profile
- Device Tray Profile
- Collections Profile
- Base System Profile
- Software Identity Profile
- Firmware Update Profile
- Record Log Profile
- Text Console Redirection Profile
- Sensors Profile
- Alarms Profile

The SMASH Proxy seeks to be compliant with the SMASH standard being developed by the SMWG. Because this standard is still under development, some differences may exist between the SMASH Proxy implementation and the SMASH standard.

You can access the most recent versions of all BladeCenter and BladeCenter T documentation from the IBM Web site. Complete the following steps to check for updated BladeCenter and BladeCenter T documentation and technical updates:

1. Go to the IBM Support Web site (www.ibm.com/pc/support/).
2. In the Learn section, select **Publications**.
3. On the Publications page, in the **Brand** field, select **Servers**.
4. In the **Family** field, select **BladeCenter** or**BladeCenter H** or **BladeCenter T**.

5. Select **Continue** to view a list of results.

The SMASH Proxy is supported on the following operating systems:

- RedHat Enterprise Linux® ES 4.0 Update 1 and later
- SUSE Linux Enterprise Server 9.0 Service Pack 1 and later

The only output language supported by the SMASH Proxy is English, identified by the ISO 690-2 code `eng`. SMASH CLP does not include support for allowing a user to select a locale. In text mode output, values such as date or time are returned in U.S. English format.

The SMASH Proxy architectural model

See Figure 8 on page 33 for a model of the SMASH Proxy implementation, displaying its concrete components.

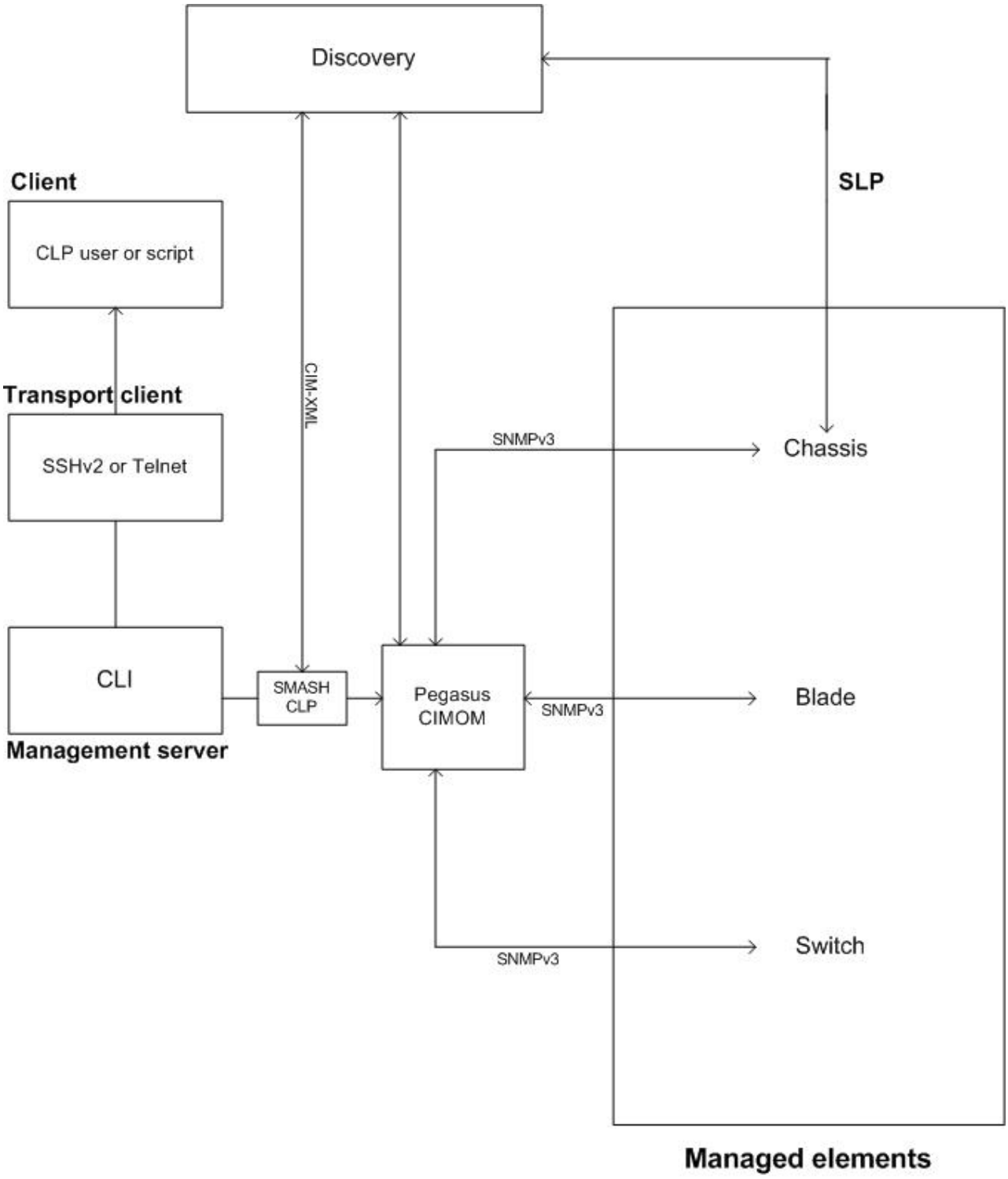

Figure 8. SMASH Proxy architectural model

This model expands on the basic architectural model described in "SMASH CLP architectural model" on page 3. The model depicts the presence of the *CIM Object Manager* (*CIMOM*) and the method of discovery in a SMASH Proxy implementation.

The flow in Figure 8 is as follows:

1. The client establishes a Telnet or SSH connection to the SMASH Proxy and logs in as clpuser, which brings up the SMASH CLP shell.
2. The client issues the **oemiicmdiscover** command to kick off the *Discovery* program.
3. The discovery program performs Service Location Protocol (SLP) queries to detect elements being managed by the SMASH CLP.
4. The client issues a command in order to manage the chassis, blades, or switches.
5. The discovery program communicates with the Pegasus CIMOM to add the managed elements (Pegasus is an open-source implementation of the DMTF CIM and WBEM standards).
6. The SMASH CLP command processor talks to the Pegasus CIMOM, which invokes the appropriate CIM providers to extract managed element data.
7. The requested status and data is returned to the client.

Chapter 5. Using the SMASH Proxy

This section details SMASH Proxy utilization and includes discussion of the following topics:

Before you begin

You must correctly configure the BladeCenter unit before you can use the SMASH Proxy CLI to manage it. Hardware and software required for the SMASH Proxy CLI are as follows:

SMASH Proxy hardware
The SMASH Proxy requires a network connection to the BladeCenter chassis that are management targets. The IBM server that the Proxy resides on should have at least 512 MB of memory and 40 MB of available disk space.

SMASH Proxy software
The SMASH Proxy requires Red Hat Enterprise Linux ES 4.0 Update 1 and SUSE Linux Enterprise Server 9.0 Service Pack 1 (SLES 9).

BladeCenter hardware
The SMASH Proxy requires no special hardware other than a network connection to use it with a BladeCenter chassis.

To use the SOL feature with a blade, the blade must support SOL functionality.

BladeCenter firmware
Make sure you are using the latest version of firmware for your blade servers, management modules, and other BladeCenter components for all chassis that the SMASH Proxy will manage.

Go to the IBM Support Web site (www.ibm.com/pc/support/) for the latest information about upgrading the firmware for BladeCenter components. The latest instructions are in the documentation that comes with the updates. The SMASH Proxy CLI is supported on:

- BladeCenter management module firmware level 1.21F and later
- BladeCenter T management module firmware level 1.07G and later
- BladeCenter H management module firmware level 1.01 and later

The SOL feature has additional firmware requirements. For more information, access the *IBM eServer™ BladeCenter and BladeCenter Serial over LAN Setup Guide*:

1. Go to the IBM Personal computing support - Troubleshooting Serial over LAN issues - IBM BladeCenter Web site (www-307.ibm.com/pc/support/site.wss/document.do?lndocid=MIGR-59728).
2. On the left menu, select **Publications**.
3. On the Publications page, in the **Brand** field, select **Servers**.
4. In the **Family** field, select **BladeCenter** or **BladeCenter H**, or **BladeCenter T**. Select **Continue** to go to the next step.
5. Using the menu beside **Refine results**, select **Serial over LAN**.
6. Select **Serial over LAN (SOL) Setup Guide - IBM BladeCenter, T**.

SMASH Proxy supported BladeCenter components

The following tables contain a listing of SMASH Proxy supported BladeCenter components with machine types and part numbers.

Table 9. Supported BladeCenter chassis

Model	Machine type
BladeCenter Entry Chassis	7967
BladeCenter Chassis	8677
BladeCenter T Chassis (ac)	8730
BladeCenter T Chassis (dc)	8720
BladeCenter Chassis H Chassis	8852

Table 10. Supported blades

Model	Machine type
HS20	7981
HS20	8678
HS20-Ref	8832
HS20-800-MHz	8843
LS20	8850
HS40	8839
JS20 GA1	8842 (21X)
JS20 GA2	8842 (4XX)
JS21	8844
HS21	8853

Table 11. Supported BladeCenter management modules

Name	Part number	FRU number
Management Module	48P7081	39M4945
Advanced Management Module	25R5779	25R5777

Table 12. Supported BladeCenter switches

Name	Part number	FRU number
IBM BladeCenter Optical Pass-thru Module	02R9080	02R9082
Cisco Systems Intelligent Gigabit Ethernet Switch Module for IBM BladeCenter	13N2281	13N2285
Cisco Fiber Switch Module	13N2285	13N2286
Cisco Systems Fiber Intelligent Gigabit Ethernet Switch Module for IBM BladeCenter	26K6547	
Brocade Entry SAN Switch Module for IBM BladeCenter	26K5601	90P0164
Qlogic Enterprise 6-port Fibre Channel Switch Module for IBM BladeCenter	26K6477	26K6481
Nortel Networks Layer 2/3 Copper Gigabit Ethernet Switch Module for IBM BladeCenter	26K6530	26K6526
Nortel Networks Layer 2/3 Fiber Gigabit Ethernet Switch Module for IBM BladeCenter	26K6531	26K6529
IBM BladeCenter 2-Port Fibre Channel Switch Module	48P7062	59P6621
IBM BladeCenter Copper Pass-Thru Module	73P6100	73P6098
Nortel Network Layer 2-7 Gigabit Ethernet Switch Module for IBM BladeCenter	73P9057	73P9004
Brocade Enterprise SAN Switch Module for IBM BladeCenter	90P0165	90P0164
Intel® Gigabit Ethernet Switch Module for IBM BladeCenter T	90P3776	
McDATA 6-port Fibre Channel Switch Module for IBM BladeCenter	32R1790	
Topspin IBM Switch Module for IBM BladeCenter	26K6454	

Installing the SMASH Proxy

The SMASH Proxy installs on Red Hat Enterprise Linux ES 4.0 Update 1 (4.1) and SUSE Linux Enterprise (SLES) Server 9.0 Service Pack 1 and Service Pack 2. In addition, the SMASH Proxy requires the following packages beyond the set of packages installed when installation defaults for these operating systems are selected:

Red Hat

```
RHEL4.0 cd2 - net-snmp-libs-5.1.2-11.i386.rpm
RHEL4.0 cd2 - lm_sensors-2.8.7-2.i386.rpm
RHEL4.0 cd2 - net-snmp-5.1.2-11.i386.rpm
RHEL4.0 cd2 - libtool-libs-1.5.6-4.i386.rpm
RHEL4.0 cd3 - tcl-8.4.7-2.i386.rpm
RHEL4.0 cd3 - expect-5.42.1-1.i386.rpm
```

SUSE

```
SLES9.0 cd2 - expect-5.42.1-1.i386.rpm
```

The SMASH Proxy installation script performs a check for all prerequisites. If any required components are missing, they will be listed and the installation will not continue.

To install the SMASH Proxy, perform the following steps:

1. Log in as a user with administrative authority (for example, root).
2. Run the script `installSmashProxy.sh` located at the top of the product tree. It is not necessary to change to the directory of this script to run it.

Please note the following information:

- `installSmashProxy.sh -h` displays the usage syntax for the installation script.
- By default, the SMASH Proxy CIMOM is installed in `/opt/ibm/cimom`. The remainder of the SMASH Proxy product is installed in `/opt/ibm/smash`.
- The base directory `/opt/ibm/smashProxy` is relocatable. You can override its default values by using `-p [ProductInstallDir]` when invoking `installSmashProxy.sh` from the command line.
- If the required CIMOM (IBMCimCore) is already on the system, the SMASH Proxy uses it and no new instance is installed. This is most common when IBM Director Server has been previously installed.
- Ports 5988, 5989, and 9879 must be open in order to install the SMASH Proxy CIMOM. If another CIMOM is running on these ports, you must remove it before you can install the SMASH Proxy.
- The installation script checks the environment, performs configuration, and launches a set of Red Hat Package Manager (RPM) files. See Table 13 for a list of the RPM package names for each element of the SMASH Proxy product, along with brief descriptions.

Table 13. RPM package names

Package Name	Description
IBMCimCore	SMASH Pegasus server
iicm-cimclientlib	Small-footprint CIM client library
iicm-cimschema	CIM schema
iicm_common	Common functions
iicm-credentialsserver	Credentials server
iicm-discovery	oemiicmdiscover program
iicm-mmprovider	MM provider
iicm_serialoverlan	SOL support
iicm-smashclp	SMASH CLP

Configuring SNMPv3 in the SMASH Proxy

You must configure Simple Network Management Protocol (SNMP) V3 on all chassis managed through the SMASH Proxy. The chassis SNMPv3 configuration values must match those in file /opt/ibm/smashProxy/cfg/smash_snmp.cfg. See the following example for an /opt/ibm/smashProxy/cfg/smash_snmp.cfg file with the default ship values.

SMASH SNMP configuration file

```
# SMASH SNMP Configuration File
# This file allows configuration of the SNMPv3 default parameters to use in
# communications between the SMASH proxy and managed Bladecenters.

# snmp_timeout in microseconds, default is 2.0 sec
# This is the time the SMASH Proxy will wait for a response from the managed
# Bladecenter before retrying an SNMP request.
snmp_timeout = 2000000

# session_timeout, in seconds, default is 5 min (300 sec)
# This is the amount of idle time the SMASH Proxy will allow before closing an
# SNMP session between a SMASH Proxy user and a managed Bladecenter. This only
# closes out the underlying SNMP session not the SMASH CLP user session.
# If the user/Bladecenter SNMP session has been closed due to idleness, it will
# be automatically reopened at the next user query.
session_timeout = 300

# snmp_retries, default is 3
# This is the number of times the SMASH Proxy will retry an SNMP request to a
# managed Bladecenter before sending an error back to the user.
snmp_retries = 3

# context
# This parameter is the default context to be used in Bladecenter SNMPv3
# communications.
# When the user logins to the SMASH Proxy or issues an oemiicmlogin command
# he can specify a username:context. If the user omits the context from the
# login, the default context specified in this file will be used for the SNMPv3
# request. The context in the SNMPv3 setup web interface for the Bladecenter
# must match the value submitted in the login or the value in this file (if the
# context was not specified at login). If context is blank in any of the web
# interfaces for the Bladecenters to be managed, then context should be
# commented out here.
#context = admin

# authentication_protocol
# This value should match the authentication protocol specified in the SNMPv3
# setup web interface for all managed Bladecenters. All managed Bladecenters
# must be configured with the same authentication protocol.
# Allowable values are MD5, SHA, and None
authentication_protocol = MD5


# privacy_protocol
# This value should match the privacy protocol specified in the SNMPv3 setup
# web interface for all managed Bladecenters. All managed Bladecenters must be
# configured with the same privacy protocol.
# Allowable values are DES and None
privacy_protocol = DES

# snmp_logfile
# This is the file where all debug messages will be logged.
snmp_logfile = /var/log/iicm/smash_snmp.log

# snmp_debug
# This controls logging of net-snmp library debug messages to snmp_logfile
# A value of 0 means debug is turned off.
# Any other value (e.g. 1) turns on snmp debug messages.
snmp_debug = 0

# num_sessions
# Number of simultaneous user sessions, default is 14
num_sessions = 14
```

context, *authentication_protocol*, and *privacy_protocol* must all align with the MM SNMPv3 settings. “SNMPv3 configuration in the MM” explains how to configure these settings using the MM Web interface.

SNMPv3 configuration in the MM

To configure SNMPv3 using the management module Web interface, follow these directions:

1. Using your browser, type the IP address in the following format:

   `http://xxx.xxx.xx.xxx/private/main.ssi`

   replacing *xxx.xxx.xx.xxx* with the IP address of the MM you are configuring (for example, 192.168.70.125).
2. Scroll down the left navigation menu, and click **Network Protocols**.
3. On the Network Protocols panel, scroll down to the Simple Network Management Protocol (SNMP) section.
4. Set the SNMPv3 agent to **Enabled** as indicated in Figure 9. Enablement of SNMPv1 and SNMP traps are not required by SMASH. You can enable each individually or both simultaneously depending on what you require for other management applications.

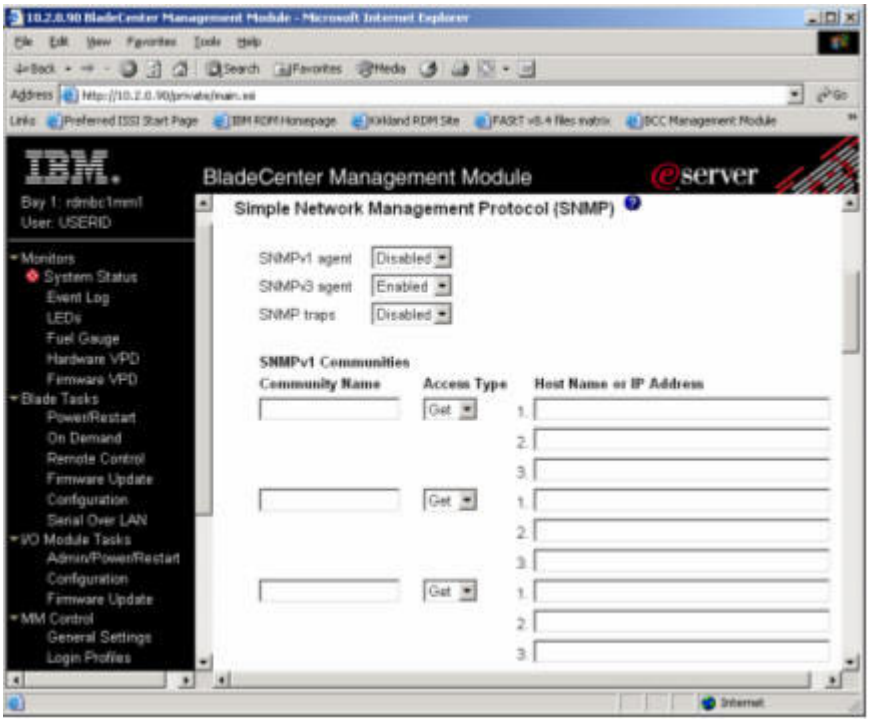

Figure 9. BladeCenter Management Module Network Protocols panel - SNMP section

5. Scroll to the bottom of the Network Protocols panel and click the **Save** button.
6. A pop-up screen will result with the following note: *Changes to SNMP and DNS settings will take effect after the next restart of the MM. Changes to other protocol settings will take effect immediately*. Click the **OK** button.
7. Go to the left navigation menu and scroll down to **Login Profiles**. Click on Login Profiles.

8. From the Login Profiles panel, select a **Login ID** to configure its profile.

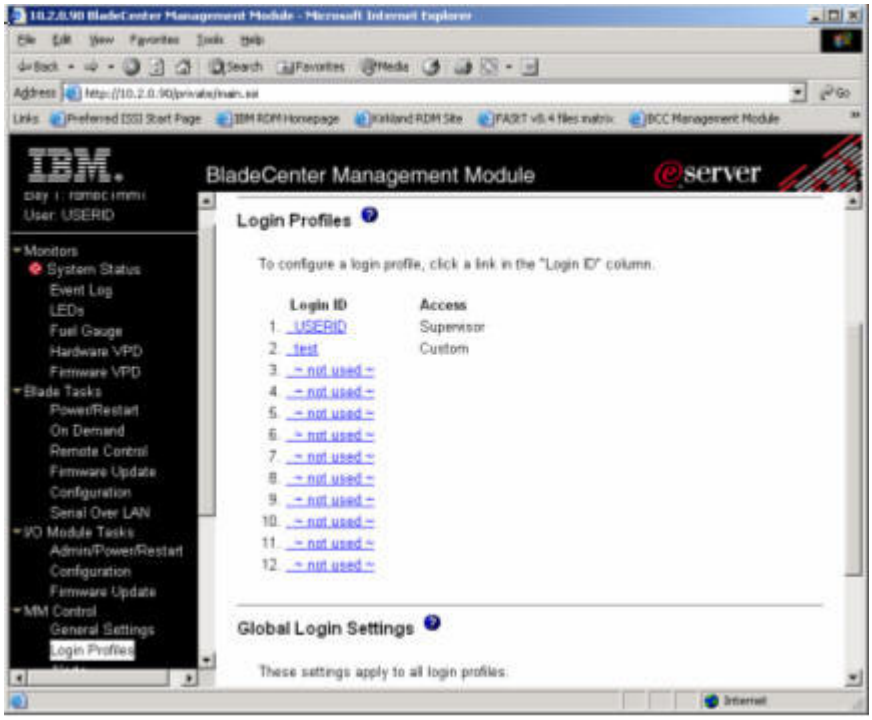

Figure 10. BladeCenter Management Module Login Profiles panel

9. Scroll to the bottom of the Login ID panel (Figure 11 on page 42), and click **Configure SNMPv3 User**.

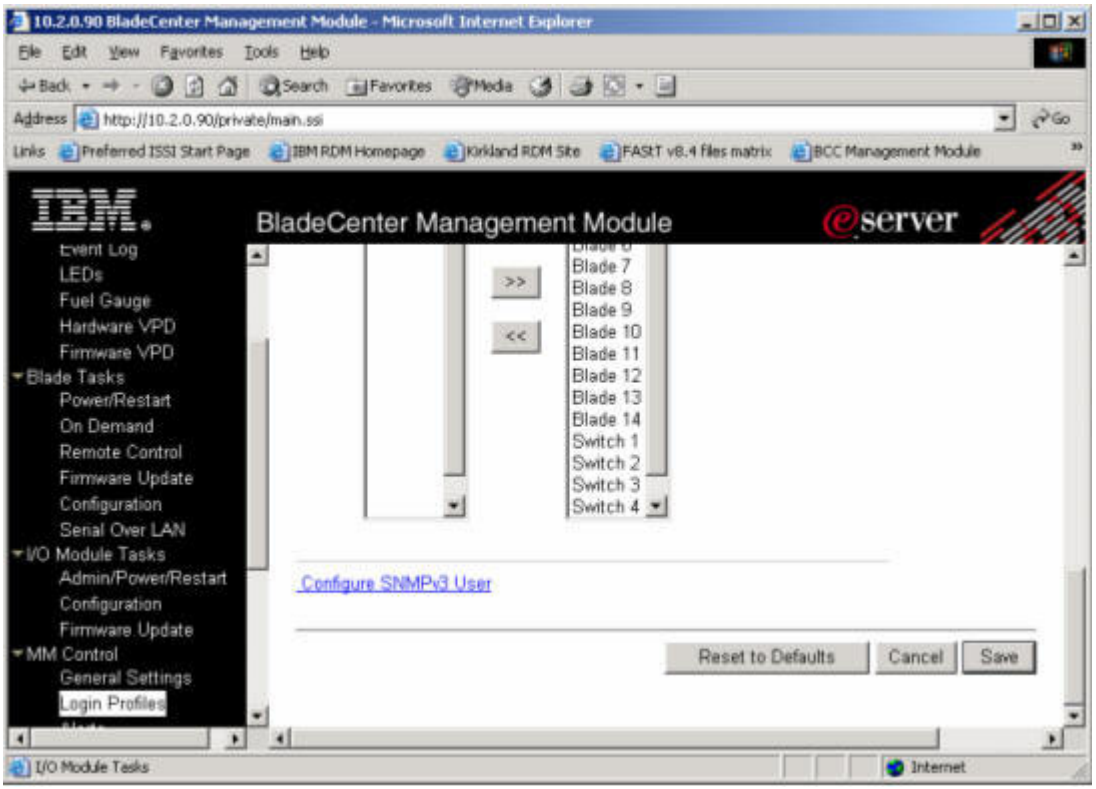

Figure 11. BladeCenter Management Module Login ID panel- Configure SNMPv3 User

10. On the SNMPv3 User Profile panel (Figure 12 on page 43), select and complete the appropriate fields. Click **Save**.

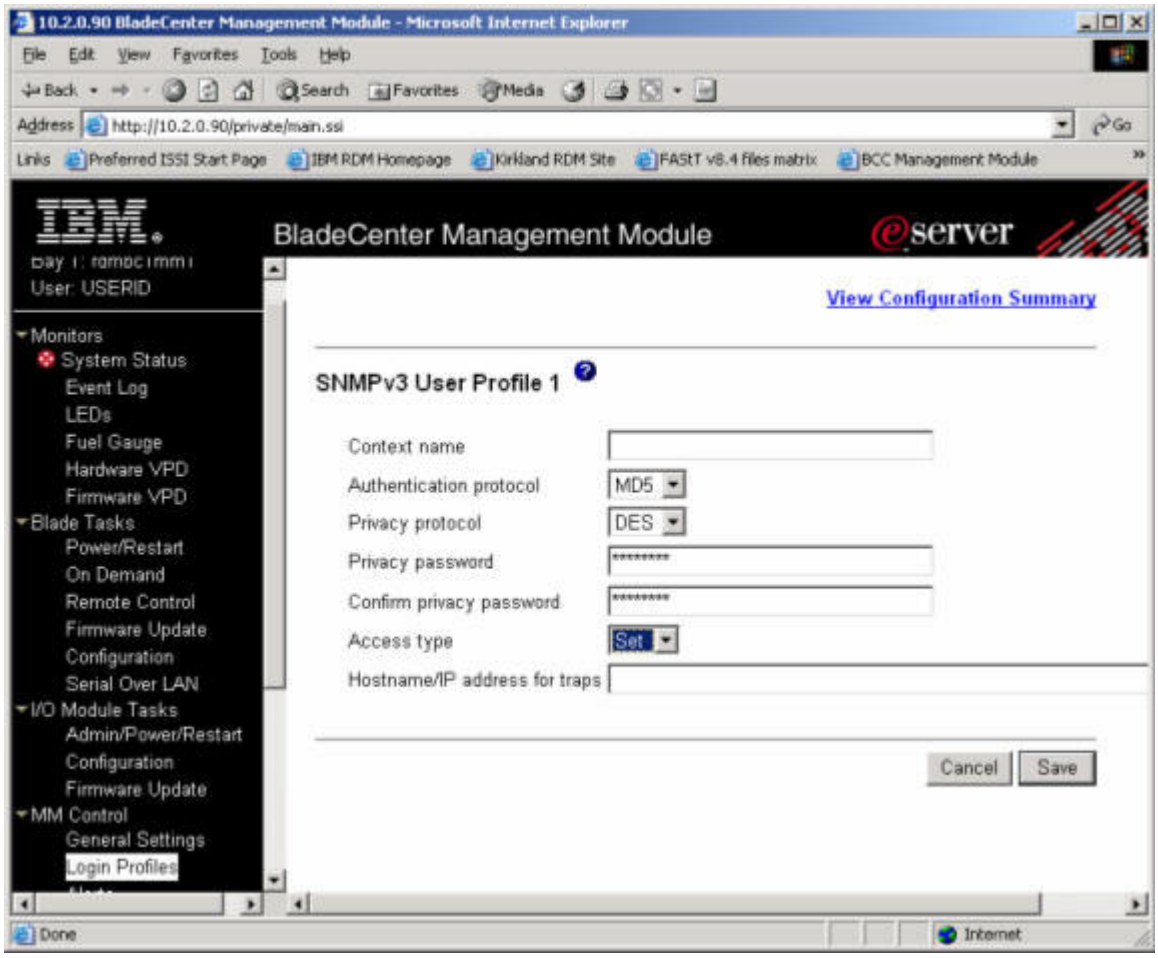

Figure 12. BladeCenter Management Module SNMPv3 User Profile panel

You can configure a total of twelve user profiles.

The following fields are configurable for each user profile:

Context name
Specifies a collection of management information accessible by an SNMPv3 manager.

Authentication protocol
You configure this option to select one of the following authentication protocols: none, Message Digest 5 (MD5), or Secure Hash Algorithm (SHA). If you do not select one of these protocols, SMASH does not require the authentication password and does not use it when processing the user profile. When you select **MD5** or **SHA**, SMASH hashes the authentication password according to the authentication protocol that you specify.

Privacy password
You use this option to encrypt outgoing messages and to decrypt incoming messages according to the privacy protocol that you specify. The length of the password should be at least 8 alphanumeric characters. If you use the privacy password, SMASH requires it to match the user's authentication password, which also must be at least 8 characters in length.

Privacy protocol
You use this option to select one of the following privacy protocols: none or Symmetric Encryption Protocol (SEP) Data Encryption Standard (DES). If you do not select this protocol, SMASH does not require the privacy password and does not use it when processing the user profile. When you select **DES**, SMASH uses the privacy password to encrypt outgoing messages and to decrypt incoming messages.

Access type
Specifies the access level for this user profile. Change this to **Set** to allow full read-write functionality by the SMASH Proxy. Change this to **Get** to allow read-only functionality by the SMASH Proxy.

Host name or IP address
Optional, you can leave this field blank for SMASH.

For more details on SNMP configuration, see "Configuring SNMPv3 in the SMASH Proxy" on page 39. An in-depth view of the BladeCenter Management Module Web interface can be found in *Management Module User's Guide - IBM BladeCenter, BladeCenter T*, which you can access from Chapter 7, "SMASH-related documentation," on page 109.

SLP configuration in the MM

Note: By default, on the MM, the SLP is set to **Multicast**. IBM recommends this setting remain unchanged.

On the BladeCenter Management Module, under Service Location Protocol (SLP), you can set the SLP to either **Broadcast** or **Multicast**. If you set it to **Broadcast**, you can do an **oemiicmdiscover** process by unicast address (for example, 10.0.0.3), subnet broadcast address (for example, 10.0.0.255) or global broadcast address (for example, 255.255.255.255), but not by multicast address. If you set it to **Multicast** with a multicast address of 239.255.255.253, the default, you can do an **oemiicmdiscover** process by unicast address, subnet broadcast address or multicast address, but not by global broadcast address.

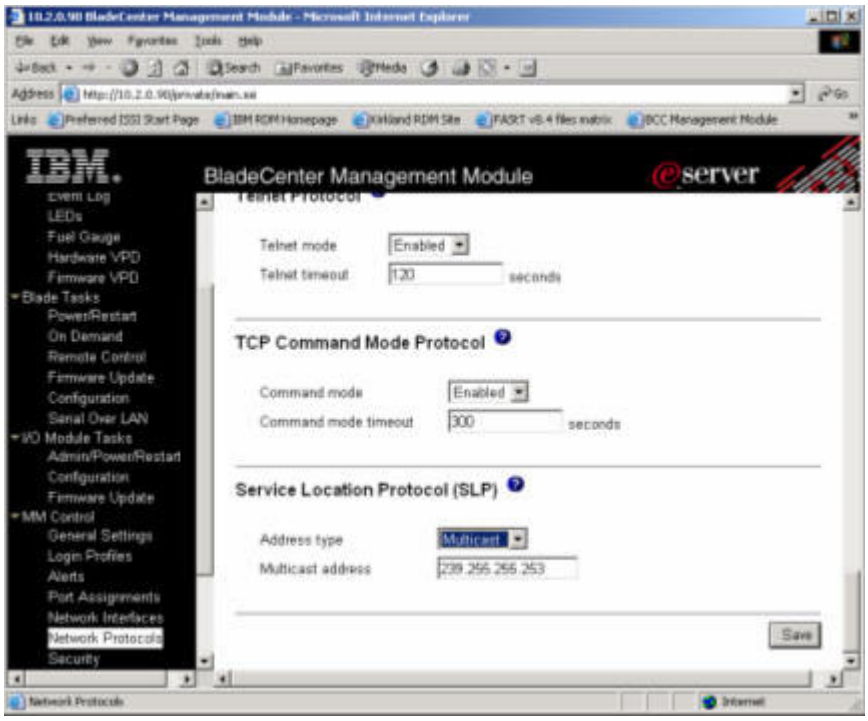

Figure 13. SLP configuration in the MM

Configuring the credentials server

You can use the `/etc/iicm/iicm_cred.conf` file to modify parameters for the credentials server, such as the maximum number of sessions and the session timeout period. When modifying the maximum number of sessions, IBM recommends that you specify a value no greater than 14.

Credentials server configuration file

```
# Configuration file for the IICM Cred programs/libraries.

# Number of accounts
# accounts = 14

# Timeout for accounts
# timeout = 900

# Log mode: base:level:sink
# log-mode = standard::/var/log/iicm/iicm_credserv.log

# Debug mode
# debug = false
```

Accessing the SMASH Proxy: Remote user access

You access the SMASH Proxy from a client computer by establishing a Telnet connection or by establishing an SSH connection to the IP address of the Proxy host server. You can establish several Telnet or SSH sessions to the SMASH Proxy, giving you the ability to have multiple CLP sessions active at the same time. Although a remote network administrator can access the SMASH Proxy interface

through Telnet, this method does not provide a secure connection. As a secure alternative to using Telnet to access the SMASH Proxy interface, SSH ensures that all data that is sent over the network is encrypted and secure.

SSH access

To log on to the SMASH Proxy using SSH, complete the following steps:

1. Make sure that the SSH service on the network-management workstation is enabled. See your operating-system documentation for additional instructions.
2. To allow logon using SSH as clpuser, make sure that the following variables are set to the given values in the `/etc/ssh/sshd_config` file:

   For Red Hat 4.1:

   ```
   PermitEmptyPasswords yes
   ```

   For SLES 9 with service pack 1:

   ```
   PasswordAuthentication yes
   PermitEmptyPasswords yes
   ChallengeResponseAuthentication no
   ```

   For SLES 9 with service pack 2:

   ```
   PasswordAuthentication yes
   PermitEmptyPasswords yes
   ChallengeResponseAuthentication no
   UsePAM yes
   ```
3. Start an SSH session to the SMASH Proxy server using the SSH client of your choice. For example, if you are using the cygwin client, open a command-line window on the network-management workstation, enter: `ssh –l clpuser 192.168.70.125`. The IP address 192.168.70.125 is a sample IP address of the SMASH Proxy. If a different IP address has been assigned to the SMASH Proxy host, use that one instead.
4. At the user name user name prompt, type the user ID for the target chassis management module. At the password prompt, type the MM password. The user ID and password are case-sensitive and are the same as those that are used for MM Web access.
5. A command prompt is displayed, allowing you to enter commands to the SMASH Proxy.

Note: If you omit -l clpuser, you get an extra login prompt before the SMASH shell user name and password login prompt.

The following example contains a sample output from an SSH login:

```
[root@repuda sbin]# ssh -l clpuser repuda
Last login: Wed Apr 19 14:45:09 2006 from repuda.raleigh.ibm.com

Username: USERID

Password:

=== SM CLP v1.0.0 SM ME Addressing v1.0.0 IICM Implementation v1.0.0 ===->
```

The SMASH Proxy has been tested with the following SSH clients:

- The SSH clients distributed with the Red Hat 4.1 and SLES 9 operating systems (see your operating-system documentation for information).
- The SSH client of Cygwin (for details, see the Cygwin Web site [www.cygwin.com]).
- PuTTy (for details, see the PuTTy Web site [www.chiark.greenend.org.uk/~sgtatham/putty/]).

Additional SSH clients are available and should work with the SMASH Proxy; however, support for these SSH clients is not implied.

Telnet access

To log on to the SMASH Proxy using Telnet, complete the following steps:

1. Open a command-line window on the network-management workstation and enter: `telnet -l clpuser 192.168.70.125`. The IP address 192.168.70.125 is a sample IP address of the SMASH Proxy. If a new IP address has been assigned to the management module, use that one instead.
2. At the user name prompt, type the user ID for the target chassis MM. At the password prompt, type the MM password. The user ID and password are case-sensitive and are the same as those that are used for Web access to the target MM.
3. A command prompt is displayed, allowing you to enter commands to the SMASH Proxy.

Note: If you omit -l clpuser, you get an extra login prompt before the SMASH shell user name and password login prompt.

The following example contains a sample output from an Telnet login:

```
[root@repuda sbin]# telnet -l clpuser repuda
Trying 9.42.237.131...
Connected to repuda.raleigh.ibm.com (9.42.237.131).
Escape character is '^]'.
Last login: Wed Apr 19 14:45:45 from repuda.raleigh.ibm.com

Username: USERID

Password:

=== SM CLP v1.0.0 SM ME Addressing v1.0.0 IICM Implementation v1.0.0 ===

->
```

The SMASH Proxy has been tested with the following Telnet clients:

- The Telnet clients distributed with the Red Hat 4.1, SLES 9, and Microsoft® Windows® XP operating systems (see your operating-system documentation for information).
- PuTTy (for details, see the PuTTy Web site [www.chiark.greenend.org.uk/~sgtatham/putty/]).

Additional Telnet clients are available and should work with the SMASH Proxy; however, support for these Telnet clients is not implied.

Accessing the SMASH Proxy: Local user access

You can access the SMASH Proxy locally on the SMASH proxy management station by establishing a Telnet or SSH session to the local host address of 127.0.0.1.

Uninstalling the SMASH Proxy

To uninstall the SMASH Proxy:

1. Log in as a user with administrative authority (for example, root).
2. Run the script [smash install directory]/_uninst/smashProxyUninstall.sh where [*smash install directory*] is the product's base directory (default is /opt/ibm/smashProxy). To prevent this script from being run inadvertently, it is originally set with no run file permission. Therefore, it should be run with a command such as: bash smashProxyUninstall.sh.

Chapter 6. Using SMASH Proxy functionality

This section details SMASH Proxy functionality and includes discussion of the following topics:

OEM verbs

oemiicmdiscover

This section contains descriptions for OEM verbs.

The general form of the OEM discovery verb, **oemiicmdiscover**, is:

```
oemiicmdiscover [options]
```

When run from the SMASH CLP shell, **oemiicmdiscover** has the following command line syntax:

```
oemiicmdiscover [-address arg] [-examine] [-help] [-output arg] [-version] [list]
```

which allows **oemiicmdiscover** to conform to the SMASH CLP standard. However, to conform to the POSIX standard, the command line syntax for **oemiicmdiscover** is as follows, when issued from the BASH shell:

```
oemiicmdiscover [--address arg] [--consoleoutput] [--debug] [--help] [--list]
```

POSIX requires that you prepend full name options (options that aren't single characters) with two dashes. You prepend single-character options in POSIX with a single dash. In the SMASH CLP standard, you prepend both full-name and single-character options with a single dash.

The **oemiicmdiscover** verb is used to discover BladeCenter chassis in the network for management and must be run after SMASH Proxy installation and before any other system management tasks can be attempted. If you specify the address option, the implementation attempts to find all targets located at the IP address specified by the option. If you do not specify the address option, the

implementation attempts to find all targets using the default SLP multicast address (239.255.255.253). For details on SLP configuration, see "SLP configuration in the MM" on page 44.

Options

The **oemiicmdiscover** verb supports the following standard options:

```
-examine
-version
-output
-help
```

The **oemiicmdiscover** verb supports the following OEM options:

```
-address (IP address for SLP discovery)
-list
```

Where the syntax is:

```
oemiicmdiscover –address [xxx.xxx.xxx.xxx]
oemiicmdiscover -list
```

You use the `address` option to control the IP address that the implementation searches. If you specify the `address` option, the implementation performs an SLP discovery using the address specified by the option. If you do not specify the `address` option, the implementation performs an SLP discovery using the default SLP multicast address. The argument to the `address` option identifies a subnet to which a broadcast should be issued, or an IP address to which a unicast is issued. There are three forms the argument to the `address` option can take:

```
->oemiicmdiscover -address 255.255.255.255
(broadcasts to local subnet [assuming broadcast is
disabled across subnets])

->oemiicmdiscover –address x.x.x.255
(broadcasts to subnet x.x.x)

->oemiicmdiscover –address 9.1.1.1
(unicasts to 9.1.1.1)
```

You use the `list` option to see the chassis that have previously been discovered. When you do not use the `list` option, no new discovery is performed.

Output

Text format

If one or more chassis are discovered, the implementation returns information about the discovered chassis. If no objects are detected, the implementation indicates this result.

XML format

The output contains a list of elements identifying modular targets that responded to the SLP query:

```
-> oemiicmdiscover -o format=clpxml
[?xml version="1.0" encoding="utf-8"?]
[response xmlns="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_Response.xsd" xmlns
xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:iicm="http://iicm.com/sma
hclp-extensions" xsi:schemaLocation="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML
Response.xsd CLPXML_Response.xsd"]
    [command]
        [inputline]oemiicmdiscover -o format=clpxml[/inputline]
```

```
[/command]
[cmdstat]
    [status]0[/status]
    [status_tag]COMMAND COMPLETED[/status_tag]
    [job]
        [job_id]4[/job_id]
    [/job]
[/cmdstat]
[oemverb]
    [verbname]oemiicmdiscover[/verbname]
    [oemdata]
        [iicm:target]
            [iicm:ufit ufct="modular" instance="1"]modular1[/iicm:ufit]
            [iicm:ipaddress]
                10.2.0.90
            [/iicm:ipaddress]
            [iicm:name]
                WMN315747695
            [/iicm:name]
        [/iicm:target]
        [iicm:target]
            [iicm:ufit ufct="modular" instance="2"]modular2[/iicm:ufit]
            [iicm:ipaddress]
                9.42.236.14
            [/iicm:ipaddress]
            [iicm:name]
                ux-bc-mm3 [1]
            [/iicm:name]
        [/iicm:target]
        [iicm:target]
            [iicm:ufit ufct="modular" instance="3"]modular3[/iicm:ufit]
            [iicm:ipaddress]
                9.42.236.167
            [/iicm:ipaddress]
            [iicm:name]
                WMN478796124
            [/iicm:name]
        [/iicm:target]
        [iicm:target]
            [iicm:ufit ufct="modular" instance="4"]modular4[/iicm:ufit]
            [iicm:ipaddress]
                9.42.236.65
            [/iicm:ipaddress]
            [iicm:name]
                ux-bc-mm2
            [/iicm:name]
        [/iicm:target]
        [iicm:target]
            [iicm:ufit ufct="modular" instance="5"]modular5[/iicm:ufit]
            [iicm:ipaddress]
                9.42.236.123
            [/iicm:ipaddress]
            [iicm:name]
                NETUS10x
            [/iicm:name]
        [/iicm:target]
        [iicm:target]
            [iicm:ufit ufct="modular" instance="6"]modular6[/iicm:ufit]
            [iicm:ipaddress]
                9.42.236.122
            [/iicm:ipaddress]
            [iicm:name]
                cebbladectr1
            [/iicm:name]
```

```
            [/iicm:target]
          [/oemdata]
      [/oemverb]
  [/response]
```

Each instance of a left bracket ([) in the above example represents a *less than* (<) symbol and each instance of a right bracket (]) represents a *more than* (>) symbol. On screen, you see the < and > symbols, not brackets (for example, </show>). The left and right brackets are used for documentation purposes only.

Note: The clpxml format conforms to the SM CLP Command Response XML Schema (www.dmtf.org/apps/org/workgroup/svrmgmt/download.php/17388/dsp0224.xsd).

Examples

The following examples illustrate the usage of the **oemiicmdiscover** verb:

```
-> oemiicmdiscover
Success
modular1        10.2.0.90       WMN315747695
modular2        9.42.236.14     ux-bc-mm3 [1]
modular3        9.42.236.167    WMN478796124
modular4        9.42.236.123    NETUS10x
modular5        9.42.236.140    Telco
modular6        9.42.236.122    cebbladectr1
modular7        9.42.236.65     ux-bc-mm2

-> oemiicmdiscover -address 10.2.0.255
Success
modular1        10.2.0.90       WMN315747695

-> oemiicmdiscover -list
Success
modular1        10.0.0.10       Telco
modular2        10.0.0.20       Enterprise
```

oemiicmremovechassis

The general form of the OEM remove chassis verb, **oemiicmremovechassis**, is:

```
-> oemiicmremovechassis [options] [targets]
```

The **oemiicmremovechassis** verb allows you to delete an individual chassis or all chassis in your admin domain. You can use the verb to reset the SMASH inventory of manageable chassis or to remove a particular chassis that has been taken permanently offline.

Valid Targets

The only valid target for the **oemiicmremovechassis** verb is a chassis or modular UFiT or UFsT. This UFiT or UFsT is the physical or logical component that represents the BladeCenter chassis.

Options

There are no OEM options defined for this OEM verb. Of the standard options, the following options are supported:

```
-examine
-version
```

```
-output
-help
```

Output

Output returns success or failure as appropriate and provide the UFiTs of the deleted chassis in the verb results.

Text format

If one or more chassis are deleted, the implementation will return information about the deleted object or objects.

XML format

The output contains a list of UFiP elements identifying modular or chassis targets that were deleted.

```
-> oemiicmremovechassis -o format=clpxml /modular*
[?xml version="1.0" encoding="utf-8"?]
[response xmlns="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_Response.xsd" xmlns:
xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:iicm="http://iicm.com/smas
hclp-extensions" xsi:schemaLocation="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_
Response.xsd CLPXML_Response.xsd"]
    [command]
        [inputline]oemiicmremovechassis -o format=clpxml /modular*[/inputline]
    [/command]
    [cmdstat]
        [status]0[/status]
        [status_tag]COMMAND COMPLETED[/status_tag]
        [job]
            [job_id]5[/job_id]
        [/job]
    [/cmdstat]
    [oemverb]
        [verbname]oemiicmremovechassis[/verbname]
        [oemdata]
            [iicm:target]
                [iicm:ufit ufct="modular" instance="1"]modular1[/iicm:ufit]
            [/iicm:target]
            [iicm:target]
                [iicm:ufit ufct="modular" instance="2"]modular2[/iicm:ufit]
            [/iicm:target]
            [iicm:target]
                [iicm:ufit ufct="modular" instance="3"]modular3[/iicm:ufit]
            [/iicm:target]
        [/oemdata]
    [/oemverb]
[/response]
```

Each instance of a left bracket ([) in the above example represents a *less than* (<) symbol and each instance of a right bracket (]) represents a *more than* (>) symbol. On screen, you see the < and > symbols, not brackets (for example, </show>). The left and right brackets are used for documentation purposes only.

Note: The clpxml format conforms to the SM CLP Command Response XML Schema (www.dmtf.org/apps/org/workgroup/svrmgmt/download.php/17388/dsp0224.xsd).

Examples

The following examples illustrate the usage of the oemiicmremovechassis verb:

```
-> oemiicmremovechassis /hdwr1/chassis1
Success
Removed chassis:
UFiT: chassis1

-> oemiicmremovechassis /modular3
Success
Removed chassis:
UFiT: modular3

-> oemiicmremovechassis /modular*
Success
Removed chassis:
UFiT: modular2
UFiT: modular6
UFiT: modular5
UFiT: modular4
UFiT: modular7
```

oemiicmlogin

The general form of the OEM login verb, **oemiicmlogin**, is:

`oemiicmlogin [options]`

You use the **oemiicmlogin** verb to establish new credentials for the user session.

Valid Targets

The **oemiicmlogin** verb is for the user session in general and does not operate against any user-specified target.

Options

The **oemiicmlogin** verb supports the following standard options:

`-examine`

`-version`

`-output`

`-help`

The **oemiicmlogin** verb supports the following OEM options:

`-userid`

`-password`

The syntax is as follows:

`oemiicmlogin –userid [userid] -password [password]`

When you run the **oemiicmlogin** verb, the credentials are set as *active*. No validation or authentication is performed at this time.

For details on credentials handling, see “Handling chassis credentials” on page 103.

Output

Text format

The implementation returns the currently active user ID.

XML format

```
-> oemiicmlogin -userid USERID -password PASSW0RD -o format=clpxml
[?xml version="1.0" encoding="utf-8"?]
[response xmlns="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_Response.xsd" xmlns
xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:iicm="http://iicm.com/sma
hclp-extensions" xsi:schemaLocation="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML
Response.xsd CLPXML_Response.xsd"]
    [command]
        [inputline]oemiicmlogin -userid USERID -password PASSW0RD -o format=clp
ml[/inputline]
    [/command]
    [cmdstat]
        [status]0[/status]
        [status_tag]COMMAND COMPLETED[/status_tag]
        [job]
            [job_id]1[/job_id]
        [/job]
    [/cmdstat]
    [oemverb]
        [verbname]oemiicmlogin[/verbname]
        [oemdata]
            [iicm:name]USERID[/iicm:name]
        [/oemdata]
    [/oemverb]
[/response]
```

Each instance of a left bracket ([) in the above example represents a *less than* (<) symbol and each instance of a right bracket (]) represents a *more than* (>) symbol. On screen, you see the < and > symbols, not brackets (for example, </show>). The left and right brackets are used for documentation purposes only.

Note: The clpxml format conforms to the SM CLP Command Response XML Schema (www.dmtf.org/apps/org/workgroup/svrmgmt/download.php/17388/dsp0224.xsd).

Examples

The following example illustrates the usage of the **oemiicmlogin** verb:

```
-> oemiicmlogin -userid USERID -password PASSW0RD
Success
Current username is USERID
```

oemiicmlogoff

The general form of the OEM logoff verb, **oemiicmlogoff**, is:

```
oemiicmlogoff [options]
```

You use the **oemiicmlogoff** verb to return to the previous set of credentials for the user.

Valid targets

The **oemiicmlogoff** verb is for the user session in general and does not operate against any user-specified target.

Options

The **oemiicmlogoff** verb supports the following options:

```
-examine
```

```
-help
-output
-version
```

It does not support OEM options.

The syntax is as follows:

```
oemiicmlogoff
```

Output

Output returns success or failure as appropriate. SMASH Proxy returns no special data. Failure conditions follow the CLP specification for success or failure output format.

Text format

The implementation returns the currently active user ID.

XML format

```
-> oemiicmlogoff -o format=clpxml
[?xml version="1.0" encoding="utf-8"?]
[response xmlns="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_Response.xsd" xmlns:
xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:iicm="http://iicm.com/smas
hclp-extensions" xsi:schemaLocation="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_
Response.xsd CLPXML_Response.xsd"]
    [command]
        [inputline]oemiicmlogoff -o format=clpxml[/inputline]
    [/command]
    [cmdstat]
        [status]0[/status]
        [status_tag]COMMAND COMPLETED[/status_tag]
        [job]
            [job_id]2[/job_id]
        [/job]
    [/cmdstat]
    [oemverb]
        [verbname]oemiicmlogoff[/verbname]
        [oemdata]
            [iicm:name]USERID[/iicm:name]
        [/oemdata]
    [/oemverb]
[/response]
```

Each instance of a left bracket ([) in the above example represents a *less than* (<) symbol and each instance of a right bracket (]) represents a *more than* (>) symbol. On screen, you see the < and > symbols, not brackets (for example, </show>). The left and right brackets are used for documentation purposes only.

Examples

The following example illustrates usage of the **oemiicmlogoff** verb.

```
-> oemiicmlogoff
Success
Current username is USERID1
```

SMASH Proxy functions

A user can run any supported verb on any supported target with any supported verb options and any target property filters. This results in thousands of possible commands. The following tables display the most commonly-used commands from this large assortment of possibilities.

Note: Available SMASH Proxy verbs, targets, verb options, and target properties, can be found in the following sections:

- "SMASH CLP supported verbs and descriptions" on page 8 and "OEM verbs" on page 49
- "SMASH CLP supported verb options and descriptions" on page 9
- "SMASH Proxy supported targets (by UFcT) and associated command target properties" on page 72 and "SMASH Proxy command target property descriptions" on page 94
- "SMASH Proxy nonaddressing associations and supported physical and logical targets" on page 64

Table 14. Configuration functions

Function	Command	Notes
Set IP address and subnet mask for management module	set /modularX/chassismgr1/ipendpt1 IPv4Address=[a.a.a.a] SubnetMask=[b.b.b.b]	Where X can be any UFiT index returned by a show /modular* command.
Set gateway address	set /modularX/chassismgr1/gateway1 for management module AccessInfo=[a.a.a.a]	Where X can be any UFiT index returned by a show /modular* command.
Set IP address and subnet mask for switch	set /modularX/switchY/ipendpt1 IPv4Address=[a.a.a.a] SubnetMask=[b.b.b.b]	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
Set gateway address for switch	set /modularX/switchY/gateway1 AccessInfo=[a.a.a.a]	Where X can be any UFiT index returned by a show /modular* and Y corresponds to the switch module bay.
Enable or disable external ports for switch	set /modularX/switchY/oemiicmswitchmgt1 ExternalPortsEnabled=true OR set /modularX/switchY/ oemiicmswitchmgt1 ExternalPortsEnabled=false	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
Show the current time on the MM	show –display properties=systemtime /modularX/chassismgr1	Where X can be any UFiT index returned by a show /modular* command.

Table 14. Configuration functions (continued)

Function	Command	Notes
Set the time and date on the MM	set /modularX/chassismgr1 systemtime="[MM/DD/YYYY] [HH:MM:SS] [<+|->TZMM]	Where X can be any UFiT index returned by a show /modular* command. MM = Month, DD = Day, YYYY = Year, HH = Hour, MM = Minutes, SS = Seconds, and TZ = the Timezone offset from Greenwich mean time (GMT) in minutes. For example -240, +120, and so on. A sample command would be the following: `set /modular1/chassismgr1 systemtime="05/19/2006 02:50:00-240"`
Switch ownership of the media tray to a particular blade	set /modularX/systemY/devicetray1 CurrentAccess="Exclusive Access"	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.
See which blade currently owns the media tray	show -level all -display properties=(CurrentAccess, CurrentAccess=="Exclusive Access") /modularX	Where X can be any UFiT index returned by a show /modular* command

Table 15. Modular system functions

Function	Command	Notes
Show all modular systems that can be managed	show /modular*	
See information on all MMs in a chassis	show /modularX/chassismgr*	Where X can be any UFiT index returned by a show /modular* command.
See information for a specific MM	show /modularX/chassismgrY	Where X can be any UFiT index returned by a show /modular* command and Y can be 1 or 2, corresponding to the active and failover MMs, respectively.
See information on all switches in a chassis	show /modularX/switch*	Where X can be any UFiT index returned by a show /modular* command.
See information for a specific switch	show /modularX/switchY	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
See information on all blades in a chassis	show –display properties=(StatusDescriptions,HealthState, OperationalStatus,Dedicated=="Not Dedicated") /modularX/system*	Where X can be any UFiT index returned by a show /modular* command.
See information for a specific blade	show /modularX/systemY	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.
See information for a blade service processor	show /modularX/systemY/sp1	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.
See information for the media tray	show /modularX/devicetray1	Where X can be any UFiT index returned by a show /modular* command.

Table 16. Physical assets functions

Function	Command	Notes
Show all chassis that can be managed	show /hdwr1/chassis*	
See information for a specific chassis	show /hdwr1/chassisX	Where X can be any UFiT index returned by a show /hdwr1/chassis* command.
See information on all MMs in a chassis	show /hdwr1/chassisX/modulepkg*	Where X can be any UFiT index returned by a show /hdwr1/chassis* command.
See information for a specific MM	show /hdwr1/chassisX/modulepkgY	Where X can be any UFiT index returned by a show /hdwr1/chassis*command and Y can be 1 or 2, corresponding to the MM bay.
See information on all switches in a chassis	show –display properties=(OtherIdentifyingInfo, PartNumber,SKU,ManufactureDate,Version, PoweredOn,ManufactureDate, VendorCompatibilityStrings==”“IICM:BC:IO module”) /hdwr1/chassisX/pkg*	Where X can be any UFiT index returned by a show /hdwr1/chassis* command.
See information for a specific switch	show /hdwr1/chassisX/pkgY	Where X can be any UFiT index returned by a show /hdwr1/chassis* command and Y corresponds to the switch module bay.
See information on all blades in a chassis	show /hdwr1/chassisX/bladepkg*	Where X can be any UFiT index returned by a show /hdwr1/chassis* command.
See information for a specific blade	show /hdwr1/chassisX/bladepkgY	Where X can be any UFiT index returned by a show /hdwr1/chassis* command and Y corresponds to the blade slot.
See information for the media tray	show /hdwr1/chassisX/storagepkg1	Where X can be any UFiT index returned by a show /hdwr1/chassis* command.
See information for a fan	show /hdwr1/chassisX/fanpkgY	Where X can be any UFiT index returned by a show /hdwr1/chassis* command and Y can be 1-2, corresponding to the fans in bays 1-2, respectively.
See information for a power supply	show /hdwr1/chassisX/pwrpkgY	Where X can be any UFiT index returned by a show /hdwr1/chassis* command and Y can be 1-2, corresponding to the power supplies in bays 1-2, respectively
See information for a blade expansion unit	show /hdwr1/chassisX/bladepkgY/bladexpkg1	Where X can be any UFiT index returned by a show /hdwr1/chassis* command and Y corresponds to the blade slot.
Show all daughter cards on a blade	show /hdwr1/chassisX/bladepkgY/card*	Where X can be any UFiT index returned by a show /hdwr1/chassis* command and Y corresponds to the blade slot.

Table 17. Discovery functions

Function	Command	Notes
Discover all chassis in all subnets enabled for multicast	oemiicmdiscover	Uses default SLP multicast address of 239.255.255.253.
Discover all chassis on the local subnet	oemiicmdiscover –address 255.255.255.255	
Discover all chassis on a particular subnet	oemiicmdiscover –address x.x.x.255	Where x.x.x denotes the subnet.

Table 17. Discovery functions (continued)

Function	Command	Notes
Discover a chassis at a particular IP address	oemiicmdiscover –address x.x.x.x	Where x.x.x.x denotes the IP address of a particular management module.
List all chassis that have previously been discovered	oemiicmdiscover -list	

These sample commands using the discovery function assume that a user is running in a typical customer environment where routers are configured to block broadcasts across subnets and to enable multicast across subnets. If broadcast is enabled across subnets, then the command

oemiicmdiscover `–address 255.255.255.255`

discovers all chassis in all subnets. Similarly, if multicast is disabled across subnets, then the command

oemiicmdiscover

only discovers chassis in the local subnet.

Table 18. Access control functions

Function	Command	Notes
Change active credentials	oemiicmlogin –userid [userid] -password [password]	
Revert to previous credentials	oemiicmlogoff	

Table 19. Firmware update functions

Function	Command	Notes
Find out what kind of firmware updates are supported through the MM	show -display properties=(SupportedSynchronousActions, SupportedAsynchronousActions, SupportedTargetTypes) /modularX/chassismgr1/capabilities1/swinstallsvccap1	Where X can be any UFiT index returned by a show /modular* command.
Update firmware on the active MM	load –sourcetftp://[a.a.a.a]/[firmware file path] /modularX/chassismgr1	Where X can be any UFiT index returned by a show /modular* command.
Update the Blade Systems Management Processor firmware on a blade server	load –sourcetftp://[a.a.a.a]/[firmware file path] /modularX/systemY/sp1	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.
Update the firmware on a copper or optical pass-through module	load –sourcetftp://[a.a.a.a]/[firmware file path] /modularX/switchY	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
Query to see if a previously started firmware update is complete	show /map1/jobs1/jobX	Where X is the job ID returned when the firmware update was kicked off.

Table 20. Software identity functions

Function	Command	Notes
Find out what firmware levels are currently installed on a MM	show /modularX/chassismgrY/swinv1/swid*	Where X can be any UFiT index returned by a show /modular* command.
Query the level of a specific type of MM firmware	show –display properties=(RevisionNumber, VersionString, ReleaseDate, OEMIICM_FileName, Name, Name==”[type of firmware]”) /modularX/chassismgrY/swinv1/swid*	Where [type of firmware] can be one of the following values: “Main App” “Boot ROM” “Remote Control” “PS2toUSBController” “USBIntf”. **Note:** A particular MM can only support a subset of these firmware types.
Find out what firmware level is currently installed on a switch	show /modularX/switchY/swinv1/swid*	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
Query the level of a specific type of switch firmware	show –display properties=(RevisionNumber, VersionString, ReleaseDate, OEMIICM_FileName, Name, Name==”[type of firmware]”) /modularX/switchY/swinv1/swid*	Where [type of firmware] can be one of the following values: “Main App 1” “Main App 2” “Boot ROM” and where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
Find out what firmware level is currently installed on a blade server	show /modularX/systemY/swinv1/swid*	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.
Query the level of a specific type of blade firmware	show –display properties=(RevisionNumber, VersionString, ReleaseDate, OEMIICM_FileName, Name, Name==”[type of firmware]”) /modularX/systemY/swinv1/swid*	Where [type of firmware] can be one of the following values: “BIOS” “Diagnostics” “Blade Systems Management Processor”, X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.

Table 21. Text console redirection functions

Function	Command	Notes
Start a text console redirection session to a blade	start /modularX/chassismgr1/textredirectsapY	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.
End a text console redirection session to a blade	Esc)	

For details on the use of text console redirection functions, see “Administering text console redirection” on page 105.

Table 22. Sensor functions

Function	Command	Notes
See if there is a blade in a given bay	show /modularX/presencesensorY	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade bay number.

Table 22. Sensor functions (continued)

Function	Command	Notes
See if there is a MM in a given bay	show /modularX/presencesensor10Y	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the management module bay number.
See if there is a switch in a given bay	show /modularX/presencesensor20Y	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch bay number.
See if there is a power supply in a given bay	show /modularX/presencesensor30Y	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the power supply bay number.
See if there is a fan in a given bay	show /modularX/presencesensor40Y	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the fan bay number.
See if media tray exists	show /modularX/presencesensor501	Where X can be any UFiT index returned by a show /modular* command.
See all BladeCenter numeric sensors	show /modularX/sensor*	Where X can be any UFiT index returned by a show /modular* command.
See all BladeCenter temperature sensors	show /modularX/ntempsensor*	Where X can be any UFiT index returned by a show /modular* command.
See all BladeCenter voltage sensors	show /modularX/nvoltsensor*	Where X can be any UFiT index returned by a show /modular* command.
See all BladeCenter presence sensors	show /modularX/presencesensor*	Where X can be any UFiT index returned by a show /modular* command.
See if a blade has a blade expansion unit	show /modularX/systemY/presencesensor1	Where X can be any UFiT index returned by a show /modular* command and Y is the blade slot.
Find out the temperature of the chassis front panel	show –display properties=(CurrentReading) /modularX/ntempsensor1	Where X can be any UFiT index returned by a show /modular* command.
Find out the temperature of the active MM	show -d properties=CurrentReading /modularX/ntempsensor2	Where X can be any UFiT index returned by a show /modular* command.
Find out the voltage readings for the active MM	show –display properties=(CurrentReading) /modularX/nvoltsensor*	Where X can be any UFiT index returned by a show /modular* command.
Find out what percentage of the maximum speed a fan is running at	show –display properties=(CurrentReading) /modularX/ntachsensorY	Where X can be any UFiT index returned by a show /modular* command and Y represents the bay where the fan resides.
Find out the temperature of the blade processors	show –display properties=(CurrentReading) /modularX/systemY/ntempsensor*	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot. **Note:** A blade must be powered on to get valid sensor readings.
Find out the voltage readings for the blade	show –display properties=(CurrentReading) /modularX/systemY/nvoltsensor*	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot. **Note:** A blade must be powered on to get valid sensor readings.

Table 23. Record log functions

Function	Command	Notes
See all records in the MM event log with the most-recent record first	show /modularX/chassismgr1/logs1/log1/record*	Where X can be any UFiT index returned by a show /modular* command.
See all records in the MM event log with the oldest record first	show –output order=reverse /modularX/ chassismgr1/logs1/log1/record*	Where X can be any UFiT index returned by a show /modular* command.
See the *n* oldest records in the MM event log with the most-recent record first	show –output end,count=n /modularX/ chassismgr1/logs1/log1/record*	Where X can be any UFiT index returned by a show /modular* command.
See only the log records with a certain severity level	show –display properties=(RecordData,OEMIICM_Sever ity==″[Severity Level]″) /modularX/chassismgr1/logs1/log1/record*	Where [Severity Level] can be one of the following values: INFO,ERR, WARN and X can be any UFiT index returned by a show /modular* command.
See the log records for a certain source	show –display properties=(RecordData,OEMIICM_Sour ce==″[Source]″) /modularX/chassismgr1/logs1/log1/record*	Where [Source] can be one of the following values: SERVPROC BLADE_Y, where Y corresponds to the blade slot. X can be any UFiT index returned by a show /modular* command.
See the log records for a particular system	show –display properties=(RecordData,OEMIICM_Nam e==″[System Name]″) /modularX/chassismgr1/logs1/log1/record*	Where X can be any UFiT index returned by a show /modular* command.
Clear the MM event log	delete /modularX/chassismgr1/logs1/log1/record* or delete /modularX/chassismgr1/logs1/log1	Where X can be any UFiT index returned by a show /modular* command.

Table 24. Power control functions

Function	Command	Notes
Power on, off, or reset blade	start /modularX/systemY stop /modularX/systemY reset /modularX/systemY	Where X can be any UFiT index returned by a show /modular* command and Y is the blade slot.
Power on, off, or reset switch	start /modularX/switchY stop /modularX/switchY reset /modularX/switchY	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the switch module bay.
Power reset management module	reset /modularX/chassismgr1	Where X can be any UFiT index returned by a show /modular* command.
See which blades are powered on	show -d properties=(StatusDescriptions==″Powe red On″,Dedicated==″Not Dedicated″) /modularX/system*	Where X can be any UFiT index returned by a show /modular* command.
Failover to redundant MM	set /modularX/chassismgr2 RequestedState=″Enabled″	Where X can be any UFiT index returned by a show /modular* command.

Table 25. LED functions

Function	Command	Notes
See the status of all LEDs on the front panel of an Enterprise BladeCenter machine	show /modularX/oemiicmled*	Where X can be any UFiT index returned by a show /modular* command.
See the status of all LEDs on a blade	show /modularX/systemY/oemiicmled*	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the blade slot.

Table 25. LED functions (continued)

Function	Command	Notes
See the status of all LEDs on a copper pass-through module	show /modularX/switchY/oemiicmled*	Where X can be any UFiT index returned by a show /modular* command and Y corresponds to the bay where the copper pass-through module resides.

Table 26. Help functions

Function	Command	Notes
Get help on the SMASH implementation in general	help	
Get help on a particular verb	help [verb] or [verb] -help	
Get help on a particular target	help [target]	
Show the version of the SMASH implementation in general	version	

SMASH Proxy nonaddressing associations and supported physical and logical targets

As described in "Addressing association diagrams" on page 19, objects link to each other through associations. An *addressing association* means that you can use it to target an object with the following addressing format:

```
[parent object]/[object]
```

For example, admin1/hdwr1 .

Nonaddressing associations, however, cannot be used with a [parent object]/[object] syntax. For example, as shown in Table 27 on page 65, chassisX, with a UFiP of /hdwr1/chassisX, has a nonaddressing **SystemPackaging** association to modularX. Because it is a nonaddressing association, you cannot address modularX as /hdwr1/chassisX/modularX. Instead, you must address it as /modularX, directly. To target the **SystemPackaging** association itself, the address would be /hdwr1/chassisX=>SystemPackaging=>/modularX.

The following tables show all SMASH Proxy supported object types and their associated verbs and nonaddressing associations. For all tables:

- Each object is indicated by **bold** font.
- An object's subcomponent is indicated by `example` font.
- A subcomponents subcomponent is indicated by *italic* font.
- The next level of subcomponent is indicated by "quotes".
- Further granularity of a subcomponent is underlined.

Note: The file paths for some target UFiPs and targets of nonaddressing associations have been broken in order to fit them in the table. There are no spaces between UFiTs in UFiPs. They are in the form:

```
/modularX/capacities1/configcapacityY
```

Table 27. Physical target addressing: Chassis component

BladeCenter component	Physical target UFiT	Physical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Chassis	chassisX	/hdwr1/chassisX	SystemPackaging	/modularX	help, show, oemiicmremovechassis
			ConcreteDependency	/modularX/ntempsensor1	
Media tray	storagepkg1	/hdwr1/chassisX /storagepkg1	Realizes	/modularX/devicetray1	help, show
Management module	modulepkgY	/hdwr1/chassisX /modulepkgY	ComputerSystem Package	/modularX/chassismgrY	help, show
Power supply	pwrpkgY	/hdwr1/chassisX /pwrpkgY	none		help, show
Fan	fanpkgY	/hdwr1/chassisX /fanpkgY	none		help, show
Switch	pkgY	/hdwr1/chassisX /pkgY	ComputerSystem Package	/modularX/switchY	help, show
Blade	bladepkgY	/hdwr1/chassisX /bladepkgY	ComputerSystem Package	/modularX/systemY	help, show
Daughter card	card1	/hdwr1/chassisX /bladepkgY /card1	none		help, show
Blade expansion unit	bladexpkg1	/hdwr1/chassisX /bladepkgY /bladexpkg1	none		help, show

Table 28. Logical target addressing: Enterprise chassis or Telco chassis components (Temperature sensor, Fan speed sensor, Presence sensor, and Media tray subcomponents)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Enterprise or Telco chassis	modularX	modularX	SystemPackaging	/hdwr1/chassisX	help, show
			ConcreteDependency	/modularX/ntachsensorY OR /modularX/presencesensorY	
Temperature sensor	ntempsensor1	/modularX/ntempsensor1	ConcreteDependency	/hdwr1/chassisX	help, show
Fan speed sensor	ntachsensorY	/modularX/ntachsensorY	ConcreteDependency	/modularX	help, show
Presence sensors for blades	presencesensor1- presencesensor14	/modularX /presencesensor1 - /modularX /presencesensor14	ConcreteDependency	/modularX	help, show
Presence sensors for management modules	presencesensor101- presencesensor102	/modularX /presencesensor101 - /modularX /presencesensor102			
Presence sensors for switches and pass-thru modules	presencesensor201- presencesensor210	/modularX /presencesensor201 - /modularX /presencesensor210			
Presence sensors for power supplies	presencesensor301- presencesensor304	/modularX /presencesensor301 - /modularX /presencesensor304			
Presence sensor for fans	presencesensor401- presencesensor404	/modularX /presencesensor401 - /modularX /presencesensor404			
Presence sensor for media tray	presencesensor501	/modularX /presencesensor501			
Media tray	devicetray1	/modularX/devicetray1	Realizes	/hdwr1/chassisX /storagepkg1	help, show, set
			SharingDependency	/modularX/systemY /devicetray1	
			ServiceAffectsElement	/modularX /chassismgr1 /shareddevicesvc1	

Table 29. Logical target addressing: Enterprise chassis component (LEDs)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Identity	oemiicmled1	/modularX /oemiicmled1			help, show

Table 29. Logical target addressing: Enterprise chassis component (LEDs) (continued)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Temperature	oemiicmled2	/modularX /oemiicmled2			help, show
Information	oemiicmled3	/modularX /oemiicmled3			help, show
System error	oemiicmled4	/modularX /oemiicmled4			help, show

Table 30. Logical target addressing: Telco chassis component (LEDs)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Identity	oemiicmled1	/modularX /oemiicmled1			help, show
Critical	oemiicmled2	/modularX /oemiicmled2			help, show
Major	oemiicmled3	/modularX /oemiicmled3			help, show
Minor	oemiicmled4	/modularX /oemiicmled4			help, show

Table 31. Logical target addressing: Enterprise chassis or Telco chassis components (Redundant and active management module subcomponents)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Redundant management module	chassismgr2	/modularX/chassismgr2	ComputerSystem Package	/hdwr1/chassisX/modulepkgY	help, show
Active management module	chassismgr1	/modularX/chassismgr1	ComputerSystem Package	/hdwr1/chassisX/modulepkgY	help, load, reset, set, show
			ServiceAvailable ToElement	/modularX/chassismgr1 /swinstallsvc1	
			ServiceAffectsElement	/modularX/chassismgr1/timesvc1	
			ConcreteDependency	/modularX/ntempsensor2	
			ElementSoftwareIdentity	/modularX /chassismgr1 /swinv1/swidY	
			UseofLog	/modularX /chassismgr1 /logs1/log1	
Firmware	swinv1	/modularX/chassismgr1 /swinv1	none		help, show
"Main Application"	swid1	/modularX/chassismgr1 /swinv1/swid1	ElementSoftware Identity	/modularX/chassismgr1	help, show
"Boot ROM"	swid2	/modularX/chassismgr1 /swinv1/swid2	ElementSoftware Identity	/modularX/chassismgr1	help, show
"Remote control"	swid3	/modularX/chassismgr1 /swinv1/swid3	ElementSoftware Identity	/modularX/chassismgr1	help, show
"PS/2-to-USB controller"	swid4	/modularX/chassismgr1 /swinv1/swid4	ElementSoftware Identity	/modularX/chassismgr1	help, show
"USB interface"	swid5	/modularX/chassismgr1 /swinv1/swid5	ElementSoftware Identity	/modularX/chassismgr1	help, show
"Event log"	log1	/modularX/chassismgr1 /logs1/log1	UseofLog	/modularX/chassismgr1	delete, help, show
Event log record	recordZ	/modularX/chassismgr1 /logs/logs1/recordZ	none		delete, help, show

Note: Different management modules might have a different subset of the five swid types listed.

Table 32. Logical target addressing: Enterprise chassis or Telco chassis components (Active management module subcomponent)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Software installation					

Table 32. Logical target addressing: Enterprise chassis or Telco chassis components (Active management module subcomponent) (continued)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Service	swinstallsvc1	/modularX /chassismgr1 /swinstallsvc1	Service AvailableTo Element	/modularX/chassismgr1 OR /modularX/switchY OR /modularX/systemY/sp1	help, show
			ElementCapabilities	/modularX/chassismgr1 /swinstallsvccap1	
"Capabilities"	swinstallsvccap1	/modularX /chassismgr1 /capabilities1 /swinstallsvccap1	ElementCapabilities	/modularX/chassismgr1/ swinstallsvc1	help, show
Time	timesvc1	/modularX /chassismgr1 /timesvcl	ServiceAffectsElement	/modularX/chassismgr1	help, show
Sharing of media tray service	shareddevicesvcl	/modularX /chassismgr1 /shareddevicesvcl	ServiceAffectsElement	/modularX/devicetray1	help, show
Text console redirection					
Service	textredirectsvc1	/modularX /chassismgr1/text redirectsvc1	ServiceAccessBySAP	/modularX/chassismgr1 /textredirectsapZ	help, show
SOL session for blade in bay Z	textredirectsapZ	/modularX /chassismgr1 /textredirectsapZ	ServiceAccess AvailableBySAP	/modularX/chassismgr1 /textredirectsvc1	help, show, start
			SAPAvailable ForElement	/modularX/systemZ	
IP Address/Subnet mask	ipendpt1	/modularX /chassismgr1 /ipendpt1	RemoteAccess AvailableToElement	/modularX /chassismgr1 /gateway1	help, show, set
Gateway	gateway1	/modularX /chassismgr1 /gateway1	RemoteAccess AvailableToElement	/modularX /chassismgr1 /ipendpt1	help, show, set

Table 33. Logical target addressing: Enterprise chassis or Telco chassis components (Temperature and voltage sensor subcomponents)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing Associations	Target of nonaddressing associations	Verbs
Temperature sensor	ntempsensor2	/modularX /ntempsensor2	ConcreteDependency	/modularX/chassismgr1	help, show
	ntempsensor3	/modularX /ntempsensor3	ConcreteDependency	/modularX/chassismgr2	
Voltage sensors	nvoltsensor*	/modularX /nvoltsensor*	n/a		show
+5 V	nvoltsensor1	/modularX /nvoltsensor1	none		help, show
+3.3 V	nvoltsensor2	/modularX /nvoltsensor2	none		help, show
+12 V	nvoltsensor3	/modularX /nvoltsensor3	none		help, show
-5 V	nvoltsensor4	/modularX /nvoltsensor4	none		help, show
+2.5 V	nvoltsensor5	/modularX /nvoltsensor5	none		help, show
+1.8 V	nvoltsensor6	/modularX /nvoltsensor6	none		help, show

Table 34. Logical target addressing: Enterprise chassis or Telco chassis components (Switch subcomponent)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Switch	switchY	/modularX/switchY	Computer SystemPackage	/hdwr1/chassisX/pkgY	help, load, reset, set, show, start, stop
			ServiceAvailableTo Element	/modularX/chassismgr1 /swinstallsvc1	
			ElementSoftware Identity	/modularX/switchY /swinv1/swidZ	
			ServiceAffectsElement	/modularX/switchY /oemiicmswitchmgt1	
Firmware	swinv1	/modularX/switchY /swinv1	none		help, show

Table 34. Logical target addressing: Enterprise chassis or Telco chassis components (Switch subcomponent) (continued)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
"Main app 1"	swid1	/modularX/switchY /swinv1/swid1	ElementSoftware Identity	/modularX/switchY	help, show
"Main app 2"	swid2	/modularX/switchY /swinv1/swid2	ElementSoftware Identity	/modularX/switchY	help, show
"Boot ROM"	swid3	/modularX/switchY /swinv1/swid3	ElementSoftware Identity	/modularX/switchY	help, show
IP address/subnet mask	ipendpt1	/modularX /switchY/ipendpt1	RemoteAccess AvailableToElement	/modularX/switchY/ gateway1	help, show, set
Gateway	gateway1	/modularX/switchY /gateway1	RemoteAccess AvailableToElement	/modularX/switchY/ ipendpt1	help, show, set
External port status	oemiicmswitc hmgt1	/modularX/switchY /oemiicmswit chmgt1	ServiceAffectsElement	/modularX/switchY	help, show, set

Table 35. Logical target addressing: Enterprise or Telco chassis component (Switch subcomponent - LEDs [pass-thru module only])

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
External port 1	oemiicmled1	/modularX/switchY /oemiicmled1			help, show
External port 2	oemiicmled2	/modularX/switchY /oemiicmled2			help, show
External port 3	oemiicmled3	/modularX/switchY /oemiicmled3			help, show
External port 4	oemiicmled4	/modularX/switchY /oemiicmled4			help, show
External port 5	oemiicmled5	/modularX/switchY /oemiicmled5			help, show
External port 6	oemiicmled6	/modularX/switchY /oemiicmled6			help, show
External port 7	oemiicmled7	/modularX/switchY /oemiicmled7			help, show
External port 8	oemiicmled8	/modularX/switchY /oemiicmled8			help, show
External port 9	oemiicmled9	/modularX/switchY /oemiicmled9			help, show
External port 10	oemiicmled10	/modularX/switchY /oemiicmled10			help, show
External port 11	oemiicmled11	/modularX/switchY /oemiicmled11			help, show
External port 12	oemiicmled12	/modularX/switchY /oemiicmled12			help, show
External port 13	oemiicmled13	/modularX/switchY /oemiicmled13			help, show
External port 14	oemiicmled14	/modularX/switchY /oemiicmled14			help, show
Internal port 1	oemiicmled15	/modularX/switchY /oemiicmled15			help, show
Internal port 2	oemiicmled16	/modularX/switchY /oemiicmled16			help, show
Internal port 3	oemiicmled17	/modularX/switchY /oemiicmled17			help, show
Internal port 4	oemiicmled18	/modularX/switchY /oemiicmled18			help, show
Internal port 5	oemiicmled19	/modularX/switchY /oemiicmled19			help, show
Internal port 6	oemiicmled20	/modularX/switchY /oemiicmled20			help, show
Internal port 7	oemiicmled21	/modularX/switchY /oemiicmled21			help, show
Internal port 8	oemiicmled22	/modularX/switchY /oemiicmled22			help, show
Internal port 9	oemiicmled23	/modularX/switchY /oemiicmled23			help, show
Internal port 10	oemiicmled24	/modularX/switchY /oemiicmled24			help, show
Internal port 11	oemiicmled25	/modularX/switchY /oemiicmled25			help, show

Table 35. Logical target addressing: Enterprise or Telco chassis component (Switch subcomponent - LEDs [pass-thru module only]) (continued)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Internal port 12	oemiicmled26	/modularX/switchY /oemiicmled26			help, show
Internal port 13	oemiicmled27	/modularX/switchY /oemiicmled27			help, show
Internal port 14	oemiicmled28	/modularX/switchY /oemiicmled28			help, show

Note: The switch LEDs are only available for the Copper pass-thru module and not for other switches.

Table 36. Logical target addressing: Enterprise chassis or Telco chassis components (Blade subcomponent, Part 1 - CPU temperature sensors)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Blade	systemY	/modularX/systemY	ComputerSystem Package	/hdwr1/chassisX /bladepkgY	help, reset, set, show, start, stop
			ConcreteDependency	/modularX/systemY /ntempsensorZ OR /modularX/systemY /nvoltsensorZ	
			ElementSoftware Identity	/modularX/systemY /swinv1/swidZ	
Blade expansion unit presence sensor	presencesensor1	/modularX /systemY /presencesensor1	ConcreteDependency	/hdwr1/chassisX /bladepkgY /bladexpkg1	help, show
CPU temperature sensors	ntempsensor*	/modularX /systemY /ntempsensor*	n/a		show
"CPU1"	ntempsensor1	/modularX/systemY /ntempsensor1	ConcreteDependency	/modularX/systemY	help, show
"CPU2"	ntempsensor2	/modularX/systemY /ntempsensor2	ConcreteDependency	/modularX/systemY	help, show
"CPU3"	ntempsensor3	/modularX/systemY /ntempsensor3	ConcreteDependency	/modularX/systemY	help, show
"CPU4"	ntempsensor4	/modularY/systemY /ntempsensor4	ConcreteDependency	/modularX/systemY	help, show
"CPU5"	ntempsensor5	/modularX/systemY /ntempsensor5	ConcreteDependency	/modularX/systemY	help, show
"CPU6"	ntempsensor6	/modularX/systemY /ntempsensor6	ConcreteDependency	/modularX/systemY	help, show

Table 37. Logical target addressing: Enterprise chassis or Telco chassis components (Blade subcomponent, Part 2 - Planar voltage sensors [non-IPMI blades])

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Planar voltage sensors (non-IPMI blades)	nvoltsensor*	/modularX /systemY /nvoltsensor*	n/a		show
"+5 V"	nvoltsensor1	/modularX /systemY /nvoltsensor1	ConcreteDependency	/modularX/systemY	help, show
"+3.3 V"	nvoltsensor2	/modularX /systemY /nvoltsensor2	ConcreteDependency	/modularX/systemY	help, show
"+12 V"	nvoltsensor3	/modularX /systemY /nvoltsensor3	ConcreteDependency	/modularX/systemY	help, show
"+2.5 V"	nvoltsensor4	/modularX /systemY /nvoltsensor4	ConcreteDependency	/modularX/systemY	help, show
"+1.5 V"	nvoltsensor5	/modularX /systemY /nvoltsensor5	ConcreteDependency	/modularX/systemY	help, show
"+1.25 V"	nvoltsensor6	/modularX /systemY /nvoltsensor6	ConcreteDependency	/modularX/systemY	help, show

Table 37. Logical target addressing: Enterprise chassis or Telco chassis components (Blade subcomponent, Part 2 - Planar voltage sensors [non-IPMI blades]) (continued)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
"VRM1"	nvoltsensor7	/modularX /systemY /nvoltsensor7	ConcreteDependency	/modularX/systemY	help, show

Table 38. Logical target addressing: Enterprise chassis or Telco chassis components (Blade subcomponent, Part 3 - Planar voltage sensors [IPMI blades])

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Planar voltage sensors (IPMI blades)	nvoltsensor*	/modularX/systemY /nvoltsensor*	n/a		show
	nvoltsensor1 - nvoltsensor25	/modularX/systemY /nvoltsensor1 - /modularX/systemY /nvoltsensor25	ConcreteDependency	/modularX/systemY	help, show

Table 39. Logical target addressing: Enterprise Chassis or Telco Chassis components (Blade subcomponent, Part 4 - Service processor, firmware, and device tray)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Service processor	sp1	/modularX/systemY /sp1	Service AvailableToElement	/modularX /chassismgr1 /swinstallsvc1	help, show, load
Firmware	swinv1	/modularX/systemY /swinv1	none		help, show
"BIOS"	swid1	/modularX/systemY /swinv1/swid1	Element SoftwareIdentity	/modularX/systemY	help, show
"Diagnostics"	swid2	modularX/systemY /swinv1/swid2	Element SoftwareIdentity	/modularX/systemY	help, show
"Blade system management processor"	swid3	modularX/systemY /swinv1/swid3	Element SoftwareIdentity	/modularX/systemY	help, show
Device tray	devicetray1	/modularX/systemY /devicetray1	SharingDependency	/modularX /devicetray1	help, set, show
			ElementCapabilites	/modularX/systemY /capabilities1 /sharingcap1	
"Device tray sharing capabilities"	sharingcap1	/modularX/systemY /capabilities1 sharingcap1	ElementCapabilites	/modularX/systemY /devicetray1	help, show

Table 40. Logical target addressing: Enterprise chassis or Telco chassis components (Blade subcomponent - LEDs)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
System Error	oemiicmled1	/modularX/systemY /oemiicmled1			help, show
Information	oemiicmled2	/modularX/systemY /oemiicmled2			help, show
Identity	oemiicmled3	/modularX/systemY /oemiicmled3			help, show
Power	oemiicmled4	/modularX/systemY /oemiicmled4			help, show
Media tray	oemiicmled5	/modularX/systemY /oemiicmled5			help, show
KVM	oemiicmled6	/modularX/systemY /oemiicmled6			help, show

Table 41. Supported collections

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Root for all SMASH administration	admin1	/admin1 OR /	none		help, show
Collection of all physical chassis	hdwr1	/hdwr1	none		help, show

Table 41. Supported collections (continued)

BladeCenter component	Logical target UFiT	Logical target UFiP	Nonaddressing associations	Target of nonaddressing associations	Verbs
Collection of all logs	logs1	/modularX /chassismgr1/logs1	none		help, show
Collection of all capabilities for a management module	capabilities1	/modularX/chassismgr1 /capabilities1	none		help, show
Collection of all capabilities for a blade	capabilities1	/modularX/systemY /capabilities1	none		help, show
Collection of all component capacities for a chassis	capacities1	/modularX/capacities1	none		help, show
Switch Capacity	configcapacity1	/modularX/capacities1 /configcapacity1	none		help, show
Management module capacity	configcapacity2	/modularX/capacities1 /configcapacity2	none		help, show
Fan capacity	configcapacity3	/modularX/capacities1 /configcapacity3	none		help, show
Blade capacity	configcapacity4	/modularX/capacities1 /configcapacity4	none		help, show
Power supply capacity	configcapacity5	/modularX/capacities1 /configcapacity5	none		help, show

Handling general UFcTs compared to specific UFcTs

The *Server Management Managed Element Addressing Specification* maps each CIM class modeled by an SMWG profile to a general UFcT. Additionally, a CIM instance can be mapped to a more specific UFcT based on the value of a particular property. For example, the CIM class CIM_ComputerSystem is mapped to the general UFcT:

```
system
```

However, if a computer system's instance has a Dedicated property equal to "Switch", it is given the specific UFcT:

```
switch
```

If its Dedicated property is equal to "Chassis Manager" (MM), it is given the specific UFcT:

```
chassismgr
```

If its Dedicated property is equal to "Management Controller" (Blade System Management Processor), it is given the specific UFcT:

```
sp
```

Within the SMASH Proxy, a user can identify and target objects by their most-specific UFcT. However, they also show up in a list of the general UFcTs. For example, if a user has two switches, one MM, and two blades in his chassis and does a **show** `-d targets=system /modular1`, the user sees the following output:

```
Success
UFiT: modular1
UFiT: chassismgr1
UFiT: switch1
UFiT: switch2
UFiT: switch3
UFiT: system1
UFiT: system2
UFiT: system4
UFiT: system6
UFiT: system7
```

where each system object shows up with its most specific UFcT. Given the output above, if a user wants to target a particular switch, he or she can specify:

show `/modular1/switch3`

He or she cannot specify:

show `/modular1/system3`

Note: A UFiT with `system` always only refers to a blade. That is, although `system*` can show switches and MMs, system1, system2...systemX are always blades.

General UFcTs are identified in the tables found in "SMASH Proxy supported targets (by UFcT) and associated command target properties."

SMASH Proxy supported targets (by UFcT) and associated command target properties

Command target *properties* are attributes that can contain values associated with a target that the SMASH Proxy needs to process the command. Command target properties identify properties of the target's class that the command retrieves or modifies.

For example, Table 42 displays the properties for the UFcT target **system** that are, by default, displayed when running a **show** command against that target.

Table 42. system UFcT properties

UFcT	Property	Property type	Possible values for property
system	*CreationClassName*	string	"IICM_ComputerSystem"
	Name	string	[Chassis Name]|[Instance Tag]
	NameFormat	string	"Other"
	HealthState	uint16 ValueMap	0 ("Unknown"), 5 ("OK"), 10 ("Degraded/Warning"), 20 ("Major Failure")
	StatusDescriptons	string[]	"Powered On", "Powered Off"
	OperationalStatus	uint16[] ValueMap	1 ("Other")
	Dedicated	uint16[] ValueMap	0 ("Not Dedicated")
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled")
	RequestedState	**uint16 ValueMap**	**2 ("Enabled"), 3 ("Disabled")**

Note: The *Chassis Name* (as seen above in `[Chassis Name]|[Instance Tag]`) is the name given to the BladeCenter server in the Web interface (**MM Control->General Settings->MM Information->Name**; see Figure 14 on page 73) and might or might not be the same as the Ethernet host name given to the BladeCenter server. In addition to this fact, it is important that you make sure that the name you select for each chassis is unique.

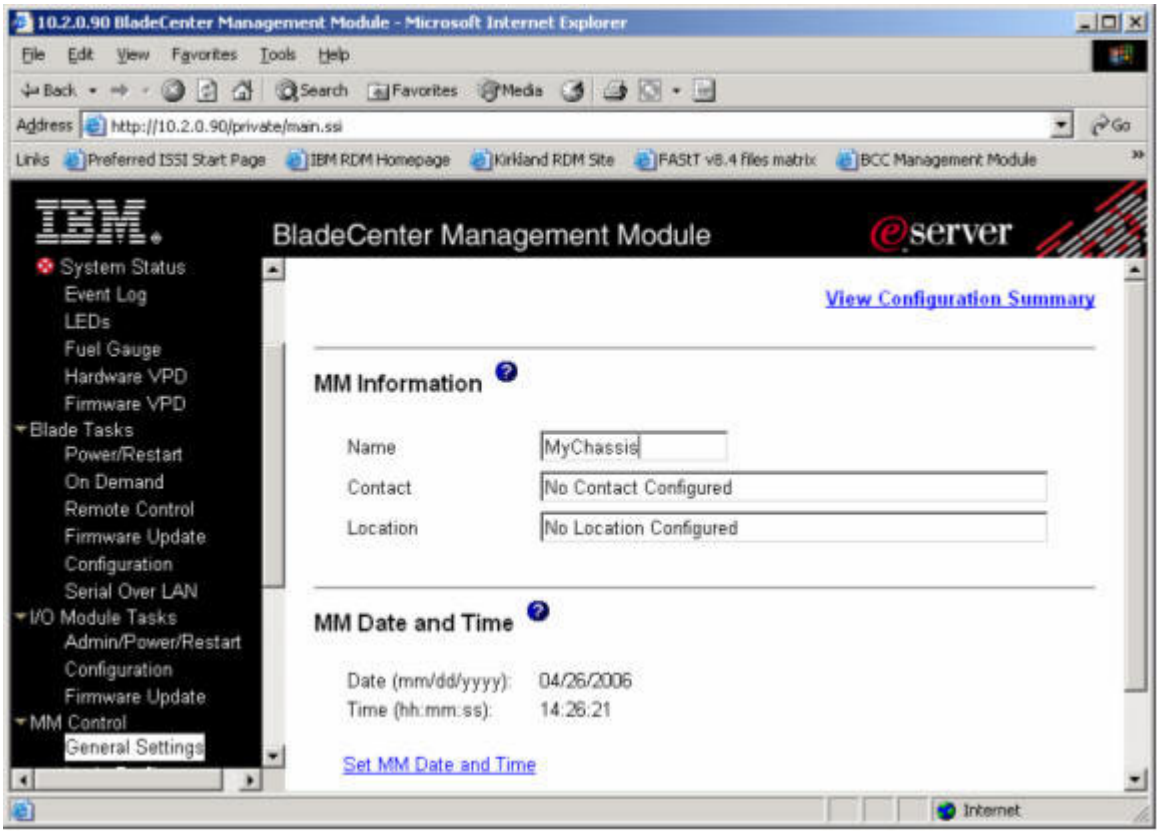

Figure 14. BladeCenter management module web interface - General settings

You can also use the **show** command as follows:

- You can limit the number of properties displayed by issuing a **show** command using the property name with the `display` or `–d` option. For example, if you only want to show the current HealthState value for system1, issue:

  ```
  show -d properties=HealthState system1
  ```

  which displays the following output:

  ```
  Success
  UFiT: system1
  UFiP: /modular1/system1
  Properties:
  HealthState: OK (5)
  ```

- You can control format of what is displayed by using the `output` option, or `–o`. For example,

  ```
  show -o format=clpxml -d properties=HealthState system1
  ```

  displays the following output:

  ```
  [?xml version="1.0" encoding="utf-8"?]
  [response xmlns="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_Response.xsd" xmlns:
  xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:iicm="http://iicm.com/smas
  hclp-extensions" xsi:schemaLocation="http://schemas.dmtf.org/SMASH/1.0.0/CLPXML_
  Response.xsd CLPXML_Response.xsd"]
      [command]
          [inputline]show -o format=clpxml -d properties=HealthState system1[/inpu
  tline]
      [/command]
      [cmdstat]
          [status]0[/status]
          [status_tag]COMMAND COMPLETED[/status_tag]
  ```

```
        [job]
            [job_id]2[/job_id]
        [/job]
    [/cmdstat]
    [show]
        [target]
            [instance]
                [ufit ufct="system" instance="1"]system1[/ufit]
                [ufip]/modular1/system1[/ufip]
                [properties]
                    [property]
                        [name]HealthState[/name]
                        [value]
                            [val]5[/val]
                            [valstring]OK[/valstring]
                        [/value]
                        [type]uint16[/type]
                        [description]Health State of ComputerSystem0=Unknown, 5=
OK, 10=Warning, 20=Error[/description]
                        [readonly]true[/readonly]
                        [valuemap]0, 5, 10, 15, 20, 25, 30, ..[/valuemap]
                        [values]Unknown, OK, Degraded/Warning, Minor failure, Ma
jor failure, Critical failure, Non-recoverable error, DMTF Reserved[/values]
                    [/property]
                [/properties]
            [/instance]
        [/target]
    [/show]
[/response]
```

Note: Each instance of a left bracket ([) in the above example represents a *less than* (<) symbol and each instance of a right bracket (]) represents a *more than* (>) symbol. On screen, you see the < and > symbols, not brackets (for example, </show>). The left and right brackets are used for documentation purposes only.

- You can show the values of several properties for an object by issuing a **show** command using this type of syntax:

  ```
  show –display properties=(HealthState,OperationalStatus) system1
  ```

  which displays the following output:

  ```
  Success
  UFiT: system1
  UFiP: /modular1/system1
  Properties:
  HealthState: OK (5)
  OperationalStatus:
  Other (1)
  ```

  Note: There are no spaces allowed between property names.

- You must specify the `all` or `–a` option if you want to show OEM targets, key properties or OEM properties. For example, issuing a **show** command without the `–a` option, such as

  ```
  show system1
  ```

  displays the following output:

  ```
  Success
  UFiT: system1
  UFiP: /modular1/system1
  Properties:Status
  Descriptions: Powered Off
  NameFormat: Other
  HealthState: OK (5)
  OperationalStatus:
  Other (1)
  ```

```
Dedicated:
Not Dedicated (0)
EnabledState: Disabled (3)
RequestedState: Disabled (3)
Verbs:
Standard: help load reset set show start stop
UFiT: capabilities1
UFiT: swinv1
UFiT: devicetray1
UFiT: presencesensor1
UFiT: ntempsensor1
UFiT: ntempsensor2
UFiT: ntempsensor3
UFiT: ntempsensor4
UFiT: nvoltsensor1
UFiT: nvoltsensor2
UFiT: nvoltsensor3
UFiT: nvoltsensor4
UFiT: nvoltsensor5
UFiT: nvoltsensor6
UFiT: nvoltsensor7
UFiT: sp1
```

However, with the –a option:

```
show –a system1
```

you see this output:

```
Success
UFiT: system1
UFiP: /modular1/system1
Properties:
StatusDescriptions:
Powered Off
NameFormat: Other
CreationClassName: IICM_ComputerSystem
Name: WMN315747695|blade__1
HealthState: OK (5)
OperationalStatus:
Other (1)
Dedicated:
Not Dedicated (0)
EnabledState: Disabled (3)
RequestedState: Disabled (3)
Verbs:
Standard: help load reset set show start stop
UFiT: capabilities1
UFiT: swinv1
UFiT: devicetray1
UFiT: presencesensor1
UFiT: ntempsensor1
UFiT: ntempsensor2
UFiT: ntempsensor3
UFiT: ntempsensor4
UFiT: nvoltsensor1
UFiT: nvoltsensor2
UFiT: nvoltsensor3
UFiT: nvoltsensor4
UFiT: nvoltsensor5
UFiT: nvoltsensor6
UFiT: nvoltsensor7
UFiT: oemiicmled1
UFiT: oemiicmled2
UFiT: oemiicmled3
```

```
UFiT: oemiicmled4
UFiT: oemiicmled5
UFiT: oemiicmled6
UFiT: sp1
```

(In the following tables, the key properties are shown in italic, such as *CreationClassName*. OEM properties are shown with an OEMIICM prefix, for example, *OEMIICM_Source*.)

- You can issue a **show** against a UFiT and the SMASH Proxy displays all properties, verbs, and contained targets. However, if you also want to see the associations for the UFiT, you must specify `–display all` or `–display associations`. For example,

  ```
  show –display associations system1
  ```

  displays the following output:

  ```
  Success
  UFiT: system1
  UFiP: /modular1/system1
  Associations:
  SystemComponent <-- /modular1
  OwningCollectionElement --> /modular1/system1/capabilities1
  OwningCollectionElement --> /modular1/system1/swinv1
  SystemDevice --> /modular1/system1/devicetray1
  SystemDevice --> /modular1/system1/presencesensor1
  SystemDevice --> /modular1/system1/ntempsensor1
  SystemDevice --> /modular1/system1/ntempsensor2
  SystemDevice --> /modular1/system1/ntempsensor3
  SystemDevice --> /modular1/system1/ntempsensor4
  SystemDevice --> /modular1/system1/nvoltsensor1
  SystemDevice --> /modular1/system1/nvoltsensor2
  SystemDevice --> /modular1/system1/nvoltsensor3
  SystemDevice --> /modular1/system1/nvoltsensor4
  SystemDevice --> /modular1/system1/nvoltsensor5
  SystemDevice --> /modular1/system1/nvoltsensor6
  SystemDevice --> /modular1/system1/nvoltsensor7
  SystemDevice --> /modular1/system1/oemiicmled1
  SystemDevice --> /modular1/system1/oemiicmled2
  SystemDevice --> /modular1/system1/oemiicmled3
  SystemDevice --> /modular1/system1/oemiicmled4
  SystemDevice --> /modular1/system1/oemiicmled5
  SystemDevice --> /modular1/system1/oemiicmled6
  SystemComponent --> /modular1/system1/sp1
  ComputerSystemPackage --> /hdwr1/chassis1/bladepkg1
  ConcreteDependency --> /modular1/system1/ntempsensor1
  ConcreteDependency --> /modular1/system1/ntempsensor2
  ConcreteDependency --> /modular1/system1/ntempsensor3
  ConcreteDependency --> /modular1/system1/ntempsensor4
  ConcreteDependency --> /modular1/system1/nvoltsensor1
  ConcreteDependency --> /modular1/system1/nvoltsensor2
  ConcreteDependency --> /modular1/system1/nvoltsensor3
  ConcreteDependency --> /modular1/system1/nvoltsensor4
  ConcreteDependency --> /modular1/system1/nvoltsensor5
  ConcreteDependency --> /modular1/system1/nvoltsensor6
  ConcreteDependency --> /modular1/system1/nvoltsensor7
  ElementSoftwareIdentity --> /modular1/system1/swinv1/swid1
  ElementSoftwareIdentity --> /modular1/system1/swinv1/swid2
  ElementSoftwareIdentity --> /modular1/system1/swinv1/swid3
  ```

For a list of verb options and their descriptions, see “SMASH CLP supported verb options and descriptions” on page 9. A specific discussion about the `display` option and its arguments can be found in “Display and Output verb options” on page 11.

Using the propertyvalues parameter

You can use the property values with the –display option through the **propertyname==propertyvalue** parameter to filter when the selected target properties are output.

For example, Table 43 shows the following properties for the UFcT target record.

Table 43. record UFcT properties

UFcT	Property	Property type	Possible values for property
record	*CreationClassName*	string	"IICM_LogRecord"
	RecordID	string	[Chassis Name]|[Instance Tag]
	LogCreationClassName	string	"IICM_RecordLog"
	LogName	string	"BCMMEventLog"
	MessageTimestamp	string	variable
	OEMIICM_Severity	string	"INFO", "WARN", "ERR"
	OEMIICM_Source	string	SERVPROC, BLADE_x
	OEMIICM_Name	string	variable
	RecordData	string	variable
	DataFormat	string	string severity string source string name string date string time string text
	Locale	string	"en_US"
	Caption	string	"Bladecenter Management Module Event Log Record"
	ElementName	string	"record"

You can then issue a **show** –display properties=(RecordID,OEMIICM_Severity=="ERR") record*, where *ERR* is the specific property value for OEMIICM_Severity for which you want results. The command would result in the following output:

```
Success
UFiT: log1
UFiT: record3
UFiP: /modular2/chassismgr1/logs1/log1/record3
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:12:18:16 Text:Can not read power-on VPD for blade 1.
UFiT: record23
UFiP: /modular2/chassismgr1/logs1/log1/record23
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929Date:04/18/06 Tim
e:11:36:18 Text:Power Supply 4 Fault
UFiT: record24
UFiP: /modular2/chassismgr1/logs1/log1/record24
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:11:36:18 Text:Power Supply 3 Fault
UFiT: record25
UFiP: /modular2/chassismgr1/logs1/log1/record25
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:11:36:18 Text:Power Supply 2 Fault
UFiT: record26
UFiP: /modular2/chassismgr1/logs1/log1/record26
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:11:36:18 Text:Power Supply 1 Fault
UFiT: record56
UFiP: /modular2/chassismgr1/logs1/log1/record56
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:10:51:20 Text:Power Supply 4 Fault
```

```
UFiT: record57
UFiP: /modular2/chassismgr1/logs1/log1/record57
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:10:51:20 Text:Power Supply 3 Fault
UFiT: record58
UFiP: /modular2/chassismgr1/logs1/log1/record58
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:10:51:20 Text:Power Supply 2 Fault
UFiT: record59
UFiP: /modular2/chassismgr1/logs1/log1/record59
Properties:
RecordData: Severity:ERR Source:SERVPROC Name:WMN315652929 Date:04/18/06 Tim
e:10:51:20 Text:Power Supply 1 Fault
```

In the following tables, when you see *variable* in the Possible values for property column it means that there is not a fixed set of CIM values that the property can assume, but rather, the values vary based on data from the specific hardware.

A property type of `uint16 ValueMap` indicates that the property has both an integer value and a value-mapped string value. The string values are listed in parentheses in the Possible values for property column. When you display `ValueMap` type properties through the **show** command, the SMASH Proxy displays both the integer and string value of the property. When you use a `ValueMap` type property in a **set** command or `display` filter, it is valid to specify either the integer or string value of the property.

List of properties

Table 44. admin UFcT properties

UFcT	Property	Property type	Possible values for property
admin	*CreationClassName*	string	"IICM_MAPAdminDomain"
	Name	string	"SMASH-CLP-IICM-Domain"
	Description	string	"Administrative domain for SMASH CLP"
	ElementName	string	"MAPProvider Admin Domain"

Table 45. bladepkg UFcT properties

UFcT	Property	Property type	Possible values for property
bladepkg (pkg)	*CreationClassName*	string	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name]|[Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:BladeServer"
	PackageType	uint16 ValueMap	16 ("Blade")
	PartNumber	string	variable
	SerialNumber	string	variable
	SKU	string	variable
	ManufactureDate	datetime	variable
	Version	string	variable
	PoweredOn	boolean	true, false
	RemovalConditions	uint16 ValueMap	4 ("Removable when on or off")
	Model	string	variable
	Manufacturer	string	variable
	OtherIdentifyingInfo	string	variable (UUID of blade)

Table 46. bladexpkg UFcT properties

UFcT	Property	Property type	Possible values for property
bladexpkg (pkg)	*CreationClassName*	string	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name] | [Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:BladeServer:BSE"
	PackageType	uint16 ValueMap	17 ("Blade Expansion")
	PoweredOn	boolean	true, false
	RemovalConditions	uint16 ValueMap	4 ("Removable when on or off")

Table 47. capabilities UFcT properties

UFcT	Property	Property type	Possible values for property
capabilities (concretecollection)	*InstanceID*	string	[Hostname | [Instance Tag]
	Description	string	"Collection of all Software Installation Capabilities for Management Module"
	ElementName	string	"Capabilities"

Table 48. capacities UFcT properties

UFcT	Property	Property type	Possible values for property
capacities (concretecollection)	*InstanceID*	string	[Hostname | [Instance Tag]
	Description	string	"Collection of all configuration capacity information for Modular System"
	ElementName	string	"Capacities"

Table 49. card UFcT properties

UFcT	Property	Property type	Possible values for property
card	*CreationClassName*	string	"IICM_Card"
	Tag	string	[Hostname | [Instance Tag]
	HostingBoard	boolean	false
	Manufacturer	string	variable
	SKU	string	variable
	SerialNumber	string	variable
	Version	string	variable
	PartNumber	string	variable
	ManufactureDate	datetime	variable
	PackageType	uint16 ValueMap	9 ("Module/Card")
	VendorCompatibilityStrings	string	"IBM:BC:BladeServer:DaughterCard"

Table 50. chassis UFcT properties

UFcT	Property	Property type	Possible values for property
chassis	*CreationClassName*	string	"IICM_Chassis"
	Tag	string	[Chassis Name] | [Instance Tag]
	PackageType	uint16 ValueMap	3 ("Chassis/Frame")
	MultipleSystemSupport	uint16 ValueMap	1 ("True")
	Model	string	variable
	Manufacturer	string	variable
	SerialNumber	string	variable
	SKU	string	variable
	PartNumber	string	variable
	Version	string	variable
	ManufactureDate	datetime	variable
	OtherIdentifyingInfo	string[]	variable (UUID)

Table 51. chassismgr UFcT properties

UFcT	Property	Property type	Possible values for property
chassismgr (system)	*CreationClassName*	string	"IICM_HWCtrlPoint"
	Name	string	[Chassis Name]|[Instance Tag]
	EnabledState	uint16 ValueMap	2 ("Enabled"), 0 ("Unknown"), 1 ("Other")
	OtherEnabledState	string	"Failover" **Note:** The **show** command displays the OtherEnabledState property only if the value of the EnabledState property is "Other".
	RequestedState	**uint16 ValueMap**	**2 ("Enabled"), 5 ("No Change")**
	Dedicated	uint16[] ValueMap	29 ("Chassis Manager")
	OEMIICM_ChassisUUID	string	variable
	NameFormat	string	"Other"
	OperationalStatus	uint16[] ValueMap	11 ("In Service"), 15 ("Dormant"), 13 ("Lost Communication")
	IdentifyingDescriptions	string[]	"SLP Name Attribute", "Physical Slot", "Serial Number"
	OtherIdentifyingInfo	string[]	variable
	Systemtime	datetime	variable

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 52. configcapacity UFcT properties

UFcT	Property	Property type	Possible values for property
configcapacity	*Name*	string	[Chassis Name]|[Instance Tag]
	ObjectType	uint16 ValueMap	5 ("I/O Slots"), 0 ("Other"), 3 ("Fans"), 2 ("Power Supplies")
	OtherTypeDescription	string	"Management Modules", "Blade Servers" **Note:** The **show** command will display the OtherTypeDescription property only if the value of the ObjectType property is "Other".
	MinimumCapacity	uint64	0, 1
	MaximumCapacity	uint64	2, 4, 10, 14

Table 53. devicetray UFcT properties

UFcT	Property	Property type	Possible values for property
devicetray (logicalmodule)	*SystemCreationClassName*	string	"IICM_ModularSystem" OR "IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_LogicalModule"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	ModuleNumber	string	1
	LogicalModuleType	uint16 ValueMap	2 ("Device Tray")
	Availability	uint16 ValueMap	3 ("Running/Full Power")
	EnabledState	uint16 ValueMap	2 ("Enabled")
	RequestedState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")
	CurrentAccess	**uint16 ValueMap**	**2 ("No Access"), 3("Exclusive Access")**

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 54. fanpkg UFcT properties

UfcT	Property	Property type	Possible values for property
fanpkg (pkg)	*CreationClassName*	string	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name]|[Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:Cooling:Fan"
	PackageType	uint16 ValueMap	7 ("Fan")
	RemovalConditions	uint16 ValueMap	4 ("Removable when on or off")

Table 55. gateway UFcT properties

UFcT	Property	Property type	Possbile values for property
gateway (remotesap)	*CreationClassName*	string	"IICM_RemoteServiceAccessPoint"
	Name	string	[Chassis Name]|[Instance Tag]
	SystemCreationClassName	string	"IICM_HWCtrlPoint"
	SystemName	string	[Chassis Name]|[Instance Tag]
	AccessInfo	**string**	**variable**
	InfoFormat	uint16 ValueMap	3 ("IPv4 Address")
	AccessContext	string	"Default Gateway"

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 56. hdwr UFcT properties

UFcT	Property	Property type	Possible values for property
hdwr (concretecollection)	*InstanceId*	string	"hdwr1"
	Description	string	"Collection of all chassis in Admin domain"
	ElementName	string	"Hardware"

Table 57. ipendpt UFcT properties

UFcT	Property	Property type	Possible values for property
ipendpt	*SystemCreationClassName*	string	"IICM_HWCtrlPoint" OR "IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_IPProtocolEndPoint"
	Name	string	[Chassis Name]|[Instance Tag]
	IPv4Address	**string**	**variable**
	SubnetMask	**string**	**variable**

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 58. log UFcT properties

UFcT	Property	Property type	Possible values for property
log	*InstanceID*	string	[Chassis Name]|[Instance Tag]
	Name	string	"BCMMEventLog"
	MaxNumberofRecords	uint16 ValueMap	0 **Note:** 0 here means "Undefined" because there is a fixed maximum space that log records can consume but not a fixed maximum number of records.
	EnabledState	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")
	EnabledDefault	string	2 ("Enabled")
	Caption	string	"BladeCenter Management Module Event Log"
	ElementName	string	"log"

Table 59. logs UFcT properties

UFcT	Property	Property type	Possible values for property
logs (concretecollection)	*InstanceID*	string	[Chassis Name]|[Instance Tag]
	Description	string	"Collection of all Logs on Management Module"
	ElementName	string	"Logs"

Table 60. modular UFcT properties

UFcT	Property	Property type	Possible values for property
modular (system)	*CreationClassName*	string	"IICM_ModularSystem"
	Name	string	[Chassis Name]|[Instance Tag]
	NameFormat	string	"Other"
	HealthState	uint16 ValueMap	25 ("Critical failure"), 15 ("Minor failure"), 5 ("OK")
	OperationalStatus	uint16[] ValueMap	11 ("In Service"), 15 ("Dormant"), 13 ("Lost Communication")
	Dedicated	uint16[] ValueMap	2 ("Other")
	OtherDedicatedDescriptions	string[]	"Modular"

Table 61. modulepkg UFcT properties

UFcT	Property	Property type	Possible values for property
modulepkg (pkg)	*CreationClassName*	string	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name]|[Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:CMM:MM", "IBM:BC:CMM:AMM" **Note:** IBM:BC:CMM:MM is Management Module 1, IBM:BC:CMM:AMM is Advanced Management Module.
	PackageType	uint16 ValueMap	9 ("Module/Card")
	PartNumber	string	variable
	SKU	string	variable
	RemovalConditions	uint16 ValueMap	4 ("Removable when on or off")
	ManufactureDate	datetime	variable
	Manufacturer	string	variable
	Version	string	variable

Table 62. ntachsensor UFcT properties

UFcT	Property	Property type	Possible values for property
ntachsensor (sensor, numsensor)	*SystemCreationClassName*	string	"IICM_ModularSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_NumericSensor"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	BaseUnits	uint16 ValueMap	65 ("Percentage")
	UnitModifier	sint32	0
	RateUnits	uint16 ValueMap	0 ("None")
	CurrentReading	sint32	variable
	SensorType	uint16 ValueMap	5 ("Tachometer")
	ElementName	string	[Chassis Name]|[Instance Tag]
	NormalMax	sint32	100
	NormalMin	sint32	0
	MaxReadable	sint32	100
	MinReadable	sint32	0
	Resolution	sint32	1
	EnabledState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	6 ("Error"), 2 ("OK")
	Caption	string	"Fan Speed - % of Maximum"

Table 63. ntempsensor UFcT properties (for modular temperature sensors)

UFcT	Property	Property type	Possible values for property
ntempsensor (numsensor)	*SystemCreationClassName*	string	"IICM_ModularSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_NumericSensor"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	BaseUnits	uint16 ValueMap	2 ("Degrees C")
	UnitModifier	sint32	-2
	RateUnits	uint16 ValueMap	0 ("None")
	CurrentReading	sint32	variable
	SensorType	uint16 ValueMap	2 ("Temperature")
	ElementName	string	[Chassis Name]|[Instance Tag]
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	15 ("Dormant"), 6 ("Error"), 2 ("OK")
	Caption	string	"Front Panel Temperature", "Management Module Temperature"

Table 64. ntempsensor UFcT properties (for system temperature sensors)

UFcT	Property	Property type	Possible values for property
ntempsensor (numsensor)	*SystemCreationClassName*	string	"IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_NumericSensor"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	BaseUnits	uint16 ValueMap	2 ("Degrees C")
	UnitModifier	sint32	-2
	RateUnits	uint16 ValueMap	0 ("None")
	CurrentReading	sint32	variable
	SensorType	uint16 ValueMap	2 ("Temperature")
	PossibleStates	string[]	{"Normal", "Upper Critical", "Upper Fatal", "Disabled"}
	CurrentState	string	{"Normal", "Upper Critical", "Upper Fatal", "Disabled"}
	NormalMax	sint32	variable
	UpperThresholdCritical	sint32	variable
	UpperThresholdFatal	sint32	variable
	SupportedThresholds	sint32	{3 ("UpperThresholdCritical"), 5 ("UpperThresholdFatal")}
	EnabledThresholds	sint32	{3 ("UpperThresholdCritical"), 5 ("UpperThresholdFatal")}
	ElementName	string	[Chassis Name]|[Instance Tag]
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	10 ("Stopped"), 6 ("Error"), 2 ("OK")
	Caption	string	"Blade CPU x Temperature" where x is the CPU number for the blade.

Table 65. nvoltsensor UFcT properties

UFcT	Property	Property type	Possible values for property
nvoltsensor (numsensor)	*SystemCreationClassName*	string	"IICM_ModularSystem" OR "IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_NumericSensor"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	BaseUnits	uint16 ValueMap	5 ("Volts")
	UnitModifier	sint32	-2
	RateUnits	uint16 ValueMap	0 ("None")
	CurrentReading	sint32	variable
	SensorType	uint16 ValueMap	3 ("Voltage")
	PossibleStates	string[]	{"Lower Critical", "Upper Critical", "Normal"}
	CurrentState	string	{"Lower Critical", "Upper Critical", "Normal"}
	ElementName	string	"+5V", "+3.3V", "+12V", "-5V", "+2.5V", "1.8V"
			"+5 Volts", "+3.3 Volts", "+12 Volts", "+2.5 Volts", "1.5 Volts", "+1.25 Volts", "VRM1" (non-IPMI blades)
			variable
	NormalMax	sint32	variable (for chassis only)
	NormalMin	sint32	variable (for chassis only)
	LowerThresholdCritical	sint32	variable
	UpperThresholdCritical	sint32	variable
	SupportedThresholds	uint16[] ValueMap	{2 ("LowerThresholdCritical"), 3 ("UpperThresholdCritical")}
	EnabledThresholds	uint16[] ValueMap	{2 ("LowerThresholdCritical"), 3 ("UpperThresholdCritical")}
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled") (for chassis only)
	OperationalStatus	uint16[] ValueMap	15 ("Stopped"), 6 ("Error"), 2 ("OK")
			10 ("Stopped"), 6 ("Error"), 2 ("OK") (for blades)
	Caption	string	"Planar xxx Voltage" where xxx is the ElementName (Planar)
			"Blade xxx Voltage" where xxx is the ElementName (blades)

Table 66. oemiicmled UFcT properties

UFcT	Property	Property type	Possible values for property
oemiicmled	*SystemCreationClassName*	string	"IICM_ModularSystem" OR "IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_Indicator"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	Caption	string	various
	ElementName	string	various
	IndicatorDeviceType	uint16 ValueMap	2 ("LED")
	IndicatorType	uint16 ValueMap	3 ("Visible")
	IndicatorCommand	uint16 ValueMap	0 ("Off"), 1 ("On"), 2 ("Blinking"), 3 ("Not Available")
	ControlMode	uint16 ValueMap	2 ("Automatic"), 3 ("Manual")
	EnabledState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")

Table 67. oemiicmled UFcT properties (for Enterprise Chassis LEDs)

UFcT	Property	Property type	Possible values for property
oemiicmled [Enterprise Chassis LEDs]	*SystemCreationClassName*	string	"IICM_ModularSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_Indicator"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	Caption	string	["Front Panel Identity LED", "Front Panel Temperature LED", "Front Panel Information LED", "Front Panel System Error LED"]
	ElementName	string	["Front Panel Identity LED", "Front Panel Temperature LED", "Front Panel Information LED", "Front Panel System Error LED"]
	IndicatorDeviceType	uint16 ValueMap	2 ("LED")
	IndicatorType	uint16 ValueMap	3 ("Visible")
	IndicatorCommand	uint16 ValueMap	0 ("off"), 1 ("on"), 2 ("blinking"), 3 ("not available")
	ControlMode	uint16 ValueMap	2 ("Automatic"), 3 ("Manual")
	EnabledState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")

Table 68. oemiicmled UFcT properties (for Telco Chassis LEDs)

UFcT	Property	Property type	Possible values for property
oemiicmled [Telco Chassis LEDs]	*SystemCreationClassName*	string	"IICM_ModularSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_Indicator"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	Caption	string	["Front Panel Critical LED", "Front Panel Major LED", "Front Panel Minor LED", "Front Panel Identity LED"]
	ElementName	string	["Front Panel Critical LED", "Front Panel Major LED", "Front Panel Minor LED", "Front Panel Identity LED"]
	IndicatorDeviceType	uint16 ValueMap	2 ("LED")
	IndicatorType	uint16 ValueMap	3 ("Visible")
	IndicatorCommand	uint16 ValueMap	0 ("Off"), 1 ("On"), 2 ("Blinking"), 3 ("Not Available")
	ControlMode	uint16 ValueMap	2 ("Automatic"), 3 ("Manual")
	EnabledState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")

Table 69. oemiicmled UFcT properties (for Blade System LEDs)

UFcT	Property	Property type	Possible values for property
oemiicmled [Blade System LEDs]	*SystemCreationClassName*	string	"IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_Indicator"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	Caption	string	["Blade x System Error LED", "Blade x Information LED", "Blade x Identity LED", "Blade x Power LED", "Blade x Media Tray LED", Blade x KVM LED"]
	ElementName	string	["Blade x System Error LED", "Blade x Information LED", "Blade x Identity LED", "Blade x Power LED", "Blade x Media Tray LED", Blade x KVM LED"]
	IndicatorDeviceType	uint16 ValueMap	2 ("LED")
	IndicatorType	uint16 ValueMap	3 ("Visible")
	IndicatorCommand	uint16 ValueMap	0 ("Off"), 1 ("On"), 2 ("Blinking"), 3 ("Not Available")
	ControlMode	uint16 ValueMap	2 ("Automatic"), 3 ("Manual")
	EnabledState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")

Table 70. oemiicmled UFcT properties (for CPM Switch LEDs)

UFcT	Property	Property type	Possible values for property
oemiicmled [CPM Switch LEDs]	*SystemCreationClassName*	string	"IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_Indicator"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	Caption	string	["Switch x External Port y LED", "Switch x Internal Port y"]
	ElementName	string	["Switch x External Port y LED", "Switch x Internal Port y"]
	IndicatorDeviceType	uint16 ValueMap	2 ("LED")
	IndicatorType	uint16 ValueMap	3 ("Visible")
	IndicatorCommand	uint16 ValueMap	0 ("Off"), 1 ("On")
	ControlMode	uint16 ValueMap	2 ("Automatic")
	EnabledState	uint16 ValueMap	2 ("Enabled")
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")

Table 71. oemiicmswitchmgt UFcT properties

UFcT	Property	Property type	Possible values for property
oemiicmswitchmgt	*SystemCreationClassName*	string	"IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_SwitchManagementService"
	Name	string	[Chassis Name]|[Instance Tag]
	Started	boolean	true
	OEMIICM_ExternalPortsEnabled	**boolean**	**true, false**

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 72. pkg UFcT properties

UFcT	Property	Property type	Possible values for property
pkg	*CreationClassName*	string	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name]|[Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:IOModule"
	PackageType	uint16 ValueMap	1 ("Other")
	OtherPackageType	string	"Switch"
	SKU	string	variable
	Version	string	variable
	PartNumber	string	variable
	ManufactureDate	datetime	variable
	Manufacturer	string	variable
	PoweredOn	boolean	true, false
	RemovalConditions	uint16 ValueMap	4 ("Removable when on or off")
	OtherIdentifyingInfo	string	variable

Table 73. presencesensor UFcT properties

UFcT	Property	Property type	Possible values for property
presencesensor (sensor)	*SystemCreationClassName*	string	"IICM_ModularSystem" OR "IICM_ComputerSystem"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_Sensor"
	DeviceID	string	[Chassis Name]|[Instance Tag]
	SensorType	uint16 ValueMap	11 ("Presence")
	PossibleStates	string[]	{"Absent", "Present"}
	CurrentState	string	"Absent", "Present"
	ElementName	string	[Chassis Name]|[Instance Tag]
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled") (Blade Expansion Unit presence sensor)
			2 ("Enabled")(all other component sensors)
	EnabledDefault	uint16 ValueMap	2 ("Enabled")
	OperationalStatus	uint16[] ValueMap	2 ("OK")
	Caption	string	"Presence Sensor for [component] in Bay x where component is Fan, Blade, Blade Server Expansion Unit, Management Module, Switch, Power Module, or Presence Sensor for Media Tray"

Table 74. pwrpkg UFcT properties

UFcT	Property	Property type	Possible values for property
pwrpkg (pkg)	*CreationClassName*	string	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name]|[Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:PowerModule"
	PackageType	uint16 ValueMap	6 ("Power Supply")
	PartNumber	string	variable
	SKU	string	variable
	ManufactureDate	datetime	variable
	Version	string	variable
	RemovalConditions	uint1 ValueMap	4 ("Removable when on or off")
	Manufacturer	string	variable
	OtherIdentifyingInfo	string	variable

Table 75. record UFcT properties

UFcT	Property	Property type	Possible values for property
record	*CreationClassName*	string	"IICM_LogRecord"
	RecordID	string	[Chassis Name]|[Instance Tag]
	LogCreationClassName	string	"IICM_RecordLog"
	LogName	string	"BCMMEventLog"
	MessageTimestamp	string	variable
	OEMIICM_Severity	string	"INFO", "WARN", "ERR"
	OEMIICM_Source	string	SERVPROC, BLADE_x
	OEMIICM_Name	string	variable
	RecordData	string	variable
	DataFormat	string	string severity string source string name string date string time string text
	Locale	string	"en_US"
	Caption	string	"Bladecenter Management Module Event Log Record"
	ElementName	string	"record"

Table 76. SESSION UFcT properties

UFcT	Property	Property type	Possible values for property
SESSION	**CurrentDefaultTarget**	**string**	**variable**
	KeepTime	**uint16**	**variable (default is 60 seconds)**
	WaitBehavior	**boolean**	**true, false**
	OutputFormat	**uint16 ValueMap**	**2 (“text”), 5 (“clpxml”)**
	OutputVerbosity	**uint16 ValueMap**	**2 (“error”), 3 (“terse”), 4 (“Verbose”)**
	OutputLanguage	uint16 ValueMap	“eng”
	OutputPosition	**uint16 ValueMap**	**2 (“begin”), 3 (“end”)**
	OutputOrder	**uint16 ValueMap**	**2 (“default”), 3 (“reverse”)**
	OutputCount	**uint32**	**variable (default is -1 (all))**

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 77. shareddevicesvc UFcT properties

UFcT	Property	Property type	Possible values for property
shareddevicesvc	*SystemCreationClassName*	string	“IICM_HWCtrlPoint”
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	“IICM_SharedDeviceManagement Service”
	Name	string	[Chassis Name]|[Instance Tag]
	Started	string	true
	EnabledState	uint16 ValueMap	2 (“Enabled”)
	RequestedState	uint16 ValueMap	2 (“Enabled”)
	EnabledDefault	uint16 ValueMap	2 (“Enabled”)
	OperationalStatus	uint16[] ValueMap	2 (“OK”)

Table 78. sharingcap UFcT properties

UFcT	Property	Property type	Possible values for property
sharingcap	*InstanceID*	string	[Chassis Name]|[Instance Tag]
	SupportedAccessModes	uint16[] ValueMap	2 (“No Access”), 3 (“Exclusive Access”)

Table 79. sp UFcT properties

UFcT	Property	Property type	Possible values for property
sp (system)	*CreationClassName*	string	“IICM_ComputerSystem”
	Name	string	[Chassis Name]|[Instance Tag]
	Dedicated	uint16[] ValueMap	28 (“Management Controller”)
	StatusDescriptions	string[]	“Powered On”, “Powered Off”
	NameFormat	string	″Other″
	OperationalStatus	uint16[] ValueMap	1 (″Other″)

Table 80. storagepkg UFcT properties

UFcT	Property	Property type	Possible values for property
storagepkg (pkg)	*CreationClassName*	string[]	"IICM_PhysicalPackage"
	Tag	string	[Chassis Name]|[Instance Tag]
	VendorCompatibilityStrings	string[]	"IBM:BC:MediaTray"
	PackageType	uint16 ValueMap	15 ("Storage Media Package (i.e., Disk or Tape Drive)")
	SKU	string	variable
	Version	string	variable
	PoweredOn	boolean	true
	Manufacturer	string	variable

Table 81. swid UFcT properties

UFcT	Property	Property type	Possible values for property
swid	*CreationClassName*	string	"IICM_SoftwareIdentity"
	InstanceID	string	[Chassis Name]|[Instance Tag]
	RevisionNumber	uint16	variable
	VersionString	string	variable
	ReleaseDate	datetime	variable
	OEMIICM_FileName	string	variable
	Name	string	"Main App", "Boot ROM", "Remote Control", "PS2toUSB Controller", "USBIntf" (for Management Modules)
			"BIOS", "Diagnostics", "Blade System Management Processor" (for Blades)
			"Main App 1", "Main App 2", "Boot ROM" (for Switches)

Table 82. swinv UFcT properties

UFcT	Property	Property type	Possible values for property
swinv	*InstanceID*	string	[Chassis Name]|[Instance Tag]

Table 83. swinstallsvc UFcT properties

UFcT	Property	Property type	Possible values for property
swinstallsvc	*SystemCreationClassName*	string	"IICM_HWCtrlPoint"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_SoftwareInstallationService"
	Name	string	[Chassis Name]|[Instance Tag]
	EnabledState	uint16 ValueMap	2 (″Enabled″)
	RequestedState	uint16 ValueMap	2 (″Enabled″)

Table 84. swinstallsvccap UFcT properties

UFcT	Property	Property type	Possible values for property
swinstallsvccap	*InstanceID*	string	[Chassis Name]|[Instance Tag]
	ElementName	string	"Software Installation Service Capabilities"
	SupportedSynchronousActions	uint16[] ValueMap	2 ("None supported")
	SupportedAsynchronousActions	uint16[] ValueMap	5 ("Install from URI")
	SupportedTargetTypes	string[]	"Management Module Firmware", "Blade System Management Processor Firmware", "Pass-through Module Firmware"

Table 85. switch UFcT properties

UFcT	Property	Property type	Possible values for property
switch (system)	*CreationClassName*	string	"IICM_ComputerSystem"
	Name	string	[Chassis Name]|[Instance Tag]
	NameFormat	string	"Other"
	HealthState	uint16 ValueMap	0 ("Unknown"), 5 ("OK"), 10 ("Degraded/Warning"), 20 ("Major Failure")
	StatusDescriptons	string[]	"Powered On", "Powered Off"
	OperationalStatus	uint16[] ValueMap	1 ("Other")
	Dedicated	uint16[] ValueMap	5 ("Switch")
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled")
	RequestedState	**uint16 ValueMap**	**2 ("Enabled"), 3 ("Disabled")**

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 86. system UFcT properties

UFcT	Property	Property type	Possible values for property
system	*CreationClassName*	string	"IICM_ComputerSystem"
	Name	string	[Chassis Name]|[Instance Tag]
	NameFormat	string	"Other"
	HealthState	uint16 ValueMap	0 ("Unknown"), 5 ("OK"), 10 ("Degraded/Warning"), 20 ("Major Failure")
	StatusDescriptons	string[]	"Powered On", "Powered Off"
	OperationalStatus	uint16[] ValueMap	1 ("Other")
	Dedicated	uint16[] ValueMap	0 ("Not Dedicated")
	EnabledState	uint16 ValueMap	2 ("Enabled"), 3 ("Disabled")
	RequestedState	**uint16 ValueMap**	**2 ("Enabled"), 3 ("Disabled")**

Note: Properties marked in bold font denote properties that you can modify through the **set** command.

Table 87. textredirectsap UFcT properties

UFcT	Property	Property type	Possible values for property
textredirectsap	*SystemCreationClassName*	string	"IICM_HWCtrlPoint"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_TextRedirectionSAP"
	Name	string	[Chassis Name]|[Instance Tag]

Table 88. textredirectsvc UFcT properties

UFcT	Property	Property type	Possible values for property
textredirectsvc	*SystemCreationClassName*	string	"IICM_HWCtrlPoint"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_TextRedirectionService"
	Name	string	[Chassis Name]|[Instance Tag]

Table 89. timesvc UFcT properties

UFcT	Property	Property type	Possible values for property
timesvc	*SystemCreationClassName*	string	"IICM_HWCtrlPoint"
	SystemName	string	[Chassis Name]|[Instance Tag]
	CreationClassName	string	"IICM_TimeService"
	Name	string	[Chassis Name]|[Instance Tag]
	Started	boolean	true

Some specific UFcTs in the tables have their general UFcT specified (in parentheses) where the specific UFcT would be displayed if the general UFcT is the target of a **show**. For example, if a user performed a **show** `pkg*`, where *pkg* is the general ufct of bladepkg, than not only would there be information on bladepkg displayed, but also the specific UFcTs storagepkg, pwrpkg, fanpkg, and so on.

Association properties

Association instances can be displayed using the following syntax:

```
[UFiP]=>[Association Name]
```

The following table contains a list of SMASH Proxy supported association properties. Properties marked in **bold** font denote properties that can be modified using the **set** command.

Note: A **show** [association instance] command only displays properties in the following table that are not in *italics*.

Table 90. Association properties

Association	Property	Property type	Possible values for property
ComputerSystemPackage	*Antecedent*	REF	
	Dependent	REF	
	PlatformGUID	string	
ConcreteDependency	*Antecedent*	REF	
	Dependent	REF	
Container	*GroupComponent*	REF	
	PartComponent	REF	
ElementCapabilities	*ManagedElement*	REF	
	Capabilities	REF	
ElementSoftwareIdentity	*Antecedent*	REF	
	Dependent	REF	
HostedAccessPoint	*Antecedent*	REF	
	Dependent	REF	
HostedService	*Antecedent*	REF	
	Dependent	REF	
LogManagesRecord	*Log*	REF	
	Record	REF	
MemberOfCollection	*Collection*	REF	
	Member	REF	
OwningCollectionElement	*OwningElement*	REF	
	OwnedElement	REF	

Table 90. Association properties (continued)

Association	Property	Property type	Possible values for property
PackageInChassis	*GroupComponent*	REF	
	PartComponent	REF	
	LocationWithinContainer	string	
Realizes	*Antecedent*	REF	
	Dependent	REF	
RemoteAccessAvailableToElement	*Antecedent*	REF	
	Dependent	REF	
ServiceAccessBySAP	*Antecedent*	REF	
	Dependent	REF	
ServiceAffectsElement	*AffectedElement*	REF	
	AffectingElement	REF	
	ElementEffects	uint16[]	
ServiceAvailableToElement	*ServiceProvided*	REF	
	UserOfService	REF	
SharingDependency	*Antecedent*	REF	
	Dependent	REF	
	CurrentAccess	**uint16**	
SystemComponent	*GroupComponent*	REF	
	PartComponent	REF	
SystemDevice	*GroupComponent*	REF	
	PartComponent	REF	
SystemPackaging	*Antecedent*	REF	
	Dependent	REF	
UseOfLog	*Antecedent*	REF	
	Dependent	REF	

SMASH Proxy command target property descriptions

The tables below contain descriptions for all SMASH Proxy command target properties as described by CIM.

Command target properties (A-C)

Properties (A-C)	Description
AccessContext	Type of service provided by a remote service access point.
AccessInfo	Addressing information for a remote connection.
Availability	The primary availability and status of a device.

Properties (A-C)	Description
BaseUnits	The base unit of the values returned by a sensor. All the values returned by a sensor are represented in the units obtained by `BaseUnits` * 10 raised to the power of the `UnitModifier`. For example, if `BaseUnits` is Volts and the `UnitModifier` is -6, then the units of the values returned are MicroVolts. However, if the `RateUnits` property is set to a value other than **None**, then the units are further qualified as rate units. In the previous example, if `RateUnits` is set to **Per Second**, then the values returned by the sensor are in MicroVolts Per Second. The units apply to all numeric properties of a sensor.
Caption	A short textual description (one-line string) of an object.
ControlMode	Indicates the mode in which management of the Indicator is operating.
CreationClassName	Indicates the name of the class or the subclass used in the creation of an instance. When used with the other key properties of this class, this property allows all instances of a class and its subclasses to be uniquely identified.
CurrentDefaultTarget	The `CurrentDefaultTarget` is the CLP session environment setting that establishes a default base address for all command targets that are expressed as a relative target address and is used as the command target if no command target is specified in a command.
CurrentReading	The current value indicated by a sensor.
CurrentState	The current state indicated by a sensor. This is always one of the PossibleStates.

Command target properties (D-F)

Properties (D-F)	Description
DataFormat	A free-form string describing a LogRecord's data structure.

Properties (D-F)	Description
Dedicated	Enumeration indicating whether the ComputerSystem is a special-purpose system (dedicated to a particular use) or a general-purpose system (is **not** dedicated). A value of **Management Controller** indicates this instance represents specialized hardware dedicated to systems management (for example, a Baseboard Management Controller [BMC] or service processor). The management scope of a **Management Controller** is typically a single managed system in which it is contained. A value of **Chassis Manager** indicates this instance represents a system dedicated to management of a blade chassis and its contained devices.
Description	A textual description of the object.
DeviceID	An address or other identifying information used to uniquely name a LogicalDevice.
EnabledDefault	An enumerated value indicating an administrator's default or startup configuration for the Enabled State of an element.
ElementName	A user-friendly name for an object.
EnabledState	An integer enumeration that indicates the enabled and disabled states of an element.
EnabledThresholds	An array representing the thresholds that are currently enabled for a sensor.

Command target properties (G-I)

Properties (G-I)	Description
HealthState	Indicates the current health of an element.
HostingBoard	Boolean indicating that, if true, the card is a motherboard or a baseboard in a chassis. If false, the card is a daughter card.
IdentifyingDescriptions	An array of free-form strings providing explanations and details behind the entries in the `OtherIdentifyingInfo` array. Note, each entry of this array is related to the entry in `OtherIdentifyingInfo` that is located at the same index.
IndicatorCommand	Defines the indicator instructions that are currently being issued to the Indicator device.
IndicatorDeviceType	The physical type of the Indicator device.
IndicatorType	Specifies sensory form of the presentation to a human.
InstanceID	Within the scope of the instantiating namespace, `InstanceID` opaquely and uniquely identifies an instance of a class.

Properties (G-I)	Description
IPv4Address	The IPv4 address that a ProtocolEndpoint represents.

Command target properties (J-L)

Properties (J-L)	Description
KeepTime	Establishes the amount of time in seconds that a CLP Session should retain a CLP command job status. A `KeepTime` value of zero shall be interpreted to mean **keep forever**.
Locale	A locale indicates a particular geographical, political, or cultural region. The `Locale` specifies the language used in creating the RecordForLog data.
LogicalModuleType	Identifies the type of LogicalModule that an instance represents: • **Device Tray** indicates that the device is a device or media tray, for example, in a modular system.
LogCreationClassName	A scoping Log's CreationClassName.
LogName	A scoping Log's Name.
LowerThresholdCritical	A Sensor's threshold values specify the ranges (minimum and maximum values) for determining whether a sensor is operating under Normal, NonCritical, Critical or Fatal conditions.

Command target properties (M-O)

Properties (M-O)	Description
ManufactureDate	The date that a PhysicalElement was manufactured.
Manufacturer	The name of the organization responsible for producing a PhysicalElement.
MaximumCapacity	Maximum number of elements of type, ObjectType, that can be installed.
MaxNumberofRecords	Maximum number of records that can be captured in the Log. If undefined, a value of zero should be specified.
MaxReadable	Indicates the largest value of the measured property that can be read by the NumericSensor.
MessageTimestamp	Timestamp for a LogRecord.
MinimumCapacity	Minimum number of elements of type, ObjectType, that must be installed.
MinReadable	Indicates the smallest value of the measured property that can be read by the NumericSensor.

Properties (M-O)	Description
Model	The name by which the PhysicalElement is generally known.
ModuleNumber	Physical or logical slot that a logical module occupies.
MultipleSystemSupport	Indicates whether or not the chassis supports multiple systems, for example, server blades.
Name	Defines the label by which an object is known.
NameFormat	Format of the Name field.
NormalMax	Provides guidance for the user as to the normal maximum range for the NumericSensor.
NormalMin	NormalMin provides guidance for the user as to the normal minimum range for the NumericSensor.
ObjectType	The type of object (power supply, fan, and so on) whose capacities are indicated.
OEMIICM_ChassisUUID	Universal Unique Identifier (UUID) of a chassis.
OEMIICM_ExternalPortsEnabled	Boolean status of a switch's external ports. If value is true, the external ports are enabled; otherwise, they are disabled.
OEMIICM_FileName	Firmware file name.
OEMIICM_Name	Name of the system generating the log entry.
OEMIICM_Severity	Severity of the log entry generated.
OEMIICM_Source	Source of the log entry generated (for example, SERVPROC).
OperationalStatus	Indicates the current statuses of an element.
OtherDedicatedDescriptions	A string describing how or why a system is dedicated when the `Dedicated` array includes the value 2 (**Other**).
OtherEnabledState	A string that describes the enabled or disabled state of an element when the `EnabledState` property is set to 1 (**Other**).
OtherIdentifyingInfo	Captures data in addition to `Tag` information.
OtherPackageType	A string describing the package when the instance's PackageType property is 1 (**Other**).
OtherTypeDescription	A string describing the object type when the ObjectType property is 0 (**Other**).
OutputCount	Establishes the default CLP session command output count number when multiple items are returned in the output.

Properties (M-O)	Description
OutputFormat	Establishes the default CLP session command output format (text or clpxml). **Note:** The clpxml format conforms to the SM CLP Command Response XML Schema (www.dmtf.org/apps/org/workgroup/svrmgmt/download.php/17388/dsp0224.xsd).
OutputLanguage	Establishes the default CLP session command output language. For SMASH, the only supported language is en_US.
OutputOrder	Establishes the default CLP session command output order when multiple items are returned in the command output (default, reverse).
OutputPosition	Establishes the default CLP session command output beginning or ending position when multiple items are returned in the command output (begin, end).
OutputVerbosity	Establishes the default CLP session output level of detail (terse, error, verbose).

Command target properties (P-R)

Properties (P-R)	Description
PackageType	Enumeration defining the type of the PhysicalPackage.
PartNumber	The part number assigned by the organization that is responsible for producing or manufacturing a PhysicalElement.
PossibleStates	Enumerates the string outputs of the sensor.
PoweredOn	Boolean that indicates whether a PhysicalElement is powered on (TRUE) or is currently off (FALSE).
RateUnits	Specifies if the units returned by this sensor are rate units. All the values returned by this sensor are represented in the units obtained by BaseUnits * 10 raised to the power of the UnitModifier. This is true unless this property (`RateUnits`) has a value different than **None**. For example, if `BaseUnits` is Volts and the UnitModifier is -6, then the units of the values returned are MicroVolts. But, if the `RateUnits` property is set to a value other than **None**, then the units are further qualified as rate units. In the above example, if `RateUnits` is set to **Per Second**, then the values returned by the sensor are in MicroVolts Per Second. The units apply to all numeric properties of the sensor.
RecordData	A string containing LogRecord data.

Properties (P-R)	Description
RecordID	`RecordID`, with the `MessageTimestamp` property, serves to uniquely identify the LogRecord within a MessageLog.
ReleaseDate	The date the software was released.
RemovalConditions	Addresses whether a PhysicalPackage is removable with or without power being applied.
RequestedState	An integer enumeration that indicates the last requested or required state for an element.
Resolution	Indicates the ability of a sensor to resolve differences in the measured property. The units for this measurement are determined by BaseUnit*UnitModifier/RateUnit.
RevisionNumber	The revision or maintenance release component of a software's version information - for example, **3** from version 12.1(3)T. This property is defined as a numeric value to allow the determination of **newer** versus. **older** releases. A **newer** revision is indicated by a larger numeric value.

Command target properties (S-U)

Properties (S-U)	Description
SensorType	The type of the sensor, for example, Voltage or Temperature sensor. A description of the different sensor types is as follows: • **Temperature** sensor measures the environmental temperature. • **Voltage** sensor measures electrical voltage and current readings. • **Tachometer** measures speed and revolutions of a device. For example, a fan device can have an associated **Tachometer** that measures its speed. • **Presence** sensor detects the presence of a PhysicalElement.
SerialNumber	A manufacturer-allocated number used to identify a PhysicalElement.
SKU	The stock-keeping unit number for a PhysicalElement.
Started	A Boolean that indicates whether the service has been started (TRUE), or stopped (FALSE).
StatusDescriptions	Strings describing the various `OperationalStatus` array values. Note that entries in this array are correlated with those at the same array index in `OperationalStatus`.

Properties (S-U)	Description
SubnetMask	The mask for the IPv4 address of this ProtocolEndpoint, if one is defined.
SupportedAccessModes	Describes the types of shared access that are supported for a referenced Logicaldevice.
SupportedAsynchronousActions	Enumeration indicating what operations are run as asynchronous jobs.
SupportedSynchronousActions	Enumeration indicating what operations are run without the creation of a job.
SupportedTargetTypes	An array containing a list of SoftwareIdentity `TargetType` properties that a service **knows** how to install.
SupportedThresholds	An array representing the thresholds supported by a sensor.
SystemCreationClassName	The `CreationClassName` of the scoping system.
SystemName	The `Name` of the scoping system.
SystemTime	Pseudo-property representing the current date and time on a computer system.
Tag	An arbitrary string that uniquely identifies the PhysicalElement and serves as the key of the element.
UnitModifier	The unit multiplier for the values returned by a sensor. All the values returned by this sensor are represented in the units obtained by BaseUnits * 10 raised to the power of the `UnitModifier`. For example, if `BaseUnits` is Volts and the `UnitModifier` is -6, then the units of the values returned are MicroVolts. However, if the `RateUnits` property is set to a value other than **None**, then the units are further qualified as rate units. In the above example, if `RateUnits` is set to **Per Second**, then the values returned by the sensor are in MicroVolts Per Second. The units apply to all numeric properties of the sensor.
UpperThresholdCritical	The sensor's threshold values specify the ranges (minimum and maximum values) for determining whether the sensor is operating under **Normal**, **NonCritical**, **Critical** or **Fatal** conditions.
UpperThresholdFatal	The sensor's threshold values specify the ranges (min and max values) for determining whether the sensor is operating under **Normal**, **NonCritical**, **Critical** or **Fatal** conditions.

Command target properties (V-Z)

V	Description
VendorCompatibilityStrings	An array of strings that identify the component that is compatible with, and can be inserted in a slot that reports this string as one of the array element in the VendorCompatibilityStrings. This allows system administrators to determine whether it is appropriate to insert a package into a slot.
Version	A string that indicates the version of a PhysicalElement.
VersionString	A string representing the complete software version information.
WaitBehavior	When TRUE, the CLP session does not return a command response until all spawned jobs are complete.

For a full description of the meaning of various properties and their values, please see the *CIM Schema: Version 2.10.1*. You can access this document from Chapter 7, "SMASH-related documentation," on page 109.

CIM property types

The following table contains a list of SMASH Proxy supported CIM property types.

Table 91. CIM property types

Intrinsic data type	Interpretation
uint16	Unsigned 16-bit integer
uint32	Unsigned 32-bit integer
sint32	Signed 32-bit integer
uint64	Unsigned 64-bit integer
string	UCS-2 string
boolean	Boolean
datetime	A string containing a date-time
[classname] ref	Strongly typed reference

Date, time, and interval types

datetime uses the fixed string-based format:

```
yyyymmddhhmmss.mmmmmmsutc
```

where:

- `yyyy` is a 4-digit year
- `mm` is the month
- `dd` is the day
- `hh` is the hour (24-hour clock)
- `mm` is the minute
- `ss` is the second
- `mmmmmm` is the number of microseconds

- s is a plus sign (+) or a dash (-), indicating the sign of the Universal Coordinated Time (UTC), which for all intents and purposes, is the same as Greenwich Mean Time correction field. The correction field is in minutes.

For example, you would represent Monday, May 25, 1998, at 1:30:15 PM EST as:

```
19980525133015.0000000-300
```

Managing multiple chassis

The SMASH Proxy allows management of all BladeCenter chassis in an environment from a single management station. All BladeCenter units are discovered through SLP using the oemiicmdiscover command (for details, see "OEM verbs" on page 49). After they have been discovered, the BladeCenter chassis, and their components, show up as CLP manageable objects and can be targeted by CLP commands either physically (/hdwr1/chassis1, /hdwr1/chassis2.../hdwr1/chassisn) or logically (/modular1, /modular2.../modular*n*).

Handling chassis credentials

All chassis management functions require the CLP user to enter valid credentials for the managed chassis. The SMASH Proxy manages credentials by using a Linux stack paradigm such that only one set of credentials is active at a time. When a user first establishes a CLP session, the user gets a login prompt that requests a user name and password. The user name and password become the active set of credentials. If a user wishes to manage a system with a different set of credentials, he or she must use the **oemiicmlogin** command. When a user runs the **oemiicmlogin** command, the SMASH Proxy pushes the new set of credentials on the stack and they become the active set. When a user runs the **oemiicmlogoff** command, the SMASH Proxy discards the current set of credentials and the previous set becomes the active set.

See the following procedure as an example:

1. A user is managing an environment with five chassis. Two of these chassis (chassis1 and chassis2) have their MM user name and password set to user1/password1. The other three chassis (chassis3, chassis4, and chassis5) have their MM username and/password set to user2/password2.
2. The user does an SSH to the SMASH Proxy system and gets a login prompt. The user logs in as clpuser/clppw:

   ```
   oemiicmlogin -userid user1 -password password1
   ```

   The password stack now looks as follows:

   ```
   clpuser/clppw    <=  top  (active credentials)
   ```
3. The user runs the following commands:

   show /hdwr1/chassis1 (due to invalid credentials)

   show /hdwr1/chassis2 (due to invalid credentials)

   show /hdwr1/chassis3 (due to invalid credentials)
4. The user enters **oemiicmlogin** to log in as user1/password1. The password stack now looks as follows:

   ```
   user1/password1   <= top (active credentials)
   clpuser/clppw1
   ```
5. The user runs the following commands:

   show /hdwr1/chassis1 (succeeds)

show /hdwr1/chassis2 (succeeds)

show /hdwr1/chassis3 (due to invalid credentials)

6. To manage chassis3, the user must enter a new set of active credentials as follows:

   ```
   oemiicmlogin –userid user2 –password password2
   ```

7. The password stack now looks as follows:

   ```
   user2/password2   <= top (active credentials)
   user1/password1
   clpuser/clppw
   ```

8. The user runs the following commands:

   show /hdwr1/chassis3 (succeeds)

   show /hdwr1/chassis4 (succeeds)

   show /hdwr1/chassis5 (succeeds)

   show /hdwr1/chassis1 (due to invalid credentials)

9. Although user1/password1 for chassis1 is on the stack, it is not the active set. To manage chassis1 again, the user must run the **oemiicmlogoff** command. After running this, the password stack looks as follows:

   ```
   user1/password1    <=  top  (active credentials)
   clpuser/clppw
   ```

At this point, the user can manage chassis1 and chassis2, but cannot manage the other chassis.

Because this is a proxy, a user having authorization to access the CLP system does not automatically mean that he is authorized to manage a chassis. A user can also be authorized to manage chassis 1 but not chassis 2.

Important:

1. Although the above examples use the **show** /hdwr/chassis**x** command to illustrate their point, credentials correspond to operations on all objects associated with a chassis (for example, **show** /modular1/system5).
2. The smash_snmp.cfg file in /cfg has a variable named context, which defines the default SNMPv3 context for all user IDs. If the SNMPv3 context for the user IDs that you are logging on with does not match the default SNMPv3 context, user ID can be specified as userid:context at both the login prompt and with the **oemiicmlogin** command. For example:

   ```
   Username: USERID:mycontext
   OR
   oemiicmlogin -userid USERID:mycontext -password PASSW0RD
   ```

   You can specify the SNMPv3 context using the BladeCenter MM Web interface. For details, see “SNMPv3 configuration in the MM” on page 40.

For details on the **oemiicmlogin** and **oemiicmlogoff** commands, see “OEM verbs” on page 49.

Administering text console redirection

Text console redirection allows you to start an SOL connection to the command console of a specific blade server. Typical SMASH Proxy commands follow a command-response interface. A user enters a command at a prompt. The CLP processes the command and writes a response to the screen. Text console redirection, by the nature of the function, does not return a response to the screen. For example:

1. At the command prompt, enter **start** `/modular1/chassismgr1/textredirectsap1`. This causes the SOL session for blade 1 in chassis 1 to take over the CLP session. All commands you enter after the **start** command are commands to the blade server console and not to the CLP.
2. To end the SOL session and return to the CLP session, enter **Esc (**.

You can manage multiple SOL sessions for different blades through separate CLP sessions. Up to 14 text-console redirection sections can be active concurrently. The UFiT index corresponds to the blade slot of a redirected blade server console. In other words, textredirectsap1 redirects the console for the blade in slot 1, textredirectsap2 redirects the console for the blade in slot 2, and so on.

All blades support SOL except the 8678.

Note: To conduct a text console redirection, the SMASH Proxy uses SSH to access the MM. However, SSH is disabled by default on the MM. Thus, for text console redirection to take place, you must enable it in the Web interface and restart the MM. See Figure 15 on page 106 for an image of this MM Web interface panel.

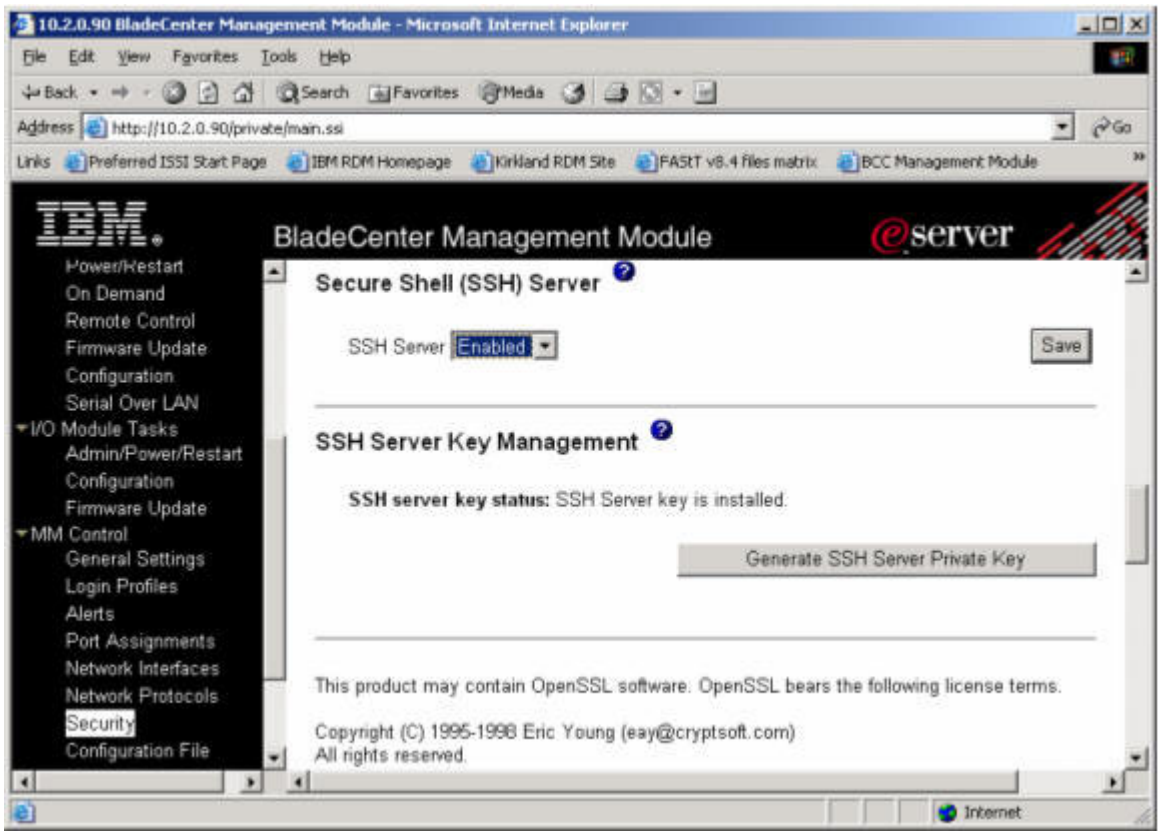

Figure 15. Enable SSH panel

Reviewing job status

In the SMASH Proxy, jobs are local to a session. You can only see the status of jobs that you spawned. When you exit the session, all history of your jobs disappear.

You run the **load** verb, without the `wait` option, asynchronously. If you specify the `wait` option, the **load** verb runs synchronously.

Note: By default, at session invocation, the WaitBehavior SESSION property is set to *true*, which means all loads will be synchronous. To make loads asynchronous, set the `WaitBehavior` SESSION property to *false*.

The SMASH Proxy handles all verbs listed in "SMASH CLP supported verbs and descriptions" on page 8 synchronously with the exception of the **load** verb. The **load** verb can be made synchronous with the `-wait` option.

It must be noted that even though it is a SMASH CLP requirement for the **exit** verb to support the `-keep` option, which establishes a holding time for the job ID and status associated with a verb, the `-keep` function does not operate in the same manner with **exit** as it does with other verbs. Instead, when you issue the **exit** `-keep` command in the shell, jobs go away as soon as a shell is exited.

Using pass-thru modules

A *pass-thru module* is a device that replaces physical cables and allows internal network physical ports in a chassis to be made accessible from the external network. In the BladeCenter chassis, the pass-thru modules are hardware components that plug into the switch bays. The BladeCenter unit supports two types of pass-thru modules:

- IBM BladeCenter Optical Pass-thru Module
- IBM BladeCenter Copper Pass-thru Module

In the SMASH Proxy, pass-thru modules are modeled as switches and show up with a UFcT of *switch*.

Implementing redundant MMs

The SMASH Proxy implements a limited redundant MM model. In this model, chassismgr1 handles the active MM. The SMASH Proxy handles failover conditions (transparent to the user) to maintain chassismgr1 as the active chassis. Therefore, a user should always target chassismgr1 for all management operations through the MM.

The only chassismgr operations relevant for chassismgr2 are:

- Viewing the status
- Making it the redundant chassis
- Viewing the software inventory

Turning on debug for the SMASH Proxy

By default, debugging is disabled for the SMASH Proxy when it is installed. If you experience a problem running the SMASH Proxy, IBM support might direct you to enable debugging. Debugging is available for various components of the Proxy.

To enable SNMP debugging, set the *snmp_debug* variable in [*install_dir*]/cfg/smash_snmp.cfg] to 1.

```
# SMASH SNMP Configuration File
# This file allows configuration of the SNMPv3 default parameters to use in
# communications between the SMASH proxy and managed Bladecenters.

# snmp_timeout in microseconds, default is 3.0 sec
# This is the time the SMASH Proxy will wait for a response from the managed
# Bladecenter before retrying an SNMP request.
snmp_timeout = 3000000

# session_timeout, in seconds, default is 5 min (300 sec)
# This is the amount of idle time the SMASH Proxy will allow before closing an
# SNMP session between a SMASH Proxy user and a managed Bladecenter. This only
# closes out the underlying SNMP session not the SMASH CLP user session.
# If the user/Bladecenter SNMP session has been closed due to idleness, it will
# be automatically reopened at the next user query.
session_timeout = 300

# snmp_retries, default is 3
# This is the number of times the SMASH Proxy will retry an SNMP request to a
# managed Bladecenter before sending an error back to the user.
snmp_retries = 3

# context
# This parameter is the default context to be used in Bladecenter SNMPv3
# communications.
# When the user logins to the SMASH Proxy or issues an oemiicmlogin command
# he can specify a username:context. If the user omits the context from the
```

```
# login, the default context specified in this file will be used for the SNMPv3
# request. The context in the SNMPv3 setup web interface for the Bladecenter
# must match the value submitted in the login or the value in this file (if the
# context was not specified at login). If context is blank in any of the web
# interfaces for the Bladecenters to be managed, then context should be
# commented out here.
#context = admin

# authentication_protocol
# This value should match the authentication protocol specified in the SNMPv3
# setup web interface for all managed Bladecenters. All managed Bladecenters
# must be configured with the same authentication protocol.
# Allowable values are MD5, SHA, and None
authentication_protocol = MD5


# privacy_protocol
# This value should match the privacy protocol specified in the SNMPv3 setup
# web interface for all managed Bladecenters. All managed Bladecenters must be
# configured with the same privacy protocol.
# Allowable values are DES and None
privacy_protocol = DES

# snmp_logfile
# This is the file where all debug messages will be logged.
snmp_logfile = /var/log/iicm/smash_snmp.log

# snmp_debug
# This controls logging of net-snmp library debug messages to snmp_logfile
# A value of 0 means debug is turned off.
# Any other value (e.g. 1) turns on snmp debug messages.
snmp_debug = 0

# num_sessions
# Number of simultaneous user sessions, default is 14
num_sessions = 14
```

To enable MM provider debugging, set the *smash_trace_level* variable in [*install_dir*]/cfg/smash_mm.cfg to a level of 5.

```
# SMASH MM Provider Configuration File

# smash_log_file is the main provider log file
smash_log_file = /var/log/iicm/smash_mm.log

# smash_snmp_cfg_file contains SNMP configuration data, including
# an snmp_logfile path.
smash_snmp_cfg_file = /etc/iicm/smash_snmp.cfg

# cim_restart_command is the command used to restart the cimserver
cim_restart_command = /etc/iicm/cimrestart.sh

# smash_slp_data_file: Discovery puts SLP output here
smash_slp_data_file = /var/log/iicm/smash_slpdata.dat
# smash_slp_data_config: Discovery puts the resulting CIM data here
smash_slp_data_config = /var/log/iicm/smash_cfgdata.dat

# smash_trace_level values:
# 0 = none, except load messages
# 1 = errors only (RASLOG_ERR)
# 2 = errors & warnings (RASLOG_WARN)
# 3 = include debug messages (RASLOG_DBG)
# 4 = include extra debug messages (low level => RASLOG_DBGLO)
# 5 = include function enter/exit messages (RASLOG_FN)
smash_trace_level = 0
```

To enable Credentials server debugging, set log-mode in iicm_cred.conf to:

```
debug::/var/log/iicm/iicm_credserv.log
```

and set debug to true.

Chapter 7. SMASH-related documentation

See the following links for currently available SMASH-related documentation:

CIM Schema
Version 2.10.1
Downloadable from:
http://www.dmtf.org/standards/cim/cim_schema_v2101/
[DMTF Confidential]

Systems Management Architecture for Server Hardware (SMASH) Architecture White Paper (SM CLP Architecture WP 1.0a)
DMTF Server Management Working Group
Downloadable from the documents section at:
http://www.dmtf.org/apps/org/workgroup/svrmgmt/
[DMTF Confidential]

Server Management Managed Element Addressing Specification (SM ME Addressing Specification)
DMTF Server Management Working Group
Downloadable from the documents section at:
http://www.dmtf.org/apps/org/workgroup/svrmgmt/
[DMTF Confidential]

Server Management Command Line Protocol Specification (SM-CLP)
DMTF Server Management Working Group
Downloadable from the documents section at:
http://www.dmtf.org/apps/org/workgroup/svrmgmt/
[DMTF Confidential]

SM CLP-CIM Mapping Specification, DMTF Server Management Working Group
Downloadable from the documents section at:
http://www.dmtf.org/apps/org/workgroup/svrmgmt/
[DMTF Confidential]

BladeCenter Serial over LAN Setup Guide
Downloadable from:
ftp://ftp.software.ibm.com/pc/pccbbs/pc_servers_pdf/31r1734.pdf

Management Module User's Guide - IBM BladeCenter, BladeCenter T
Downloadable from:
ftp://ftp.software.ibm.com/pc/pccbbs/pc_servers_pdf/24r9706.pdf

Troubleshooting Serial over LAN issues - IBM BladeCenter
Downloadable from:
http://www-307.ibm.com/pc/support/site.wss/document.do?lndocid=MIGR-59728

SM CLP Command Response XML Schema
Downloadable from:
http://www.dmtf.org/apps/org/workgroup/svrmgmt/download.php/17388/dsp0224.xsd

Get Adobe® Reader® to view the PDF files listed above.

Chapter 8. Printable PDF

Get Adobe® Reader® to view this PDF file.

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

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
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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.

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Table 92. Trademark list

Trademarks
IBM (the IBM logo)
IBM BladeCenter
eServer

Intel is a trademark of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Glossary

Terms

This glossary defines technical terms and abbreviations used in this SMASH Proxy document. If you do not find the term you are looking for, view the IBM Terminology Web site at: http://www.ibm.com/ibm/terminology.

Selection of Terms: A term is a word or group of words to be defined. In this glossary, the singular form of the noun and the infinitive form of the verb are the terms most often selected to be defined. If the term may be abbreviated, the abbreviation is indicated. The abbreviation is also defined in its proper place in the glossary.

A

absolute target address
A designation of target address which begins at the root of the containment hierarchy.

admin domain
A logical entity that is the access point for managing a chassis. Also referred to as **administrative domain**.

association
Class that contains two or more references. It represents a relationship between two or more objects.

C

CDT See **current default target**.

CIM See **common information model**.

client A logical component that manages a system via the system's manageability access point (MAP). A client may run on a management station or other system.

CLP See **command line protocol**.

common information model
A standard which provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions.

command line
Expression of a management action, including a command verb, and if specified, a command target, options and option arguments, and properties and value.

command line protocol
The command line protocol defined by SMASH, used for managing systems.

command processor
The logical entity within a MAP responsible for parsing, interpreting, and executing incoming commands and returning responses.

command response
Response returned by the CLP service to a client when a command is issued. This consists of **command status** and **command results**.

command results
The actual results of a successful command returned as part of the command response.

command status
Information returned by the CLP service to a client describing the overall status of a command.

current default target
The CLP session environment setting that establishes a default base address for all command targets that are expressed as a relative target address and is used as the command target if no command target is specified in a command entered.

E

Ethernet
A packet-based networking technology for local area networks (LANs) that allows multiple access and handles contention by using Carrier Sense Multiple Access with Collision Detection (CSMA/CD) as the access method. Ethernet is standardized in the IEEE 802.3 specification.

G

GMT See **Greenwich mean time**.

Greenwich mean time
The mean solar time at the meridian of Greenwich, England.

I

implicit command target
The target acted upon is inherent to the command being executed.

Internet Protocol (IP) address
The unique 32-bit address that specifies the location of each device or workstation in the Internet. For example, 9.67.97.103 is an IP address.

IP address
See **Internet Protocol (IP) address**.

K

KVM Keyboard, Video, Mouse.

L

LED See **Light Emitting Diode**.

Light Emitting Diode
A semiconductor chip that gives off visible or infrared light when activated.

M

manageability access point
A service of a system that provides management in accordance to specifications of SMASH.

managed element
The finest granularity of addressing which can be the target of commands or messages, or a collection thereof.

managed system
A collection of managed elements that comprise a computer system.

Management Module
The management module provides system-management functions and KVM multiplexing for all of the blade servers in the BladeCenter unit that support KVM. It controls the external keyboard, mouse, and video connections, for use by a local console, and a 10/100 Mbps Ethernet remote management connection.

MAP See **manageability access point**.

ME See **managed element**.

MM See **Management Module**.

multicast
Transmission of the same data to a selected group of destinations.

N

Non-addressing association
An association instance which is not used by the MAP in constructing the UFiP to any instances the association instance references.

O

OEM See **Original Equipment Manufacturer**.

OEM properties
Properties added to instances of a class by an OEM vendor.

OEM target
A managed element whose properties, behavior, UFcT, etc. are outside the scope of the SMASH specifications and are vendor dependent.

OEM verbs
Verbs defined by an OEM vendor which are outside the SM CLP specification.

option A term of the command line that selects a particular behavior of a command verb.

Original Equipment Manufacturer
Company incorporating SMASH standard into its own product.

P

property
An attribute of the command target.

R

relative target address
A designation of target address in relation to the current default target as opposed to an absolute target address.

S

Secure Shell Version 2 (SSHv2)
A protocol which permits secure remote access over a network from one computer to another.

SSHv2
See **Secure Shell Version 2**.

SMASH CLP
See **Systems Management Architecture for Server Hardware Command Line Protocol**.

SMASH Proxy
IBM software product which allows remote management of BladeCenter chassis in conformance with the SMASH standard.

subnet
A network divided into smaller independent subgroups, which still are interconnected.

Systems Management Architecture for Server Hardware Command Line Protocol
A command line protocol developed by the DMTF Systems Management Workgroup for managing server hardware.

T

target The specific managed element or association that is to be affected by the command verb.

target address
A string value used in a command line to identify the target for a command.

U

UFcT See **User-Friendly class Tag**.

UFiP See **User-Friendly instance Path**.

UFiT See **User-Friendly instance Tag**.

UFsT See **User-Friendly selection Tag**.

unicast
Transmission of data to a single destination.

User-Friendly class Tag
A short human friendly alias for a CIM class name. It has the same properties and methods as the CIM class it represents.

User-Friendly instance Path
The unique path to an instance formed by concatenating the UFiTs of each instance from the root instance to the terminating instance.

User-Friendly instance Tag
User friendly identifier for a specific instance of a CIM class. A UFiT is constructed by concatenating an integer suffix to the UFcT for the CIM class.

User-Friendly selection Tag
Short-hand notation for selecting all instances of a given class. A UFsT is constructed by concatenating the UFcT for a class with the character *.

V

verb The string name of a command, used as the first term of a command line.

Index

V

W

Part Number: 40K1521

Printed in USA

(1P) P/N: 40K1521