BMC Atrium CMDB 7.6.00

Data Modeling Guide

September 2009

ACTIVATE BUSINESS WITH THE POWER OF I.T.™

www.bmc.com

Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address	BMC SOFTWARE INC 2101 CITYWEST BLVD HOUSTON TX 77042-2827 USA	Telephone	713 918 8800 or 800 841 2031	Fax	713 918 8000

Outside United States and Canada

Telephone	(01) 713 918 8800	Fax	(01) 713 918 8000

If you have comments or suggestions about this documentation, contact Information Development by email at doc_feedback@bmc.com.

ITIL® is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office, and is used here by BMC Software, Inc., under license from and with the permission of OGC.

Restricted rights legend

Customer Support

You can obtain technical support by using the Support page on the BMC Software website or by contacting Customer Support by telephone or email. To expedite your inquiry, please see "Before Contacting BMC Software."

Support Website

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at `http://www.bmc.com/support`. From this website, you can:

- Read overviews about support services and programs that BMC Software offers.
- Find the most current information about BMC Software products.
- Search a database for problems similar to yours and possible solutions.
- Order or download product documentation.
- Report a problem or ask a question.
- Subscribe to receive email notices when new product versions are released.
- Find worldwide BMC Software support center locations and contact information, including email addresses, fax numbers, and telephone numbers.

Support by telephone or email

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813 or send an email message to `customer_support@bmc.com`. (In the Subject line, enter `SupID:<`*`yourSupportContractID`*`>`, such as `SupID:12345`.) Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC Software

Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - Product name
  - Product version (release number)
  - License number and password (trial or permanent)
- Operating system and environment information
  - Machine type
  - Operating system type, version, and service pack
  - System hardware configuration
  - Serial numbers
  - Related software (database, application, and communication) including type, version, and service pack or maintenance level
- Sequence of events leading to the problem
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - Product error messages
  - Messages from the operating system, such as `file system full`
  - Messages from related software

License key and password information

If you have a question about your license key or password, contact Customer Support through one of the following methods:

- E-mail `customer_support@bmc.com`. (In the Subject line, enter `SupID:<*yourSupportContractID*>`, such as `SupID:12345`.)
- In the United States and Canada, call 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
- Submit a new issue at `http://www.bmc.com/support`.

Contents

Preface

The *BMC Atrium CMDB 7.6.00 Data Modeling Guide* describes how to model business entities in BMC Atrium CMDB 7.6.00. The guide uses the Common Data Model (CDM) and extensions to that model, and explores recommended practices for using new entities effectively.

The BMC Atrium Configuration Management Database (CMDB) enables you to store and manage information about products and services that are in your environment. The BMC Atrium CMDB uses the term *class* to describe a configuration item (CI) or relationship classification. Each CI is partially classified using some common *attributes* that describe the *base class* (`BMC_BaseElement`). Specific details about each class of CI are described by attributes of *subclasses* of `BMC_BaseElement`. Relationships are also modeled as a base relationship class (`BMC_BaseRelationship`) with subclasses for different types of relationships.

As a provider of BMC Atrium CMDB data, BMC Atrium Discovery products can discover large amounts of configuration data for use by data consumers. BMC Atrium Discovery products are natural enablers for the creation of service models because they can discover many of the components, or CIs, that ultimately make up the service models. These components include:

- Computer systems (including servers, routers, physical and virtual systems, and operating systems)
- Applications
- Software servers (including specialized elements such as SAP®, Sun™, Siebel, and mainframe infrastructure components)
- Databases
- Business process definitions
- Network elements

Audience

This guide is intended for configuration managers, application administrators, asset analysts, and related IT professionals.

Conventions used in this guide

This guide illustrates how to use the classes that BMC provides for the BMC Atrium CMDB to model a particular business entity, focusing on how you use the entire model rather than on general information about a class or attribute. Although descriptions of classes and attributes are provided to give you context when determining how to model CIs, detailed information such as syntax and the type of attribute is not specified. For that level of information, see the BMC Atrium CMDB 7.6.00 Data Model Help.

This guide applies the following conventions to explain BMC Atrium CMDB concepts in both textual and graphical formats.

Terminology

In many cases you will be modeling an entity using classes from the CDM, but you might also model part of that same entity using an extension to the CDM. For models that require extensions to the CDM, the term *data model* is used. This guide is organized so that the entities are introduced first in each section, including the recommended practice for that implementation. Any classes and attributes that can be included in the deployment of these business entities in an IT infrastructure are described in an architectural diagram. Where appropriate, recommendations are provided for setting specific attributes for a given class.

Attributes are defined as either *key* or *additional*. Key attributes are those that BMC recommends that you populate for a given class to model a specific CI. Additional attributes are optional attributes that you can populate to further classify a CI or relationship.

Differentiating Name and ShortDescription attributes

A common misconception is that the caption for the CI on user interfaces and reports is represented by the `Name` attribute, when it is actually the `ShortDescription` attribute. In diagrams in this guide, the names that appear are not from the `Name` attribute, they are the `ShortDescription` attribute (which is usually just a user interface caption). Also, in modeling recommendations, `ShortDescription` is the more user-friendly label, and should always be provided and set with a value that makes sense to an end user.

Diagrams

Illustrative model diagrams help explain the concepts and modeling recommendations in this guide, and also show how you might model an entity in a real-world business environment. In these diagrams, CIs are represented by single-line boxes that contain attributes of the class or its parent class. Where applicable, key attributes are shown in the box that depicts a specific class and, in some cases, include the recommended value of those attributes.

NOTE

Illustrative diagrams are just examples, and might not reflect every possible class, attribute, or relationship that you would use for modeling all types of the represented object.

Relationships represented in illustrative model diagrams

In the diagrams, boxes illustrate how CIs in your environment should be mapped to the CDM, or how to extend the CDM to create your own data model. Lines are used to represent the type and direction of the relationship.

NOTE

Relationships in the diagrams in this guide are illustrated using Unified Modeling Language (UML) standards. The UML notation may not be consistent with the BMC Atrium CMDB 7.6.00 user interfaces (UI). Some of this discrepancy is due to the absence of a direct UML equivalent to the relationships represented, and some of it is the lack of alignment between the CDM, the UI, and UML standards.

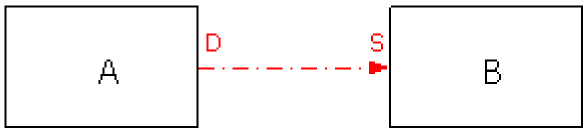
Although discrepancies may exist between the UML standards and the BMC Atrium CMDB UI, changes in the UI for future releases of BMC Atrium CMDB will enable the UI to more closely align with UML. In this guide, the conventions applied to the diagrams enable you to easily distinguish which relationship is used in a modeling scenario, regardless of how you might view them in the product.

For example, one major difference between UML standards and the BMC Atrium CMDB UI is that, in the UI, an arrow is always used to represent the source and destination of the relationship, whereas in UML, it is not. Therefore, in this guide, the diagrams more closely align with UML so that you can understand the semantic of the modeling scenario in the context of the corresponding best-practice modeling recommendations.

Although UML does not standardize colors in its rendering of relationships, they are used in the diagrams to help you easily distinguish at a glance which relationship type is recommended to model an example business object. Additionally, the source and destination of each relationship are represented by the letters S and D, respectively. The following section illustrates examples of each relationship type.

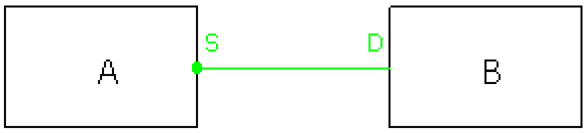
Examples of dependency relationships (arrow)

Dependency relationships are represented by dashed red lines, and contain an arrow to show the direction of the relationship. In a `BMC_Dependency` relationship, the arrow starts at A, the dependent (Destination), and ends at B, the antecedent (Source) of the relationship. Entity A is dependent on Entity B.



Example of a collection relationship (circle)

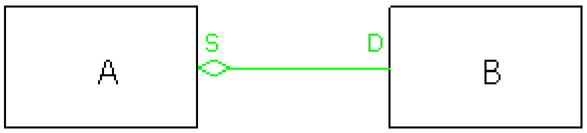
A `BMC_MemberOfCollection` relationship is represented by green lines with circle tips, as illustrated in the following diagram:



A is the collection class (Source), and B is the member class (Destination). The circle represents a collection relationship, where the collection class uses properties of the member class.

Example of a component relationship (diamond)

In a component relationship, the source CI is a group that has a component or part; its destination. Entity A is a group (Source) that has a component B (Destination). In diagrams, component relationships are represented by green lines with diamond tips.



Cardinality in relationships

Every relationship class has a cardinality that defines how many instances of the source class can be related to each instance of the destination class and vice versa. Where cardinality is specified in the diagrams, it is shown at the ends of the relationship lines as one of the following types:

- 1:1 (one to one)
- 1:* (one to many)
- *:1 (many to one)
- *:* (many to many)

Weak relationships

Where a *weak relationship* exists between two instances, that relationship is indicated by the letter W in the illustrative model diagrams. If the relationship is a weak relationship, its destination member, called the *weak* member, cannot exist without its source member, called the *strong* member. A weak relationship creates a logical composite object consisting of both member CIs.

BMC Atrium Core documentation

The following table lists the documentation available for BMC Atrium Core. Unless otherwise noted, documentation is available on the BMC Atrium Core documentation media (DVD or Electronic Product Download bundle) and on the BMC Customer Support site at `http://www.bmc.com/support`.

You can view PDF documents using Acrobat Reader, which is available through Adobe at `www.adobe.com`.

Title	Document provides	Audience	Format
BMC Atrium CMDB 7.6.00 Administrator's Guide	Information about setting permissions, configuring federation, modifying the data model, configuring an impact model, and other administrative tasks in BMC Atrium CMDB.	Administrators	Print and PDF
BMC Atrium CMDB 7.6.00 Common Data Model Diagram	Hierarchical diagram of all classes in the Common Data Model (CDM) including unique attributes and applicable relationships.	Administrators	Print and PDF
BMC Atrium CMDB 7.6.00 Data Model Help	Description and details of superclasses, subclasses, attributes, and relationship classes for each class. Contains only information about the Common Data Model at first, but you can update it to include information about data model extensions that you install.	Administrators	HTML (product media only)

Title	Document provides	Audience	Format
BMC Atrium CMDB 7.6.00 Data Modeling Guide	Best practices for using the classes that BMC provides for BMC Atrium CMDB (both the CDM and extensions) to model complex business entities, focusing on the use of multiple related CIs to model an entity rather than on general information about a class or attribute.	Administrators	Print and PDF
BMC Atrium CMDB 7.6.00 Normalization and Reconciliation Guide	Information about normalizing data in BMC Atrium CMDB and reconciling CIs from different data providers into a single production dataset.	Administrators	Print and PDF
BMC Atrium CMDB 7.6.00 Online Help	Help for using and configuring BMC Atrium CMDB, available by clicking Help in the product interface.	Users and administrators	Product Help (available from Help links after installation)
BMC Atrium CMDB 7.6.00 User's Guide	Information about using BMC Atrium CMDB, including searching for and comparing CIs and relationships, relating CIs, viewing history, running impact simulations, and viewing federated data.	Users	Print and PDF
BMC Atrium Core 7.6.00 Compatibility Matrix	Information about the BMC Atrium Core configurations that are expected to work properly based on design, testing, or general understanding of the interaction between products. Download the *BMC Atrium Core 7.6.00 Compatibility Matrix* from the BMC Customer Support site at `http://www.bmc.com/support/reg/remedy-compatibility-tables.html?c=n`.	Administrators	PDF
BMC Atrium Core 7.6.00 Concepts and Planning Guide	Information about CMDB concepts and high-level steps for planning and implementing your BMC Atrium Core implementation.	IT leaders and administrators	Print and PDF
BMC Atrium Core 7.6.00 Developer's Reference Guide	Information about creating API programs using C API functions and data structures.	Administrators and programmers	Print and PDF
BMC Atrium Core 7.6.00 Installation Guide	Information about installing, upgrading, and uninstalling BMC Atrium Core features.	Administrators	Print and PDF
BMC Atrium Core 7.6.00 Javadoc™ Help	Information about Sun™ Java™ classes, methods, and variables that integrate with BMC Atrium CMDB.	Programmers	HTML (product media only)
BMC Atrium Core 7.6.00 Master Index	Combined index of all guides.	Everyone	Print and PDF
BMC Atrium Core 7.6.00 Product Catalog and DML Guide	Information about configuring the Product Catalog and DML, updating vendor data, and creating aliases for products, manufacturers, and categorizations.	Administrators	Print and PDF

Title	Document provides	Audience	Format
BMC Atrium Core 7.6.00 Product Catalog and DML Online Help	Help for using and configuring BMC Atrium Product Catalog, available by clicking Help in the product interface.	Users and administrators	Product Help (available from Help links after installation)
BMC Atrium Core 7.6.00 Release Notes	Information about new features, known issues, and other late-breaking topics.	Everyone	Print and PDF
BMC Atrium Core 7.6.00: Taking Your Data Into Production End to End	End-to-end high-level steps for bringing data into BMC Atrium CMDB from a third-party source and making it available in your production dataset.	Administrators	Flash video (product media only)
BMC Atrium Core 7.6.00 Troubleshooting Guide	Information about resolving issues with BMC Atrium Core components, including API, filter and console error messages and their solutions.	Administrators, programmers, and BMC Support personnel	Print and PDF
BMC Atrium Core 7.6.00 Web Services Help	Information about using BMC Atrium Core Web Services, including how to publish and find interfaces in the Web Services Registry, set versions, disambiguate web services, configure security policies and encryption, and use BMC Atrium Core Web Services data structures and operations.	Administrators and programmers	HTML (product media only)
BMC Atrium Integration Engine 7.6.00 ADK Developer's Guide	Information about how to build adapters that can transfer information between an external data store and either BMC Remedy AR System forms or BMC Atrium CMDB.	Developers	Print and PDF
BMC Atrium Integration Engine 7.6.00 Online Help	Help for using and configuring BMC Atrium Integration Engine, available by clicking Help in the product interface.	Users and Administrators	Product Help (available from Help links after installation)
BMC Atrium Integration Engine 7.6.00 User's Guide	Information about how to create data exchanges, data mappings, define rules and queries, activate event-driven data exchanges, define connection settings, and conceptual information about BMC Atrium Integration Engine.	Users and Administrators	Print and PDF
Mapping Your Data to BMC Atrium CMDB 7.6.00 Classes	Mappings of common IT objects to the appropriate class in which to store them, whether part of the Common Data Model or an extension. Also includes information about further categorizing instances using key attributes and best practices for creating normalized relationships.	Administrators	Spreadsheet, print and PDF

Chapter

1 Modeling computer systems

This chapter describes how to use the CDM to model computer systems (servers, workstations, and network nodes such as routers, switches, and hubs). It details the classes, relationships, and attributes used to model computer systems, operating systems, hardware components, software inventory and patches, access points, and network interfaces.

For information on modeling applications, including modeling runtime versus installed aspects of applications, see "Modeling applications" on page 45.

The following topics are provided:

Logical identity of BMC_ComputerSystem

BMC_ComputerSystem is a class that stores CIs relating to collections of managed system elements. This is the primary class that you use to model the computers in your organization. You can use the attributes in this class to identify the purpose of each computer CI in your organization.

For example, the class contains several attributes that represent any network-addressable system, such as a server, a workstation, or a network device (router, switch, hub, load balancer, firewall, and so forth), as well as mainframes, printers, and virtual systems. Table 1-1 shows the key attributes that identify an instance of BMC_ComputerSystem.

Table 1-1: Key attributes for BMC_ComputerSystem

Attribute	Usage
Name, NameFormat	Identifies a computer system. The Name attribute should be a unique instance identifier that may not be Human Readable. Because multiple valid naming conventions may exist and can be used according to specific contexts, set the NameFormat attribute with a value indicating the heuristic used to generate the Name value. For example, in some cases, an instance of BMC_ComputerSystem will be identified by an external DNS name (a name configured in a DNS server). In other cases, a static IP address will be used. The naming conventions for NameFormat are: ■ IP—a valid IP address (decimal bytes delimited with dots). ■ DNS—a fully qualified host name, formatted as a HostName and a DomainName delimited with dots (the DomainName can also be made of multiple components delimited with dots). ■ TOKEN—Name holds a value defined by the TokenId (see "Additional Attributes for BMC_ComputerSystem" for more information on the TokenId).
Domain	Identifies the domain name of the computer, as known by the end points.
HostName	Specifies the local name of the computer, as known by the end points. This value must be set according to BMC N11n guidelines that specifies the algorithms and methods required to obtain the correct values.
SerialNumber	Specifies the serial number of the computer.

Key attributes of BMC_ComputerSystem

Table 1-2 show the attributes that further describe the role of an instance of `BMC_ComputerSystem`.

Table 1-2: The role of an instance of BMC_ComputerSystem (Sheet 1 of 2)

Attribute	Usage	
`CapabilityList`	Defines the main functions that the computer can perform. This is a character attribute in which you can enter any value listed in the description. You can enter more than one of these values; however, make sure that multiple values are delimited by commas. A computer system can be dedicated to a single function, such as printing, routing, or switching packets, or it can perform several functions. Typically, the `PrimaryCapability` attribute is set to the first value specified in `CapabilityList`. The following list illustrates the functions and values to assign to a `CapabilityList` attribute depending on the function of the computer.	
	■ Not Dedicated—0	■ SANBridge—26
	■ Unknown—1	■ SANRouter—27
	■ Other—2	■ SANDirector—28
	■ Storage—3	■ RAIDStorageDevice—29
	■ Router—4	■ Virtual Tape Library—30
	■ Switch—5	■ JBOD—31
	■ Layer 3 Switch—6	■ Workstation—32
	■ Central Office Switch—7	■ StorageSubsystem—33
	■ Hub—8	■ Storage Virtualizer—34
	■ Access Server—9	■ Media Library—35
	■ Firewall—10	■ ExtenderNode—36
	■ Print—11	■ NAS Head—37
	■ I/O—12	■ Self-contained NAS—38
	■ Web Caching—13	■ UPS—39
	■ Server—14	■ IP Phone—40
	■ Management —15	■ Management Controller—41
	■ Block Server —16	■ Chassis Manager—42
	■ File Server —17	■ Host-based RAID controller—43
	■ Mobile User Device —18	■ Storage Device Enclosure—44
	■ Repeater—19	■ Desktop—45
	■ Bridge/Extender—20	■ Laptop—46
	■ Gateway —21	■ Virtual Library System—47
	■ LoadBalancer—22	■ Blade System—48
	■ Mainframe—23	■ Blade Server—49
	■ SANSwitch—24	
	■ SANHub—25	

Table 1-2: The role of an instance of BMC_ComputerSystem (Sheet 2 of 2)

Attribute	Usage
`PrimaryCapability`	Describes the main function that the computer performs. By convention, `PrimaryCapability` is the first item in the `CapabilityList` attribute.
`ShortDescription`	Specifies a short description for the instance when the value of the `Name` attribute is encoded. `ShortDescription` should always be provided and set with a value that makes sense to an end user.

For example, a server with active firewall capabilities could have the values 14 (Server) or 10 (Firewall) for `CapabilityList`. `PrimaryCapability` would be set to Server if this is the main function of the system. However, a switch device would have `CapabilityList` = 5 (Switch) and `PrimaryCapability` = 5.

Additional attributes for BMC_ComputerSystem

Table 1-3 describes the attributes that provide additional information about an instance of `BMC_ComputerSystem`.

Table 1-3: Additional information for BMC_ComputerSystem

Attribute	Description
`Description`	The functions that the computer system can perform.
`DHCPUse`	Specifies whether the system is configured to use DHCP: Enabled = configured to use DHCP Disabled = not configured to use DHCP
`ManufacturerName`	The company that manufactured the computer.
`Model`	The model of the computer.
`OwnerContact`	The contact information that specifies how the primary system owner can be reached (such as phone number or email address).
`OwnerName`	The name of the primary system owner.
`TokenId`	A unique identifier populated by BMC Discovery products and used by the Reconciliation Engine (of the BMC Atrium CMDB) to identify instances.
`TotalPhysicalMemory`	The total physical memory, in kilobytes.

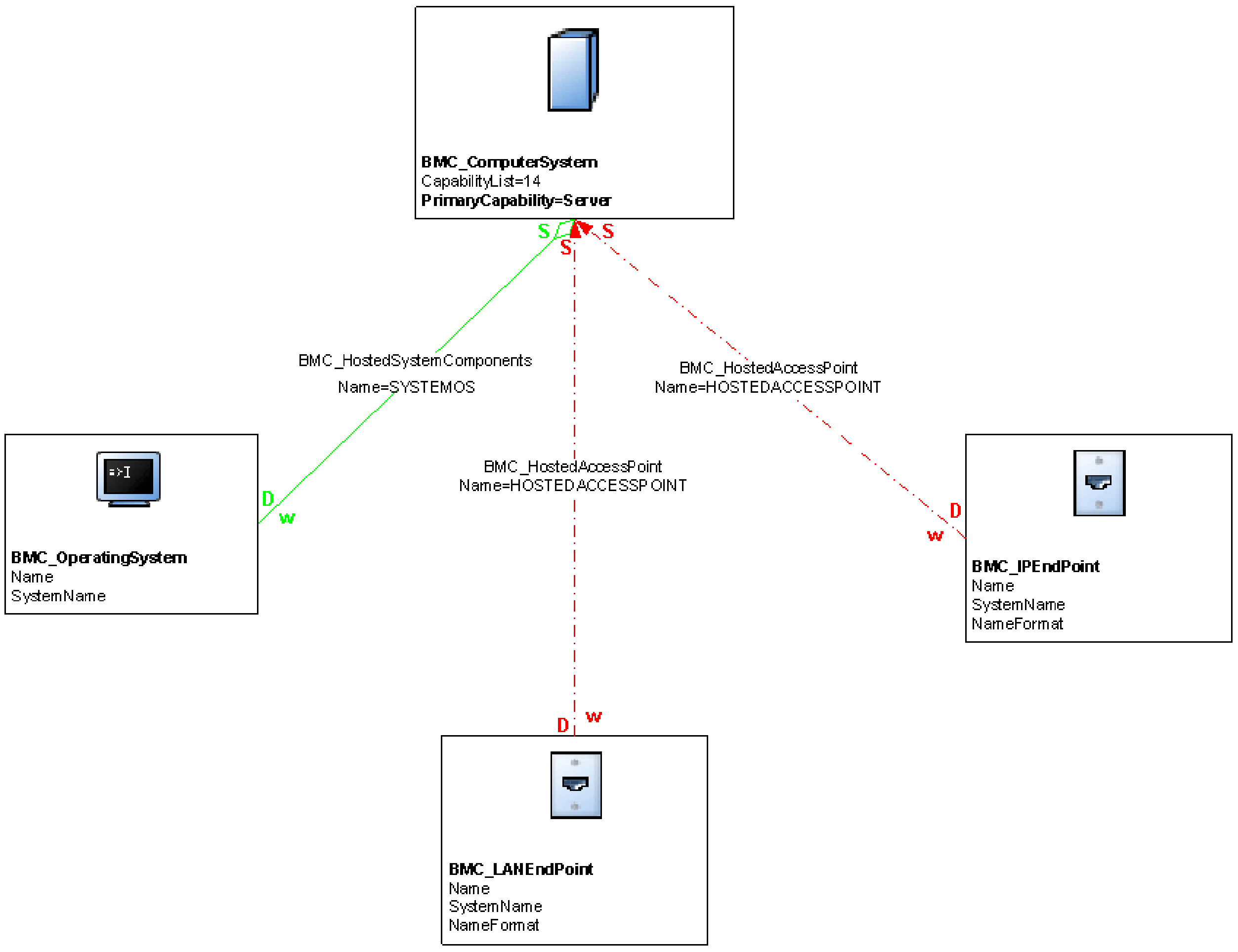
See the BMC Atrium 7.6.00 Data Model Help for more information about specific attributes.

Modeling computer systems

Computer systems are parent objects that may be represented as an aggregation of component parts (such as operating systems, hardware, software inventory, or network addresses) that are child instances related to the `BMC_ComputerSystem` instance.

Systems provide computing capabilities and aggregate one or more of the following elements: file systems, operating systems, processors, and memory (including volatile and nonvolatile storage). Therefore, additional information about a computer system might not be part of a `BMC_ComputerSystem` instance but be available from instances of other classes connected to the `BMC_ComputerSystem` instance through relationships. For example, Figure 1-1 represents a model for a server, a network-addressable computer system.

Figure 1-1: Illustrative model of a server



Servers, workstations, network devices (such as routers, switches, hubs, load balancers, or firewalls) are all instances of `BMC_ComputerSystem`, the class representing all network addressable systems. `BMC_ComputerSystem` represents an entity made up of component parts that operate as a functional whole.

The PrimaryCapability attribute is crucial to identifying whether a specific instance is a server, a router, or something else. BMC Atrium CMDB planners might use the PrimaryCapability attribute to define a vendor-specific switch used in their network, making it easy to import this data from a vendor's environment as an industry-standard item in their BMC Atrium CMDB.

Modeling software inventory and patches

Software inventory represents the products, packages, and patches that are installed on a computer. BMC_Product represents instances of installed products, whereas BMC_Patch represents instances of patches (operating system patches and product patches). BMC_Package stores information about containers or collections of related software. As a subclass of BMC_Product, BMC_Package can be used to represent suite of products. For example, you can represent Microsoft Office as an instance of BMC_Package, whereas Microsoft Word would be an instance of BMC_Product. Figure 1-2 illustrates an example model of a server with two installed products.

Figure 1-2: Illustrative model of a software inventory containing two installed products

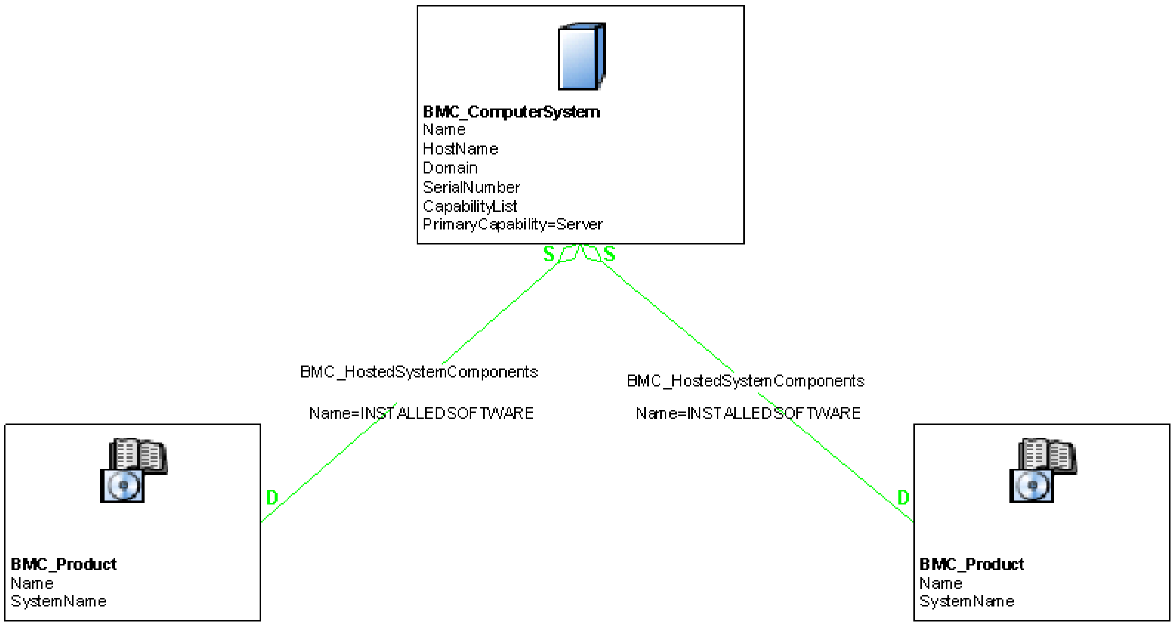
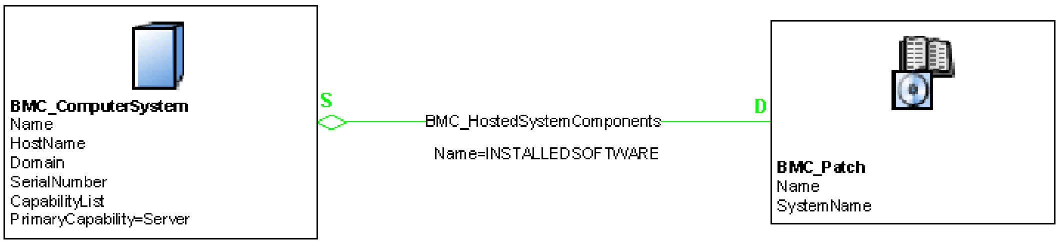


Figure 1-3 illustrates an example model of a server with one installed patch.

Figure 1-3: Illustrative model of a software inventory with a patch



You might not want to model patches in your BMC Atrium CMDB in all cases. For example, you might store patches for servers in the BMC Atrium CMDB, but it might not be necessary to do so for desktops and routers.

Both `BMC_Product` and `BMC_Patch` are subclasses of `BMC_Software` and `BMC_SystemComponent`. You should associate each instance of a product or of a patch to the parent instance of `BMC_ComputerSystem` by the `BMC_HostedSystemComponents` relationship. When modeling software inventory, be aware that the `BMC_Product` class captures only installed products or applications, not runtime aspects.

For more information about modeling runtime applications and using instances of `BMC_Product` for application modeling, see “Modeling applications” on page 45.

Logical identity of BMC_ComputerSystem (for products or patches)

Like any child instance of `BMC_ComputerSystem`, a product or a patch is identified by the `Name` attribute in conjunction with the `SystemName` attribute that represents the name of the computer instance. Thus, the `Name` attribute represents the local name of the CI in the context of the computer that is hosting it, as described in Table 1-4.

Table 1-4: Key attributes for BMC_ComputerSystem

Attribute	Usage
`Name`	Identifies the child instance in the context of the parent instance of `BMC_ComputerSystem`.
`SystemName`	Specifies the name of the computer instance. This must be the same as the parent instance of the `BMC_ComputerSystem Name` attribute. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the product or patch.

Additional attributes of BMC_ComputerSystem (for products or patches)

Table 1-5 describes attributes that provide additional information about products and patches.

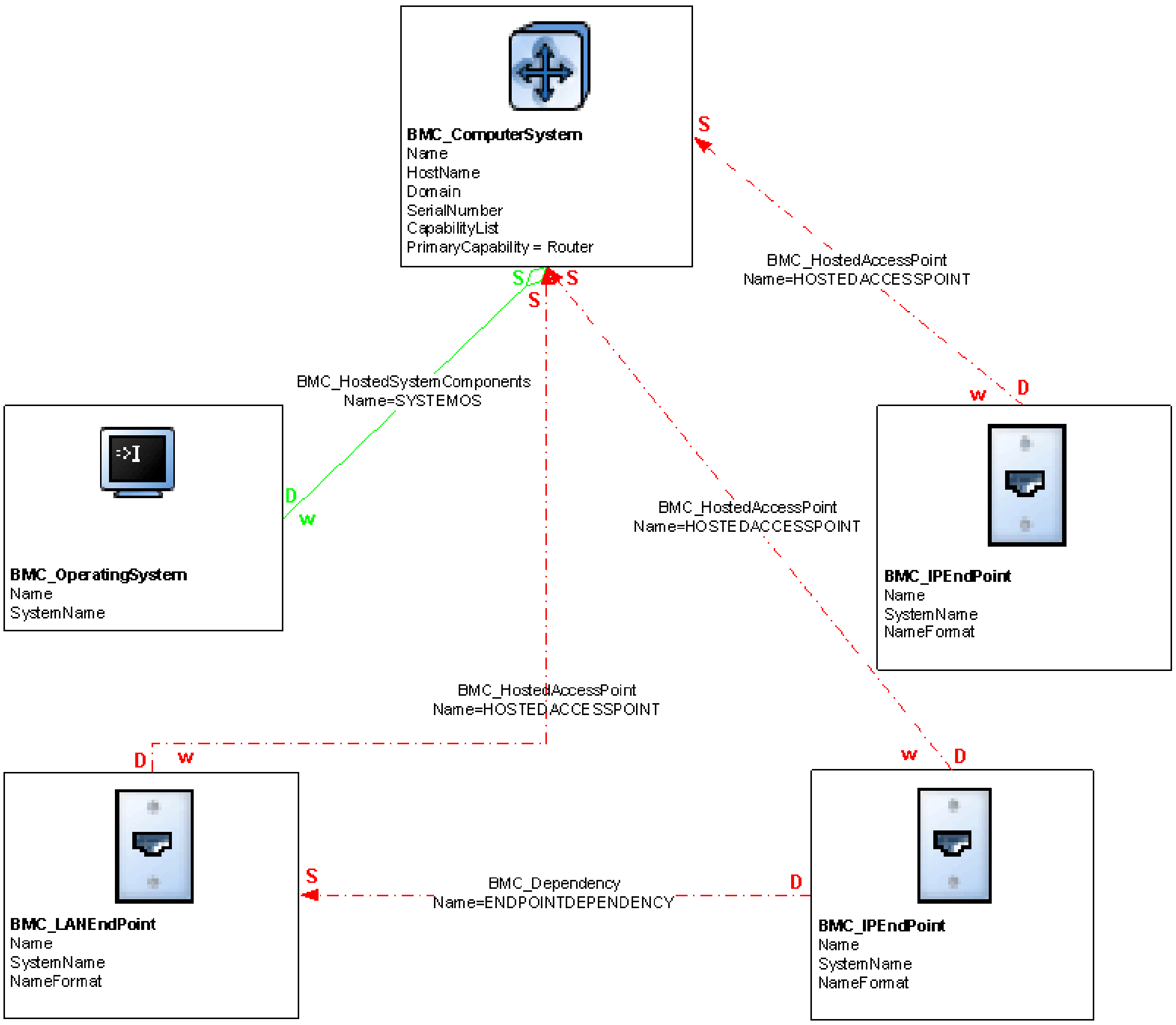
Table 1-5: Additional information about products and patches

Attribute	Description
`Description`	The description for the component.
`ManufacturerName`	The company that manufactured the component.
`SerialNumber`	The serial number of the component.
`ShortDescription`	A caption for the component.
`PatchNumber`	The version number of the patch.
`VersionNumber`	The version number of the component.

Modeling routers

Routers are modeled using the BMC_ComputerSystem class by setting the PrimaryCapability attribute to Router. Figure 1-4 illustrates an example model of a network router.

Figure 1-4: Illustrative model of a router



In the model, two BMC_IPEndpoint classes are used to represent the interfaces to the router.

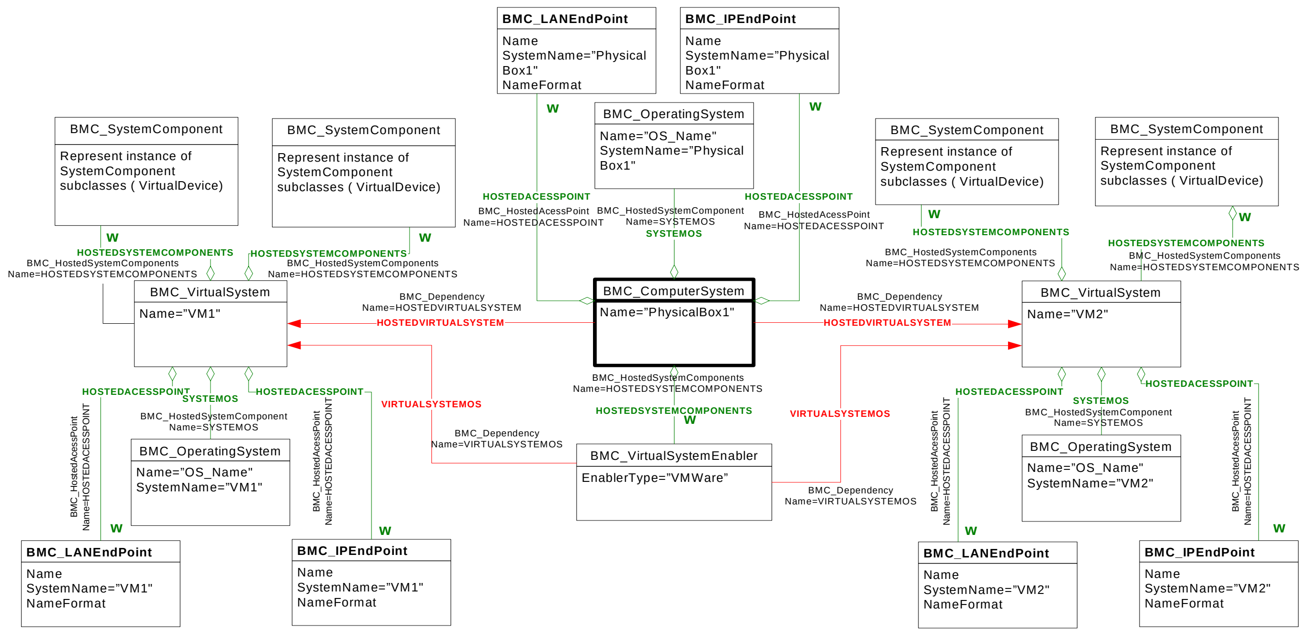
Modeling virtual systems

Virtual systems represent one or more virtual machines that are hosted by a physical computer. A virtual system has the same relationships to subcomponents and applications that a physical system does. In other words, a virtual system has an operating system (such as Windows or UNIX®), network addresses, and software. The major difference is that these subcomponents, although captured as regular CIs, are all virtual. Additionally, virtual systems can have `BMC_Genealogy` relationships that define relationships between a parent virtual system and its child virtual systems. For example, If you have a virtual system named win2k-vm1 and a clone of that system named win2k-vm2, the win2k-vm1 system is the parent and the win2k-vm2 system is the child.

When modeling virtualization in your environment, represent the physical computer system using the `BMC_ComputerSystem` class, and the virtualization software (such as Hypervisor or virtualization software), using the `BMC_VirtualSystemEnabler` class.

An example virtual system model is illustrated in Figure 1-5.

Figure 1-5: Illustrative model of a virtual system without resource pools and settings



The diagram illustrates multiple virtual systems (virtual machines, or VMs), the physical computer system that hosts them, and the virtualization operating system, which is hosted on the physical system and runs the VMs.

For example, an organization has a virtual system environment where some Solaris boxes (servers) are divided into domains, which are further divided into global zones (containers), and these global zones are further divided into zones. They need to know how to model this type of environment, specifically to address the division of the servers and to represent the virtualization software and domains.

Best practice

The recommended best practice for modeling this scenario would be:

- For the virtualization environment, use `BMC_OperatingSystem`, ensuring that the `Name` attribute is populated with the name of the OS (as illustrated in the diagram).
- For domains, use `BMC_VirtualsystemEnabler`, ensuring that the `EnablerType` attribute is set to the correct type of software (for example, LDOM Hypervisor) and relationships are set to `BMC_ComputerSystem` (the physical computer).
- For containers, create a relationship to `BMC_VirtualsystemEnabler` (Solaris Resource Manager) and relationships to the domain (or physical computer, depending on the configuration), and operating system.
- For zones, use the same model as described for containers, except that the relationship would be created to the container rather than to the domain or physical computer.

Logical identity of BMC_System and BMC_ComputerSystem (for virtual systems)

You model virtual systems as instances of `BMC_ComputerSystem`, a subclass of `BMC_System`. You should follow the same naming rules as for an instance of `BMC_ComputerSystem` class. For more information about this class, see "Logical identity of BMC_ComputerSystem" on page 18.

The key attributes for defining virtual systems are `VirtualSystemType` (for the `BMC_ComputerSystem` class) and `isVirtual` (for the `BMC_System` class). Both attributes are described in Table 1-6.

Table 1-6: Key attributes for defining virtual systems

Attribute	Usage
`VirtualSystemType`	Identifies the type of virtual machine. Values are Other (0), Unknown (1), PR/SM (2), z/VM (3), VMWare (4), Zen (10), Hyper-V (15), Solaris Container (20), VPar (25), NPar (30) and LPar (35).
`isVirtual`	Specifies whether the instance is virtual or physical. Values are NULL, No (0), or Yes (1). If you know that it is a virtual machine, use Yes. If you know that it is a physical machine, use No. If you are unsure, use NULL.

NOTE

In BMC Atrium CMDB 7.6.00, the isVirtual attribute was moved from the BMC_ComputerSystem class to the BMC_SystemComponent class to expand the virtualization scope beyond computer systems, enabling you to model potential future virtualizable entities, such as applications.

To ensure correct reconciliation with data created by BMC Software products, use NULL instead of No for the isVirtual attribute to represent an instance that is not virtual.

For more information on using the isVirtual attribute in virtualization modeling, see "Modeling orphan disks" on page 34.

Logical identity of BMC_VirtualSystemEnabler

The BMC_VirtualSystemEnabler class stores information about software that enables a collection of virtual computer systems to run on a single physical computer system (for example, VMware). This class is used to capture the virtualization OS, such as operating systems that run virtual machines (including VMware images, Solaris zones, AIX logical partitions, HP-UX virtual partitions, and so forth).

The BMC_VirtualSystemEnabler class is associated to the parent computer system instance by the BMC_HostedSystemComponents relationship. As a subclass of BMC_ComputerSystem, any instance (representing a new business CI) of the BMC_VirtualSystemEnabler class is identified, at minimum, by the Name and SystemName attributes.

The key attribute for BMC_VirtualSystemEnabler is EnablerType, described in Table 1-7.

Table 1-7: Key attributes for EnablerType

Attribute	Usage
EnablerType	Specifies the virtualization software or OS. The possible values are Other (0, the default), Unknown (1), PR/SM (2), z/VM (3), VMWare Server (4), Solaris Resource Manager (5), LPar (6), VPar (7), HP nPartitions (20), Integrity VM (25), Microsoft Hyper-V (30), VMWare ESX Server (35), VMWare Workstation (40), Xen Hypervisor (45), and LDOM Hypervisor (50).

For a complete list of attributes for the BMC_VirtualSystemEnabler class, see the BMC Atrium CMDB 7.6.00 Data Model Help.

Logical identity of BMC_VirtualSystemSettingData

The BMC_VirtualSystemSettingData class (derived from BMC_Settings) defines the virtual aspects of a virtual system through a set of virtualization-specific properties. BMC_VirtualSystemSettingData is also used as the top level class of virtual system configurations that model configuration information about virtual systems and their components. A virtual system configuration consists of one top-level instance of the BMC_VirtualSystemSettingData class that aggregates a number of instances of the BMC_ResourceAllocationSettingData class using the BMC_Component association.

Use BMC_VirtualSystemSettingData and BMC_ResourceAllocationSettingData to represent virtual system settings, and BMC_ResourcePool to model resource pools.

For example, virtual system configurations may be used to reflect configurations of virtual systems that are defined at a virtualization platform virtual systems that are currently active input requests to create new virtual systems input requests to modify existing virtual systems snapshots of virtual systems.

Key attributes for defining the virtual aspects using the BMC_VirtualSystemSettingData class are described in the following table.

Table 1-8: Key attributes for BMC_VirtualSystemSettingData

Attribute	Usage
VirtualSystemState	Specifies the state of the virtual machine. Values are Other (0), Unknown (1), Active (10), InActive (15), Suspended (20), and Disabled (25).
VirtualSystemType	Specifies a type of virtual system. Implementations are commonly capable of supporting various implementation-defined virtual system types. Use this attribute to determine the types of virtual systems supported for your configuration. For example, you might have a management application that uses an instance of the BMC_VirtualSystemSettingData class as an input parameter to an operation that creates or modifies a virtual system. Before configuring this application, you should first determine the set of valid virtual system types that are supported by the virtualization platform hosting the virtual system. To do this, inspect the values of the VirtualSystemTypesSupported array property of the instance of the CIM_VirtualSystemManagementCapabilities class that describes the capabilities of the virtualization platform. Values are Other (0),Unknown (1), LPAR (2), VM / VM Guest (3), VMware (4),Xen (20), LDOM (25), Solaris Container (30), HP nPartitions (35), VPar (40), and Microsoft Hyper-V (45).

Additional attributes for BMC_VirtualSystemSettingData

Table 1-9 describes attributes that provide additional information about an instance of BMC_VirtualSystemSettingData. All three of these attributes are introduced in BMC Atrium CMDB 7.6.00.

Each of these attributes provides enhancements to the class to account for Virtual Machine sprawl in a data center, enabling organizations to accurately track and act on VM provisioning and decommissioning to streamline their virtual environments.

Table 1-9: Additional attributes for BMC_VirtualSystemSettingData

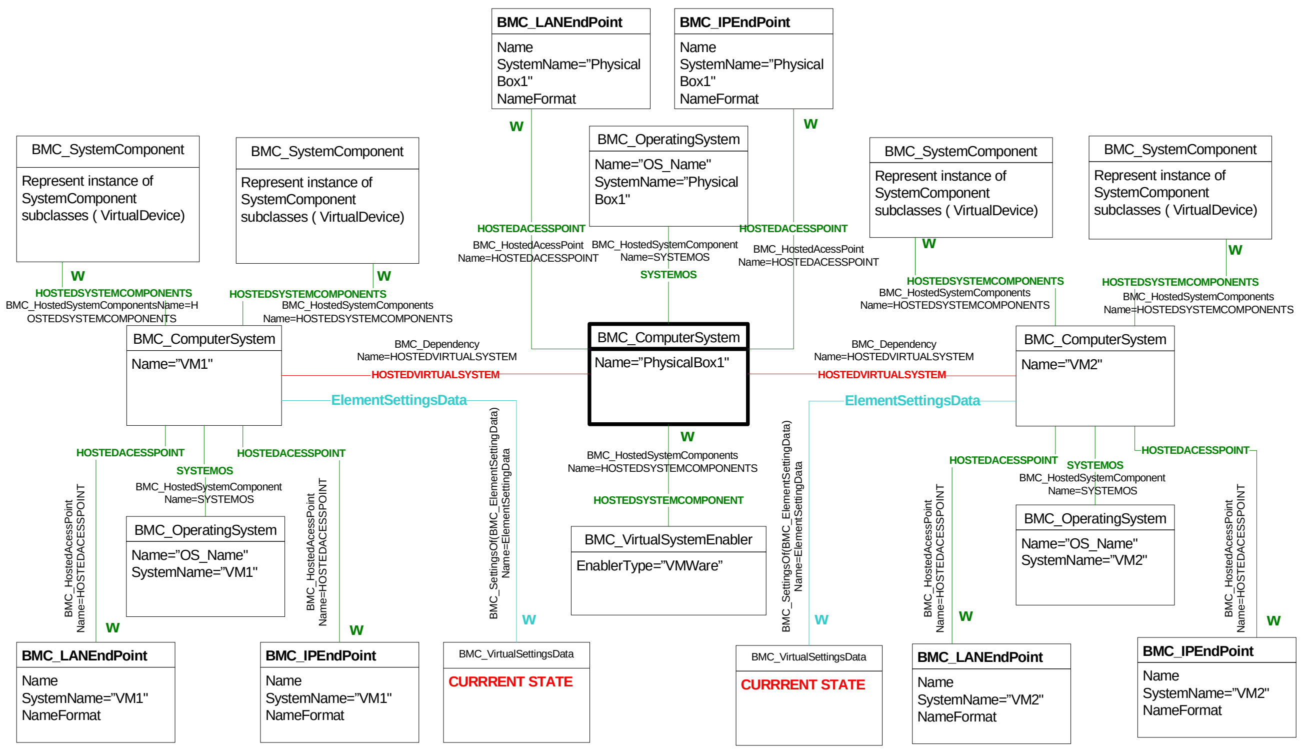
Attribute	Description
ActualProvisionDate	The date and time that a specific VM was provisioned. This attribute is important for enabling successful reporting on BMC Dashboards.
ProposedDecommissionDate	The date and time that the VM will be removed from the environment. One of the biggest problems causing virtual system sprawl is that organizations do not decommission VMs and track the information. This attribute helps to drive workflow actions such as: ■ Notifying users that their VM is about to be decommissioned and enabling them to extend the time before that occurs. ■ Automatically creating change requests to begin the decommissioning process. ■ Automatically starting a job in BMC BladeLogic, BMC Atrium Orchestrator, and other consuming applications to decommission the VM.
ActualDecommissionDate	The date of the actual decommission. There can be an extension to that date to enable you to report or track a VM. You can also compare differences between the decommission date and actual decommission date, or block future updates to the extension date.

Logical identity of BMC_ResourceAllocationSettingData

The BMC_ResourceAllocationSettingData class (derived from BMC_Settings) represents settings that specifically relate to an allocated resource that is outside the scope of the CIM class (which is typically used to represent the resource itself). These settings contain information specific to the allocation that may not be visible to the consumer of the resource itself. For example, a virtual processor may look like a 2 ghz processor to the consumer (as a virtual computer system); however, the virtualization system may use time-slicing to schedule the virtual processor so it can only use 1 ghz of CPU speed.

Use BMC_VirtualSystemSettingData and BMC_ResourceAllocationSettingData to represent virtual system settings, and BMC_ResourcePool to model resource pools. Figure 1-6 illustrates an example virtual system model including settings.

Figure 1-6: Illustrative model of a virtual system with resource settings



The key attribute for defining settings using the BMC_ResourceAllocationSettingData class is ResourceType, described in Table 1-10.

Table 1-10: Key attribute for ResourceType

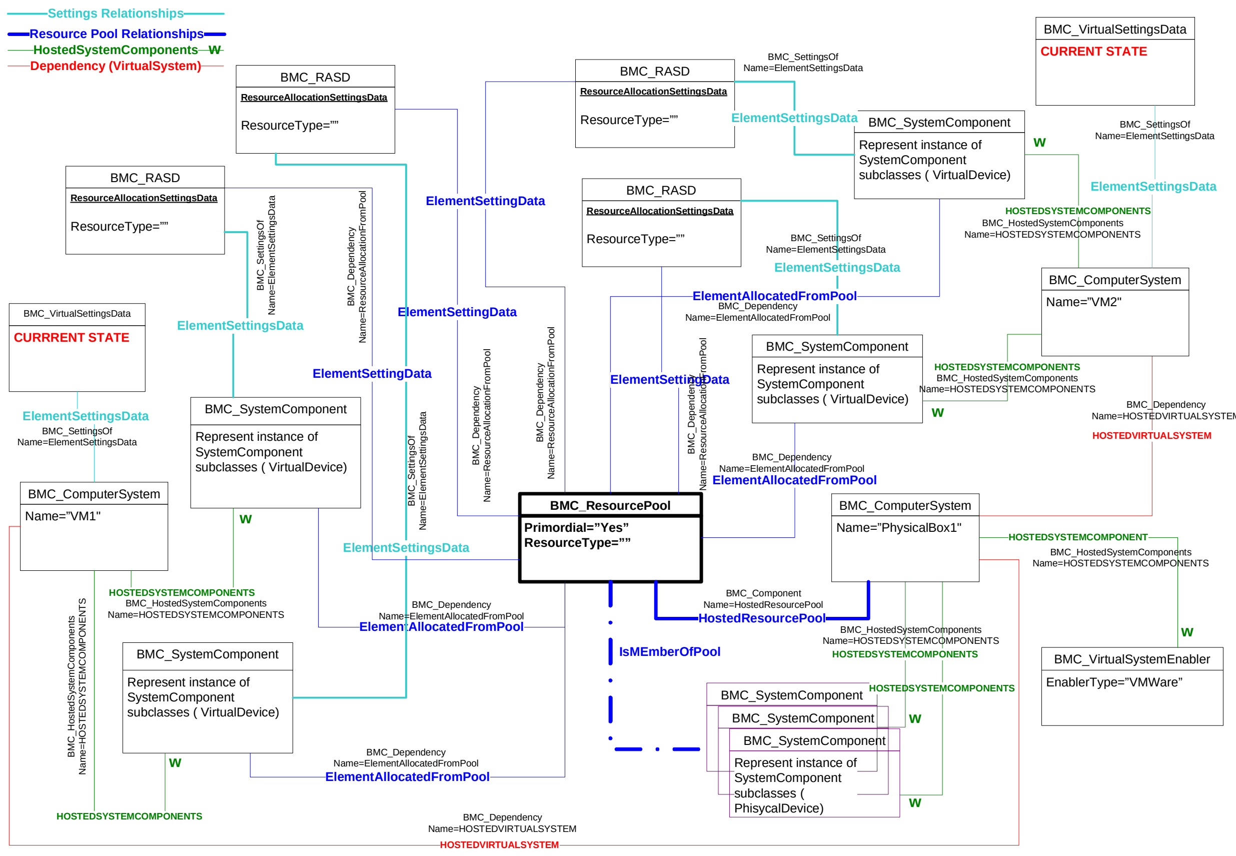
Attribute	Usage
ResourceType	Specifies the type of resource that this allocation setting represents. Values are Other (0), Computer System (2), Processor (3), Memory (4), IDE Controller (5), Parallel SCSI HBA (6), FC HBA (7), iSCSI HBA (8), IB HCA (9), Ethernet Adapter (10), Other Network Adapter (11), I/O Slot (12), I/O Device (13), Floppy Drive (14), CD Drive (15), DVD drive (16), Disk Drive (17), Tape Drive (18), Storage Extent (19), Other storage device (20), Serial port (21), Parallel port (22), USB Controller (23), Graphics controller (24), IEEE 1394 Controller (25), Partitionable Unit (26), Base Partitionable Unit (27), Power (28), Cooling Capacity (29), Ethernet Switch Port (30).

Logical identity of BMC_ResourcePool

The BMC_ResourcePool class (derived from BMC_LogicalEntity) serves as a logical entity (with associated controls) provided by the host system to allocate and assign resources. A resource pool may be used to allocate resources of a specific type. Hierarchies of resource pools may be created to provide administrative control over allocations. In cases where resources are subdivided, multiple resource pools may exist (for example, nodal boundaries in NUMA-like systems).

In systems that support over-commitment, pools represent the reservable capacity, not an upper bound or limit on the maximum amount that can be allocated. Admission control during power-on may detect and prevent systems from powering due to resource exhaustion. For example, over-commitment on a resource pool with ResourceType set to Memory would require that sufficient space be available in a backing store that might be managed through a storage resource pool. Figure 1-7 illustrates an example virtual system model including resource pools.

Figure 1-7: Illustrative model of a virtual system with resource pools


Settings Relationships
Resource Pool Relationships
HostedSystemComponents W
Dependency (VirtualSystem)
BMC_RASD
ResourceAllocationSettingsData
ResourceType=""
BMC_VirtualSettingsData
CURRENT STATE
CURRRENT STATE
BMC_SystemComponent
Represent instance of SystemComponent subclasses (VirtualDevice)
BMC_ComputerSystem
Name="VM1"
Name="VM2"
Name="PhysicalBox1"
BMC_ResourcePool
Primordial="Yes"
ResourceType=""
BMC_VirtualSystemEnabler
EnablerType="VMWare"
Represent instance of SystemComponent subclasses (PhisycalDevice)
ElementSettingsData
ElementSettingData
ElementAllocatedFromPool
HostedResourcePool
IsMEmberOfPool
HOSTEDSYSTEMCOMPONENTS
HOSTEDVIRTUALSYSTEM
BMC_SettingsOf Name=ElementSettingsData
BMC_Dependency Name=ElementAllocatedFromPool
BMC_Dependency Name=ResourceAllocationFromPool
BMC_HostedSystemComponents Name=HOSTEDSYSTEMCOMPONENTS
BMC_Dependency Name=HOSTEDVIRTUALSYSTEM
BMC_Component Name=HostedResourcePool
W

Key attributes for defining resource pools using the BMC_ResourcePool class are described in the following table.

Table 1-11: Key attributes for BMC_ResourcePool

Attribute	Usage
Primordial	Specifies how the ResourcePool is used in the activity of resource management. If set to true, this attribute indicates that the ResourcePool is a base from which resources are drawn and returned in the activity of resource management. It also indicates that this ResourcePool shall not be created or deleted by consumers of this model. However, other actions (whether they are modeled or not), may affect the characteristics or size of primordial ResourcePools. If set to false (the default), this attribute indicates that the ResourcePool serves as a concrete Resource Pool, meaning that is subject to resource allocation services functions. This distinction is important, because higher-level ResourcePools may be assembled using the Component or ElementAllocatedFromPool associations. Although the higher-level abstractions can be created and deleted, the most basic, hardware-based ResourcePools (such as primordial pools) cannot. Instead, they are physically realized as part of the system, or are actually managed by some other system and imported as if they were physically realized.
ResourceType	Specifies the type of resource that the resource pool may allocate. Values are Other (0), Computer System (2), Processor (3), Memory (4), IDE Controller (5), Parallel SCSI HBA (6), FC HBA (7), iSCSI HBA (8), IB HCA (9), Ethernet Adapter (10), Other Network Adapter (11), I/O Slot (12), I/O Device (13), Floppy Drive (14), CD Drive (15), DVD drive (16), Disk Drive (17), Tape Drive (18), Storage Extent (19), Other storage device (20), Serial port (21), Parallel port (22), USB Controller (23), Graphics controller (24), IEEE 1394 Controller (25), Partitionable Unit (26), Base Partitionable Unit (27), Power (28), Cooling Capacity (29), Ethernet Switch Port (30), Logical Disk (31), Storage Volume (32), Ethernet Connection (33).

Deprecated classes for virtual systems

The following classes were deprecated in BMC Atrium CMDB 7.5.00, and are no longer used for modeling virtualized environments:

- BMC_VirtualSystem (including all subclasses)
- BMC_VMWare
- BMC_VMWareVirtualSystem
- BMC_UnixVirtualSystem
- BMC_MFVirtualSystem
- BMC_MFVirtualSystemEnabler
- BMC_LPAR

Relationships used for virtual systems

Table 1-12 describes the relationships for virtual systems.

Table 1-12: Relationships used for virtual systems

Relationship	Relationship class	Value of Name attribute
Resource allocation and resource pool	`BMC_Dependency`	ResourceAllocationFromPool
Virtual system and resource pool	`BMC_Component`	HostedResourcePool
Allocated resource and resource pool	`BMC_Dependency`	ElementAllocatedFromPool
Managed elements and setting data	`BMC_SettingsOf`	ElementSettingData
Allocated resource and resource pool	`BMC_MemberOfCollection`	IsMemberOfPool

Modeling orphan disks

The number of virtual machines (VM) running in a virtualized infrastructure increases over time, typically because the ease of creating new VMs causes IT administrators to forget or neglect the business necessity for adding them. In most cases, the additional VMs are provisioned to address an expanding data center, but they are not necessary and can actually be prohibitive. This phenomenon, becoming increasingly popular in today's IT environments, is known as VM sprawl.

VM sprawl can not only cause the overuse of the infrastructure, but can significantly increase the cost of licenses for virtual machines that are not required. The VMs that are not necessary for the environment are actually virtual hard disks that are not connected to a virtual machine (known as orphan disks).

Orphan disks take the form of .vmdk files that are typically left in the infrastructure, taking up valuable disk space. When removing a virtual machine from the inventory, IT administrators might simply forget to delete the vmdk file.

Modeling orphan disks helps BMC Atrium CMDB planners identify these disks so that they can increase the efficiency of the data center and prevent VM sprawl.

Logical identity of BMC_DiskDrive (for orphan disks)

You model orphan disks as an instance of the `BMC_DiskDrive` class with the value of the `isVirtual` attribute set to Yes. In BMC Atrium CMDB 7.6, the `isVirtual` attribute has been moved to the `BMC_SystemComponent` class so that other child classes of `BMC_SystemComponent` can also use the attribute in the future if required.

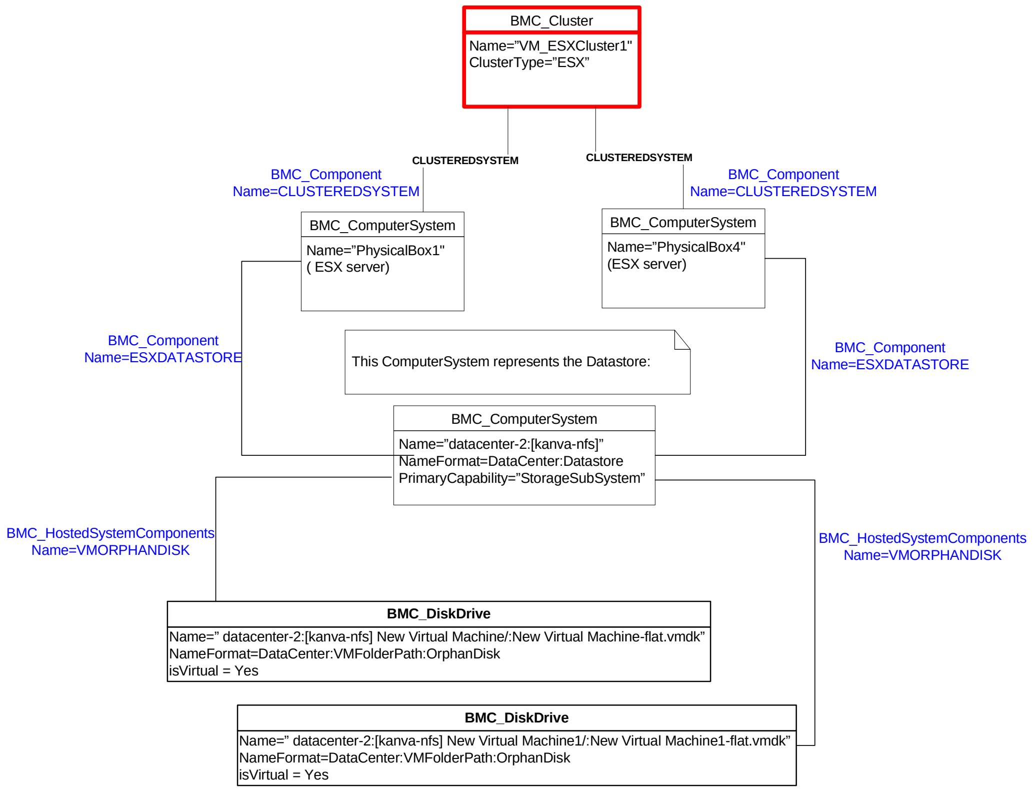
The key attribute for defining an orphan disk using the `BMC_DiskDrive` class is `isVirtual`, described in Table 1-13.

Table 1-13: Key attribute for isVirtual

Attribute	Usage
`isVirtual`	Attribute that specifies whether the instance is virtual or physical. To ensure correct reconciliation with data created by BMC Software products, use NULL instead of No for an instance that is not virtual. Set the value to Yes to indicate an orphan disk.

Figure 1-8 illustrates an example of a clustered, simple virtual system.

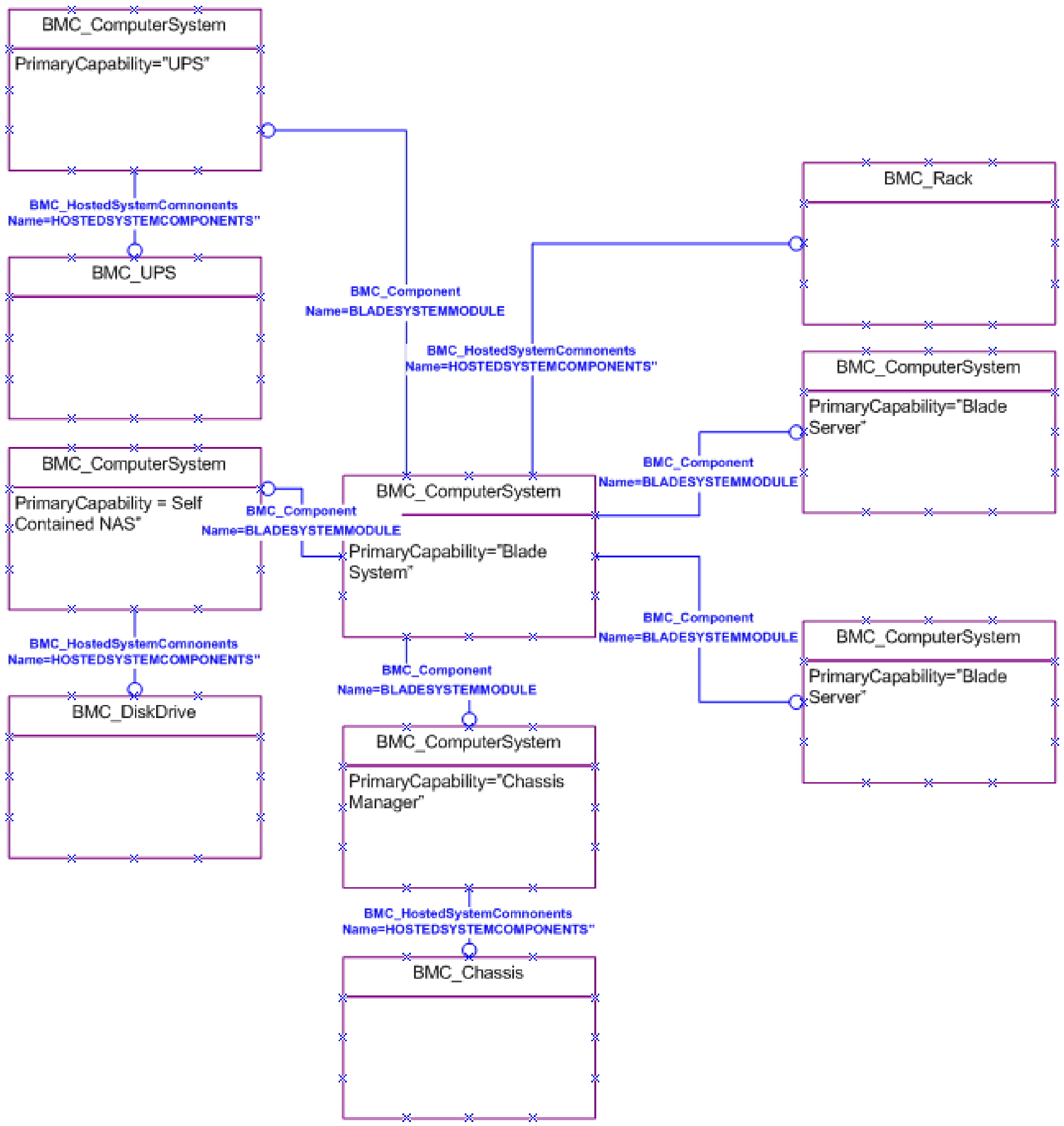
Figure 1-8: Illustrative model of a clustered, simple virtual system


BMC_Cluster
Name="VM_ESXCluster1"
ClusterType="ESX"
CLUSTEREDSYSTEM
CLUSTEREDSYSTEM
BMC_Component
Name=CLUSTEREDSYSTEM
BMC_Component
Name=CLUSTEREDSYSTEM
BMC_ComputerSystem
Name="PhysicalBox1"
(ESX server)
BMC_ComputerSystem
Name="PhysicalBox4"
(ESX server)
BMC_Component
Name=ESXDATASTORE
This ComputerSystem represents the Datastore:
BMC_Component
Name=ESXDATASTORE
BMC_ComputerSystem
Name="datacenter-2:[kanva-nfs]"
NameFormat=DataCenter:Datastore
PrimaryCapability="StorageSubSystem"
BMC_HostedSystemComponents
Name=VMORPHANDISK
BMC_HostedSystemComponents
Name=VMORPHANDISK
BMC_DiskDrive
Name=" datacenter-2:[kanva-nfs] New Virtual Machine/:New Virtual Machine-flat.vmdk"
NameFormat=DataCenter:VMFolderPath:OrphanDisk
isVirtual = Yes
BMC_DiskDrive
Name=" datacenter-2:[kanva-nfs] New Virtual Machine1/:New Virtual Machine1-flat.vmdk"
NameFormat=DataCenter:VMFolderPath:OrphanDisk
isVirtual = Yes

Modeling blade systems

Routers are modeled using the `BMC_ComputerSystem` class by setting the `PrimaryCapability` attribute to blade server. Figure 1-9 illustrates an example model of a blade environment.

Figure 1-9: Illustrative model of a blade environment


BMC_ComputerSystem
PrimaryCapability="UPS"
BMC_HostedSystemComponents Name=HOSTEDSYSTEMCOMPONENTS"
BMC_UPS
BMC_ComputerSystem
PrimaryCapability = Self Contained NAS"
BMC_HostedSystemComponents Name=HOSTEDSYSTEMCOMPONENTS"
BMC_DiskDrive
BMC_Component Name=BLADESYSTEMMODULE
BMC_Component Name=BLADESYSTEMMODULE
BMC_HostedSystemComponents Name=HOSTEDSYSTEMCOMPONENTS"
BMC_ComputerSystem
PrimaryCapability="Blade System"
BMC_Component Name=BLADESYSTEMMODULE
BMC_ComputerSystem
PrimaryCapability="Chassis Manager"
BMC_HostedSystemComponents Name=HOSTEDSYSTEMCOMPONENTS"
BMC_Chassis
BMC_Rack
BMC_ComputerSystem
PrimaryCapability="Blade Server"
BMC_Component Name=BLADESYSTEMMODULE
BMC_Component Name=BLADESYSTEMMODULE
BMC_ComputerSystem
PrimaryCapability="Blade Server"

As illustrated, to specify the blade system, the `PrimaryCapability` attribute is set to *Blade Server* for the `BMC_ComputerSystem` class. See "Key attributes of BMC_ComputerSystem" on page 19 for more information on the `PrimaryCapability` attribute.

Modeling operating systems

An operating system is software or firmware that controls the operation of a computer and directs the processing of programs. This section describes how to model Windows and UNIX operating systems.

To model a Windows or UNIX operating system, create an instance of the BMC_OperatingSystem class. Associate the instance to the parent BMC_ComputerSystem instance by a BMC_HostedSystemComponents relationship.

NOTE

This class is not reserved for servers and workstations only but is used to capture any type of operating system, such as the IOS for a Cisco network switch or router.

Logical identity of BMC_OperatingSystem

As with any related component of BMC_ComputerSystem, an operating system is identified by the Name attribute in conjunction with the SystemName attribute that represents the name of the parent instance of the computer. Therefore, the Name attribute represents the local name of the operating system CI in the context of the computer that is hosting it, as described in Table 1-14.

Table 1-14: Key attributes for BMC_OperatingSystem

Attribute	Usage
Name	Identifies the name of the child instance in the context of the parent instance of BMC_ComputerSystem. If multiple operating systems are installed on the same computer, Name must be structured so that the multiple instances have different names.
SystemName	Specifies the name of the system. This must be the same as the parent instance of the BMC_ComputerSystem Name attribute. This attribute is automatically populated from the related CI when a weak relationship is created between the operating system and a computer system.

Additional attributes for BMC_OperatingSystem

Table 1-15 describes additional attributes of BMC_OperatingSystem.

Table 1-15: Additional attributes of BMC_OperatingSystem

Attribute	Usage
Description	The description for the operating system.
ManufacturerName	The company that manufactured the operating system.
SerialNumber	The serial number of the operating system.
ShortDescription	A caption for the operating system.
VersionNumber	The version number of the operating system.

Modeling hardware components

The hardware components that make up a computer system are captured by subclasses of BMC_HardwareSystemComponent. Generally, one subclass represents one type of hardware component. Examples of hardware components include:

- Disk drive—Machine that reads data from and writes data to a disk.
- Disk partition—Logical allocation of space on a disk drive.
- Monitor—Video device attached to computer systems that displays computer operations.
- Keyboard—Set of typewriter-like keys that enables you to enter data into a computer.
- Memory—Stores information about internal storage areas in a computer.
- Processor—Device that interprets a machine instructions in a computer.
- Network port—Interfaces that connect network drives to computer systems.

For example, you might identify a specific processor as an instance of BMC_HardwareSystemComponent. Each instance representing a hardware component is associated to the parent BMC_ComputerSystem instance by the BMC_HostedSystemComponents relationship.

Logical identity of BMC_HardwareSystemComponent

Like any child instance of BMC_ComputerSystem, a hardware component is identified by the Name attribute in conjunction with the SystemName attribute that represents the name of the parent computer instance. Therefore, the Name attribute represents the local name of the hardware CI in the context of the computer that is hosting it, as described in Table 1-16.

Table 1-16: Key attributes for BMC_HardwareSystemComponent

Attribute	Usage
Name	Identifies the child instance in the context of the parent instance of BMC_ComputerSystem.
SystemName	Specifies the name of the system. This must be the same as the parent instance of BMC_ComputerSystem Name attribute. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes for BMC_HardwareSystemComponent

Table 1-17 describes additional attributes of `BMC_HardwareSystemComponent`.

Table 1-17: Additional attributes of BMC_HardwareSystemComponent

Attribute	Description
`Description`	The description for the component.
`ManufacturerName`	The company that manufactured the component.
`SerialNumber`	The serial number of the component.
`ShortDescription`	A caption for the component.
`VersionNumber`	The version number of the component.

Because each hardware component contains attributes specific to its type, see the BMC Atrium CMDB 7.6.00 Data Model Help for a complete list of `BMC_HardwareSystemComponent` types and attributes to ensure you can accurately and completely represent your specific hardware CIs.

Modeling access points

A computer provides functions for other entities to use. Access points represent those available functions. Each access point represents the configuration of access to a function or the ability to invoke a service and is modeled by the `BMC_AccessPoint` class. This characteristic is further defined by `BMC_ProtocolEndpoint`, the only direct subclass of `BMC_AccessPoint`. Among other types of access points, a network address such as an IP address, MAC address, or IPX address, is captured as a subclass of `BMC_AccessPoint`.

Instances of the `BMC_AccessPoint` class are related to a computer system through a `BMC_HostedAccessPoint` dependency relationship. Access points exist within the context of a computer system, and are associated to their parent instance of the system through the `BMC_HostedAccessPoint` dependency relationship.

For example, Figure 1-4 on page 25 illustrates an example of a computer system's relationship to an IP endpoint, in the context of modeling a router.

Logical identity of BMC_IPEndpoint

An IP address is modeled as an instance of `BMC_IPEndpoint`. Like any child instance of `BMC_ComputerSystem`, an instance of `BMC_IPEndpoint` is identified by the `Name` attribute in conjunction with the `SystemName` attribute that represents the name of the parent instance of the computer. Therefore, the `Name` attribute represents the local name of the CI in the context of the computer that is hosting it, as described in Table 1-18.

Table 1-18: Key attributes for BMC_IPEndpoint

Attribute	Usage
`Name`	Identifies the child instance in the context of the parent instance of the `BMC_ComputerSystem`. `Name` must be an IPv4 or IPv6 address, and must be formatted as decimal numbers delimited by a period, with no leading zeros.
`NameFormat`	Sets the heuristic used to generate the `Name` value. This value must be set to IP.
`SystemName`	Specifies the name of the system. This must be the same as the parent instance of the `BMC_ComputerSystem` `Name` attribute. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes for BMC_IPEndpoint

Table 1-19 details additional attributes of `BMC_IPEndpoint`.

Table 1-19: Additional attributes of BMC_IPEndpoint

Attribute	Description
`Address`	The IP address. This value must be compliant with `AddressType`.
`AddressType`	The enumeration that defines the type of address. This value must be set to either 0 (Unknown), 1 (IPv4), or 2 (IPv6).
`DNSName`	The system name based on its DNS name. The DNS name corresponds to the IP address; therefore, when you want to search a system by DNS name, you should look up the IPEndpoint CIs, and then the parent computer.
`ProtocolType`	The enumeration that categorizes and classifies instances of this class.
`ShortDescription`	A caption of the IP address.
`SubnetMask`	The IP address subnet mask.
`ManagementAddress`	The selection that defines if the IP address is a management address such as a Discovered Address or an SNMP address.

Logical identity of BMC_LANEndpoint

A MAC address is modeled as an instance of `BMC_LANEndpoint`. Like any child instance of `BMC_ComputerSystem`, a MAC address is identified by the `Name` attribute in conjunction with the `SystemName` attribute that represents the name of the parent instance of the computer. Therefore, the `Name` attribute represents the local name of the CI in the context of the computer that is hosting it, as described in Table 1-20.

Table 1-20: Key attributes for BMC_LANEndpoint

Attribute	Usage
`Name`	Identifies the MAC address. The value must be an address suffixed by an index. The index uniquely identifies a MAC address for situations where multiple identical MAC addresses are configured within the same system. The index is generally the index of the MAC address entry in the SNMP MIB.
`NameFormat`	Specifies the heuristic used to generate the `Name` value. This value must be set to MACAddress:Index.
`SystemName`	Specifies the name of the system. This must be the same as the parent instance of the `BMC_ComputerSystem` `Name` attribute. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes of BMC_LANEndpoint

Table 1-21 describes additional attributes of `BMC_LANEndpoint`.

Table 1-21: Additional attributes of BMC_LANEndpoint

Attribute	Description
`Address`	The MAC address.
`ProtocolType`	The enumeration that categorizes and classifies instances of this class, such as for 14 (for Ethernet).
`ShortDescription`	A caption of the MAC address.

Access point binding

Some access points use the services provided through another access point. You can use access point binding to establish a layering of two protocols, with the upper layer represented by the dependent and the lower layer represented by the antecedent.

This binding is modeled in the CDM by the `BMC_Dependency` relationship with the `Name` attribute set to BindsTo.

Modeling network interfaces and addresses

Network interfaces are captured by instances of BMC_NetworkPort. Although model extensions might define subclasses (like a FiberChannel port), the class that you should use for network interfaces is BMC_NetworkPort. Like other hardware components, each instance of a network port is associated to the parent instance of the BMC_ComputerSystem by the BMC_HostedSystemComponents relationship.

Network addresses are captured by BMC Discovery products as access points (inherited from BMC_AccessPoint) and therefore must always be associated to their parent instance of the computer through the BMC_HostedAccessPoint relationship. Also, a network address can have a relationship to the network interface for which it is configured. This relationship is modeled by a BMC_Dependency relationship in which the network interface is the antecedent (source) and the network address is the dependent (destination).

For more information on modeling network addresses, including an illustration of the relationships used in the model, see "Modeling network topology" on page 75.

Logical identity of BMC_NetworkPort

Like any child instance of BMC_ComputerSystem, a network port is identified by the Name attribute in conjunction with the SystemName attribute that represents the name of the parent instance of the computer. Therefore, the Name attribute represents the local name of the CI in the context of the computer that is hosting it, as described in Table 1-22.

Table 1-22: Key attributes for BMC_NetworkPort

Attribute	Usage
Name	Identifies the network address. This must be an address suffixed by an index. The index uniquely identifies a MAC address for situations where multiple identical MAC addresses are configured within the same system. The index is generally the index of the MAC address entry in the SNMP MIB.
NameFormat	Specifies the heuristic used to generate the Name value. For instance, in many cases, network interfaces are best discovered and identified using SNMP information.
PhysicalIndex	Specifies the index, which must be a valid SNMP index relative to the SNMP IF Table of the computer.
SystemName	Identifies the name of the system. This must be the same as the parent instance of the BMC_ComputerSystem Name attribute. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes for BMC_NetworkPort

Table 1-23 describes additional attributes of `BMC_NetworkPort`.

Table 1-23: Additional attributes for BMC_NetworkPort

Attribute	Description
`AutoSense`	The Boolean value that indicates whether the port can automatically determine the speed or other communications characteristics of the connected network media.
`Description`	The description for the component.
`FullDuplex`	The Boolean value that indicates whether the port is operating in full duplex mode (carrying signals in both directions).
`LinkTechnology`	The enumeration of the types of link technologies, with values such as Unknown, Other, Ethernet, IB, FC, FDDI, ATM, Token Ring, Frame Relay, Infrared, BlueTooth, or Wireless LAN.
`ManufacturerName`	The company that manufactured the component.
`MaxSpeed`	The maximum bandwidth of the port in bits/second.
`NetworkAddresses`	The list of strings specifying the network addresses for the port.
`PermanentAddress`	The network address hard-coded into the port. This address can be changed by a firmware upgrade or software reconfiguration. If it is changed, update this field. If no hard-coded address exists for the network port, leave this attribute blank.
`PhysicalDescription`	The physical description or location for the port, such as slot3/port4.
`PortType`	The enumeration of the types of ports, with values such as Ethernet, FDDI, Token Ring, WAN, or Unknown.
`ShortDescription`	A caption for the component.
`SerialNumber`	The serial number of the component.
`SpeedConfigured`	The maximum bandwidth of the port in bits/second.

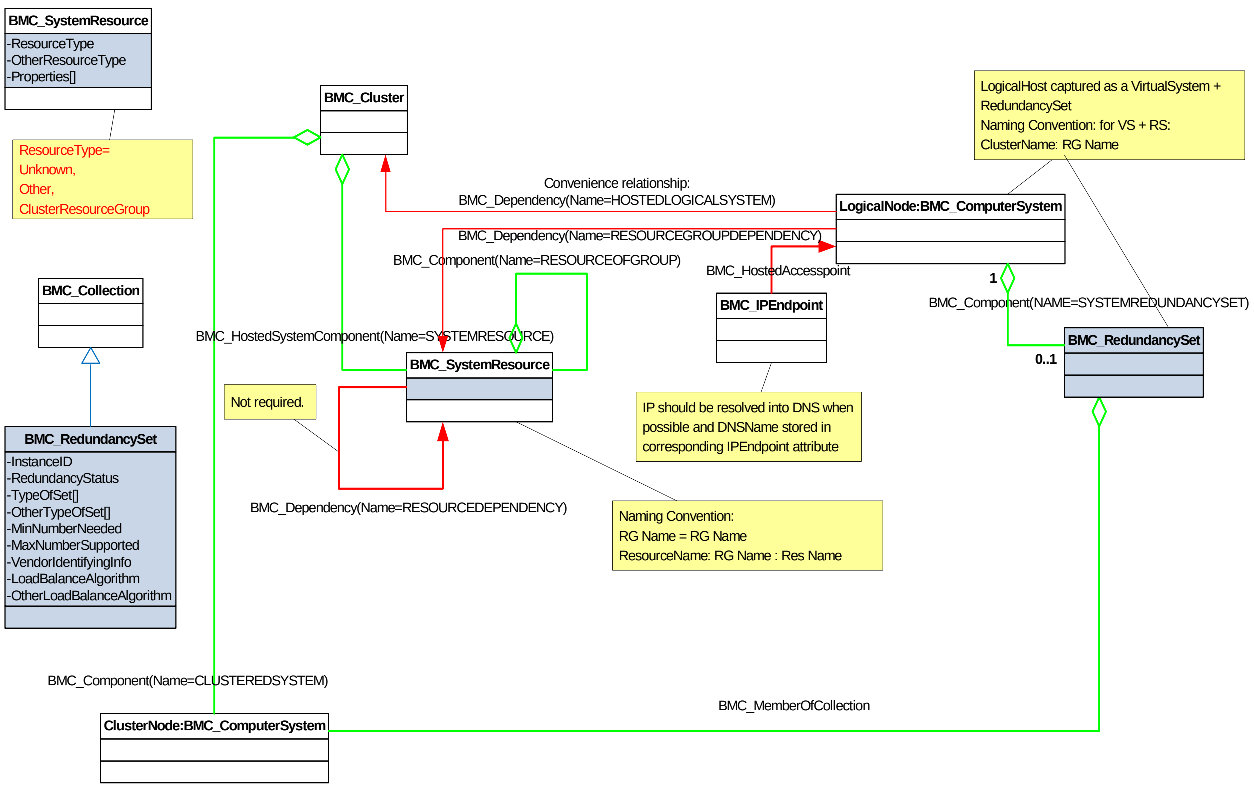
Modeling software and hardware clusters

Use the `BMC_Cluster` class to classify or update groups of software or hardware. Clusters are modeled using the `BMC_Cluster` class, which stores information about the cluster in relation to the `BMC_System` component.

Clusters help increase the performance of resources by storing groups of two or more computer systems or applications so that they operate together as a functional whole. Using clusters helps to improve and maintain the reliability, serviceability, and availability of your operating-system environment.

For example, an accounting department might create a cluster and link it with relationships to specific computer systems to obtain a complete picture of their business environment and assess the performance of computers individually and collectively. To do so, they could use the `BMC_Component` relationship to connect Computer A, Computer B, and Computer C to a performance measurement CI in the `BMC_Cluster` class.

Figure 1-10: Illustrative model of a clustered environment


BMC_SystemResource
-ResourceType
-OtherResourceType
-Properties[]
ResourceType= Unknown, Other, ClusterResourceGroup
BMC_Collection
BMC_RedundancySet
-InstanceID
-RedundancyStatus
-TypeOfSet[]
-OtherTypeOfSet[]
-MinNumberNeeded
-MaxNumberSupported
-VendorIdentifyingInfo
-LoadBalanceAlgorithm
-OtherLoadBalanceAlgorithm
BMC_Cluster
Convenience relationship: BMC_Dependency(Name=HOSTEDLOGICALSYSTEM)
BMC_Dependency(Name=RESOURCEGROUPDEPENDENCY)
BMC_Component(Name=RESOURCEOFGROUP)
BMC_HostedSystemComponent(Name=SYSTEMRESOURCE)
BMC_SystemResource
Not required.
BMC_Dependency(Name=RESOURCEDEPENDENCY)
LogicalHost captured as a VirtualSystem + RedundancySet
Naming Convention: for VS + RS:
ClusterName: RG Name
LogicalNode:BMC_ComputerSystem
BMC_HostedAccesspoint
BMC_IPEndpoint
IP should be resolved into DNS when possible and DNSName stored in corresponding IPEndpoint attribute
1
BMC_Component(NAME=SYSTEMREDUNDANCYSET)
0..1
BMC_RedundancySet
Naming Convention:
RG Name = RG Name
ResourceName: RG Name : Res Name
BMC_Component(Name=CLUSTEREDSYSTEM)
BMC_MemberOfCollection
ClusterNode:BMC_ComputerSystem

NOTE

`BMC_RedundancySet` is part of the BMC Topology Discovery CDM exension, and is not part of the Atrium Core CDM.

Chapter

2 Modeling applications

This chapter describes how to model software business entities, including applications, software servers, databases, and middleware.

The following topics are provided:

Application characteristics

Applications have characteristics that help you determine how to best use the CDM in your modeling strategy. Table 2-1 maps the characteristics of an application to the type of class you would use to model that application. Note that not all objects and relationships are required to model certain types of applications. For example, patch information may not be required in the case of software license management.

Table 2-1: Mapping application characteristics to a class

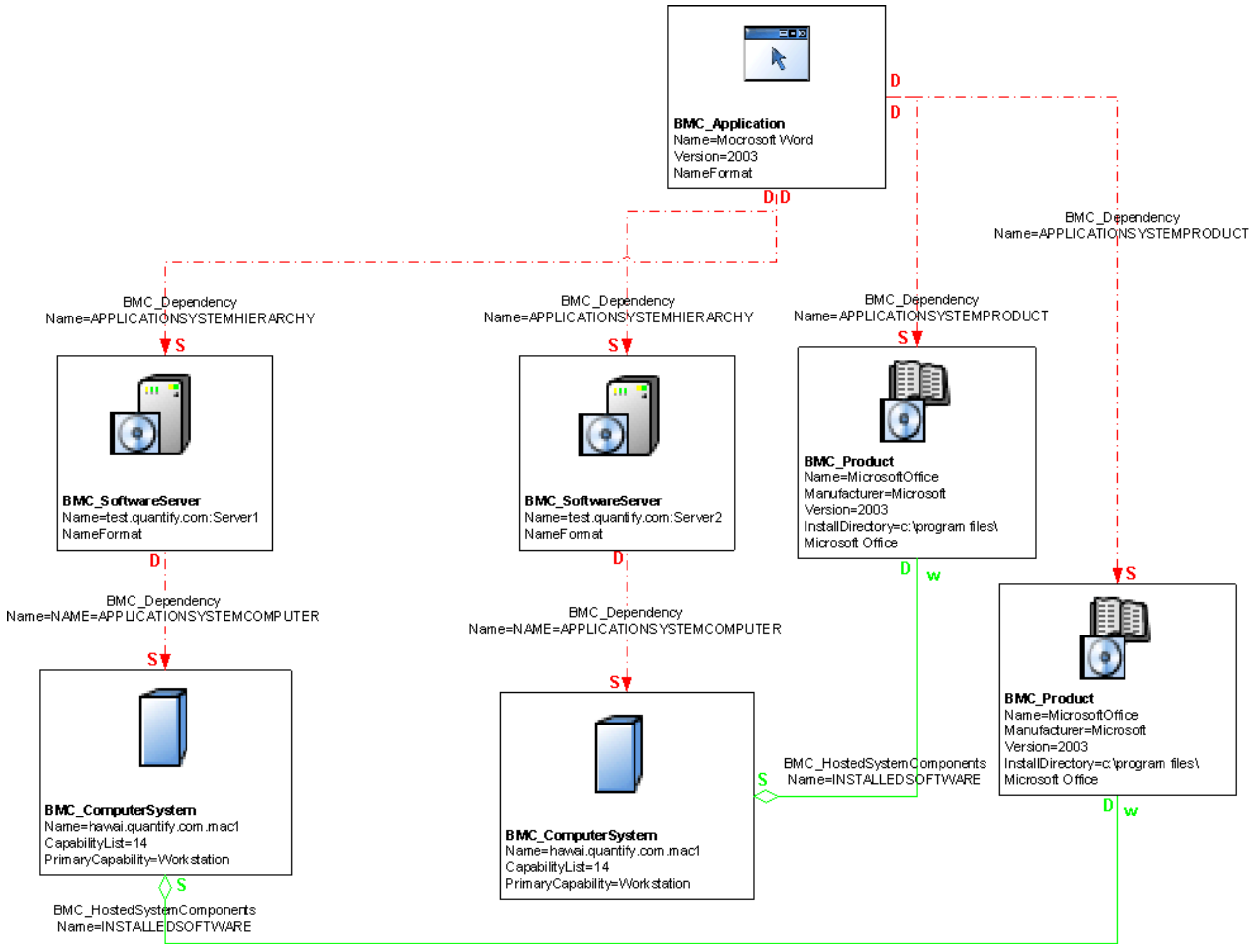
Characteristic	Description	Class

Runtime aspect	Running instances of applications and software servers	`BMC_SoftwareServer`
		`BMC_Application`
		`BMC_ApplicationInfrastructure`

Installation aspect	Identifies the product that is installed, its version, and any patch	`BMC_Product`
Service aspect	Business applications. (For business applications supporting a particular function such as payroll and trading, use the `BMC_BusinessService` class.)	`BMC_Application` `BMC_BusinessService`

Figure 2-1 on page 47 illustrates how the installed, runtime, and service aspects of an application relate to each other.

Figure 2-1: Illustrative representation of multiple application aspects


BMC_Application
Name=Mocrosoft Word
Version=2003
NameFormat
BMC_Dependency
Name=APPLICATIONSYSTEMHIERARCHY
BMC_Dependency
Name=APPLICATIONSYSTEMHIERARCHY
BMC_Dependency
Name=APPLICATIONSYSTEMPRODUCT
BMC_Dependency
Name=APPLICATIONSYSTEMPRODUCT
BMC_SoftwareServer
Name=test.quantify.com:Server1
NameFormat
BMC_SoftwareServer
Name=test.quantify.com:Server2
NameFormat
BMC_Product
Name=MicrosoftOffice
Manufacturer=Microsoft
Version=2003
InstallDirectory=c:\program files\ Microsoft Office
BMC_Dependency
Name=NAME=APPLICATIONSYSTEMCOMPUTER
BMC_Dependency
Name=NAME=APPLICATIONSYSTEMCOMPUTER
BMC_ComputerSystem
Name=hawai.quantify.com.mac1
CapabilityList=14
PrimaryCapability=Workstation
BMC_ComputerSystem
Name=hawai.quantify.com.mac1
CapabilityList=14
PrimaryCapability=Workstation
BMC_HostedSystemComponents
Name=INSTALLEDSOFTWARE
BMC_Product
Name=MicrosoftOffice
Manufacturer=Microsoft
Version=2003
InstallDirectory=c:\program files\ Microsoft Office
BMC_HostedSystemComponents
Name=INSTALLEDSOFTWARE
D
S
W

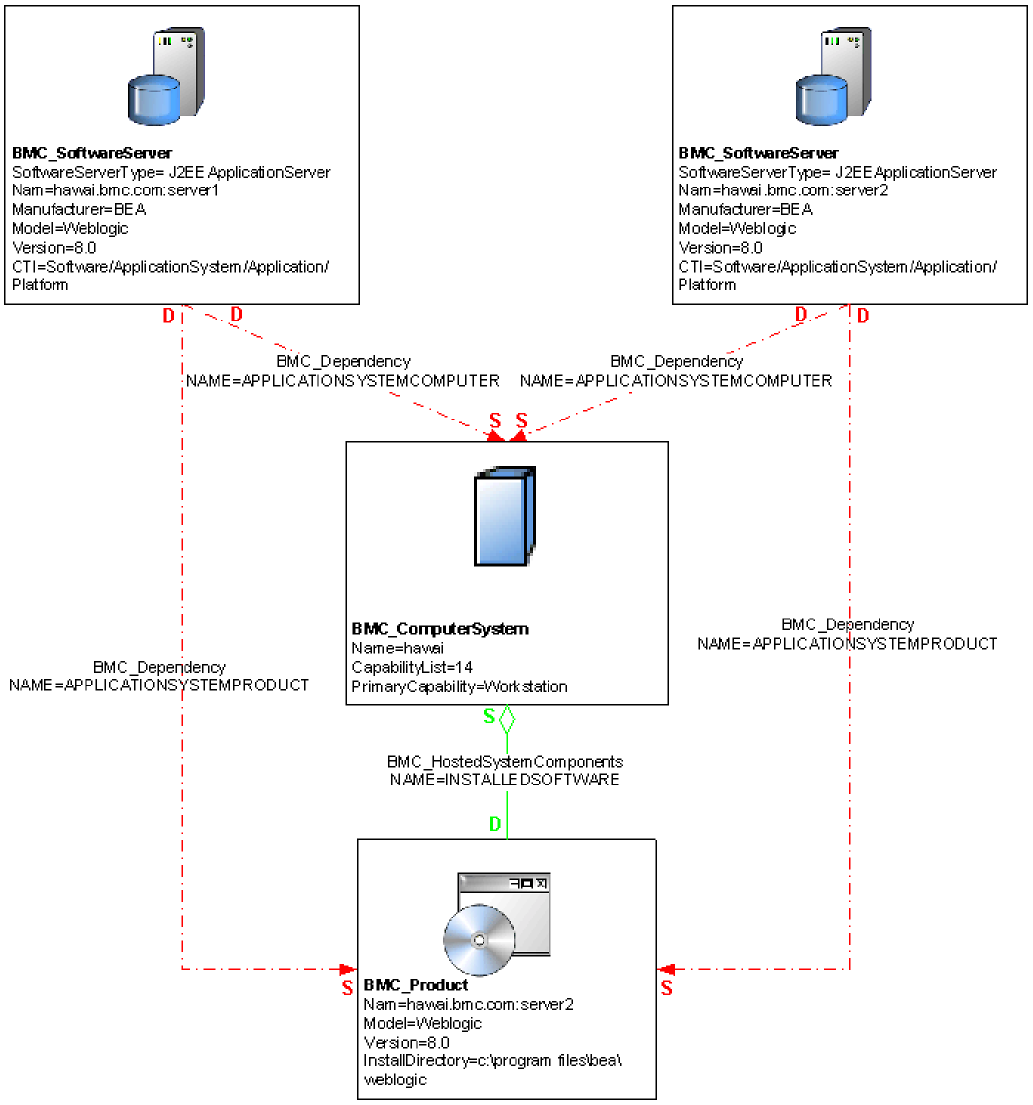
The BMC_SoftwareServer class represents the deployed, runtime aspects of applications; in other words, the instances of software actually running on a server. You instantiate this class to capture long-lived, server-type applications in your environment. When modeling applications, you must remember this distinction. To model static, installed components such as Microsoft Excel or Microsoft Word, create a BMC_Product instance.

You can also use the BMC_Product class to model noncommercial products, such as in-house software. One application can be installed once, yet have multiple instances running. For example, you can create a BMC_Product instance to represent the installed version of WebServer and create several BMC_SoftwareServer instances to represent actual instances of WebServer, one listening on port 80, another on port 8000, and a third on port 8080.

As another example, you would model Weblogic first by instantiating the BMC_Product class (to indicate where it is installed, the number of licenses, product name, and version). To add the runtime aspect, you would instantiate a BMC_SoftwareServer class.

Figure 2-2 illustrates an example of this model, where two instances of a Weblogic application server (server1 and server2) are actually instances of the same installed product.

Figure 2-2: Illustrative model of a Weblogic application


BMC_SoftwareServer
SoftwareServerType= J2EE ApplicationServer
Nam=hawai.bmc.com:server1
Manufacturer=BEA
Model=Weblogic
Version=8.0
CTI=Software/ApplicationSystem/Application/Platform
BMC_SoftwareServer
SoftwareServerType= J2EE ApplicationServer
Nam=hawai.bmc.com:server2
Manufacturer=BEA
Model=Weblogic
Version=8.0
CTI=Software/ApplicationSystem/Application/Platform
D
D
D
D
BMC_Dependency
NAME=APPLICATIONSYSTEMCOMPUTER
BMC_Dependency
NAME=APPLICATIONSYSTEMCOMPUTER
S
S
BMC_ComputerSystem
Name=hawai
CapabilityList=14
PrimaryCapability=Workstation
BMC_Dependency
NAME=APPLICATIONSYSTEMPRODUCT
BMC_Dependency
NAME=APPLICATIONSYSTEMPRODUCT
S
BMC_HostedSystemComponents
NAME=INSTALLEDSOFTWARE
D
S
S
BMC_Product
Nam=hawai.bmc.com:server2
Model=Weblogic
Version=8.0
InstallDirectory=c:\program files\bea\weblogic

IMPORTANT

Accounting for the runtime aspect of the application in this context is very important for understanding the impact of an application on a business service. You must consider capturing Weblogic patches (using the `BMC_Patch` class), because the patch will then be connected to the service through the installed product, runtime, applications and, ultimately, the service and its relationships. Consequently, an IT administrator responsible for updating patches on Weblogic would understand how the change relates to the business that Weblogic supports.

For complete descriptions of the classes described in this section for modeling applications, including examples of usage, see the BMC Atrium CMDB 7.6.00 Data Model Help. For more information about using the `BMC_Product` class to model components, see "Modeling software inventory and patches" on page 22.

Application infrastructure and hosting environment

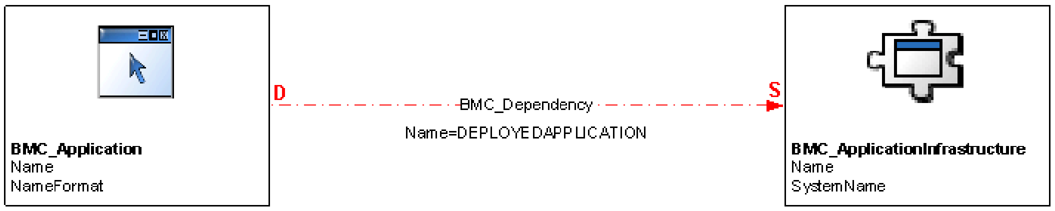
The `BMC_Application` class stores information about standalone applications, applications deployed on servers (such as SAP), and applications deployed on distributed systems (such as SAP).

The `BMC_ApplicationInfrastructure` class stores information about the framework that supports applications in a distributed or composite system. This class represents the platform to model your applications. For example, you would model SAP® as an instance of `BMC_ApplicationInfrastructure`. After an application is deployed in that platform, it can run on any application server in the SAP environment. An application can be hosted by different types of environments: an application server or application system, or a physical or virtual system. Both of these environments are detailed in the following sections.

Applications running on application servers or application systems

To model applications to run directly on top of an application server or application system, relate an instance of the `BMC_Application` class to a hosting `BMC_ApplicationInfrastructure` instance. In this model, the application has only one relationship: a dependency on the application infrastructure hosting the application. This dependency is modeled by a `BMC_Dependency` relationship, as illustrated in Figure 2-3. When using the relationship, set the `Name` value to DEPLOYEDAPPLICATION.

Figure 2-3: Illustrative model of applications running on application systems



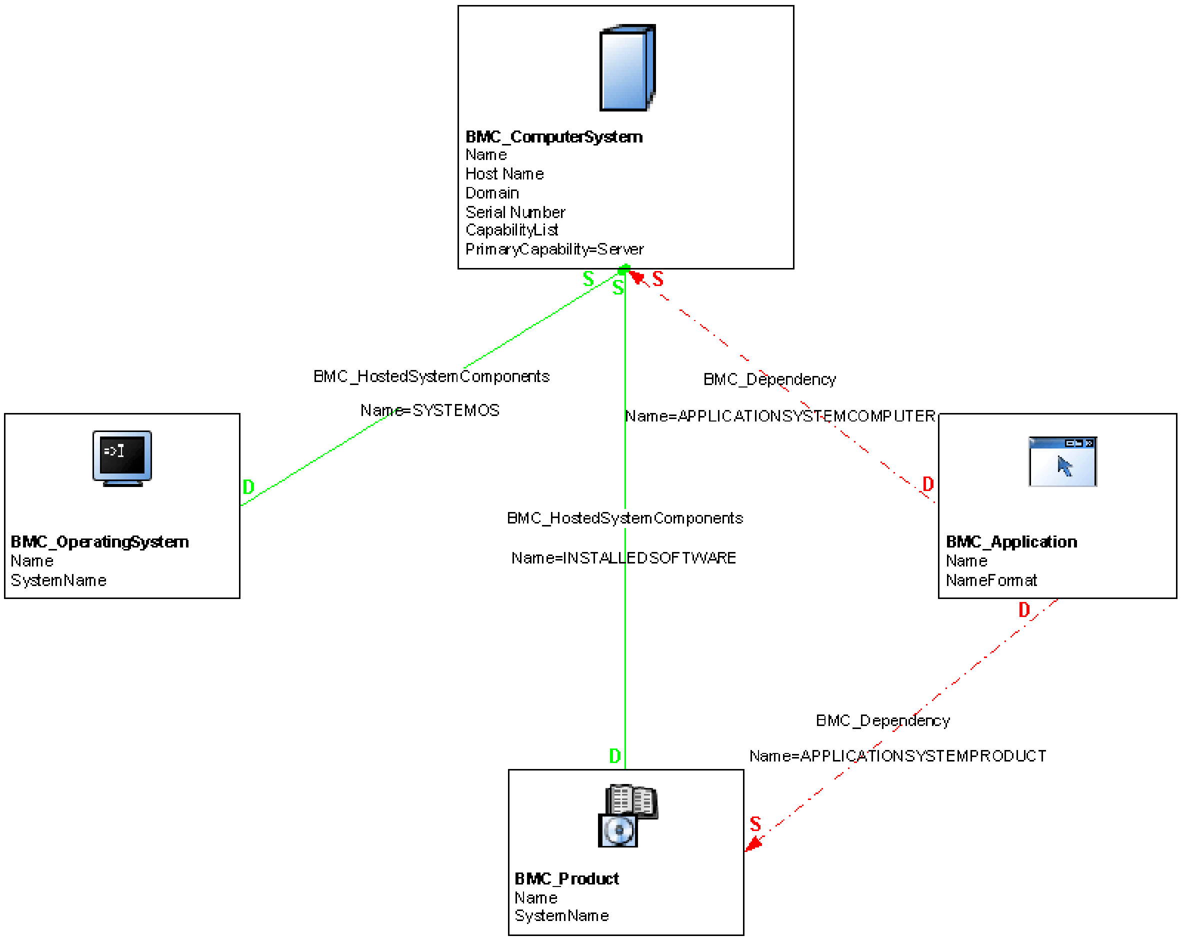
An application infrastructure cannot have any direct relationship to computers. Only applications and software servers have relationships to computers.

This model can also be applied to an application or set of applications that support or collaborate to provide a particular business function. For example, an Oracle® application infrastructure supports two applications, TimeCard and HR personal data, both stored in the BMC_Application class. The two classes relate to each other through the BMC_Dependency relationship, meaning that both the TimeCard and HR personal data applications are dependent on the supporting Oracle application infrastructure. To decompose the system into its functional components, relate an instance of this class to its component BMC_SoftwareServer instance with the BMC_Dependency class.

Applications running on computer systems

To model applications to run on computer systems (physical or virtual), relate an instance of the BMC_Application class to a hosting physical or virtual BMC_ComputerSystem instance. Figure 2-4 illustrates this model.

Figure 2-4: Illustrative model of applications running on computer systems



Relationships for applications

The relationships for modeling applications are described in Table 2-2.

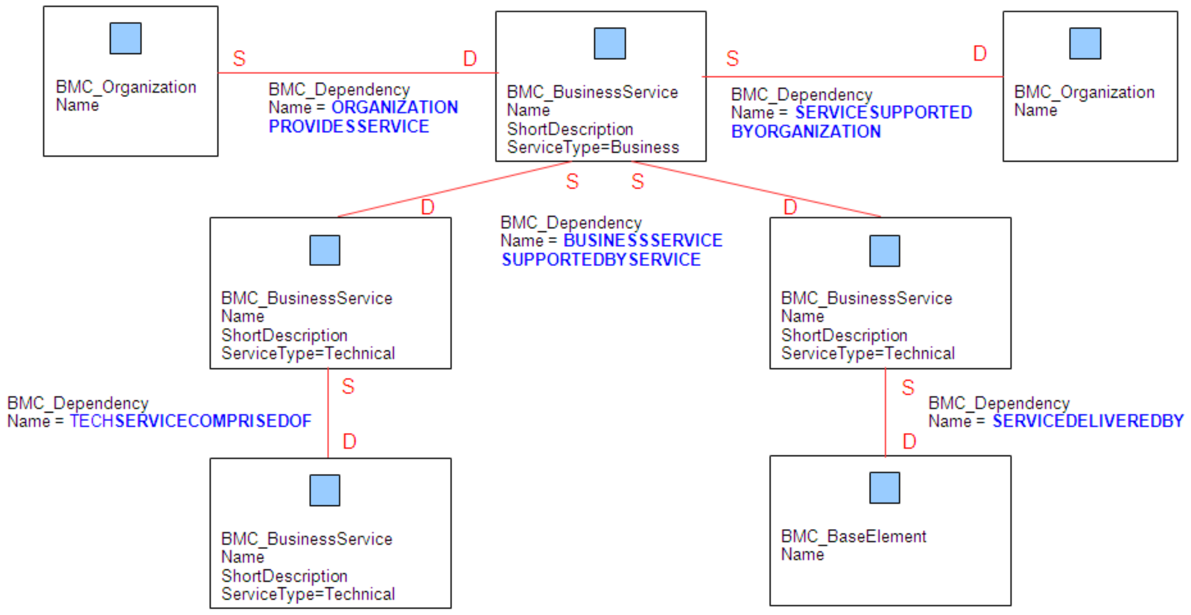
Table 2-2: Relationships for modeling applications

Relationship	Relationship class	Value of Name attribute
Application infrastructure hosting the application.	BMC_Dependency	DEPLOYEDAPPLICATION
System hosting the application (mandatory).	BMC_Dependency	APPLICATIONSYSTEMCOMPUTER
Operating system running the application (optional).	BMC_Dependency	APPLICATIONSYSTEMOS
Product representing the installed software of which this application is an instance (optional).	BMC_Dependency	APPLICATIONSYSTEMPRODUCT

Business applications and services

To model the business aspect of applications, use the `BMC_BusinessService` class. Business applications support a particular business function (such as payroll or trading) and are, generally, made up of a set of applications, servers, and databases that collaborate to provide a particular service.

Figure 2-5: Illustrative model of business services



NOTE

In this model, the BMC_BaseElement Name is typically an application or database.

Chapter

3 Modeling software servers

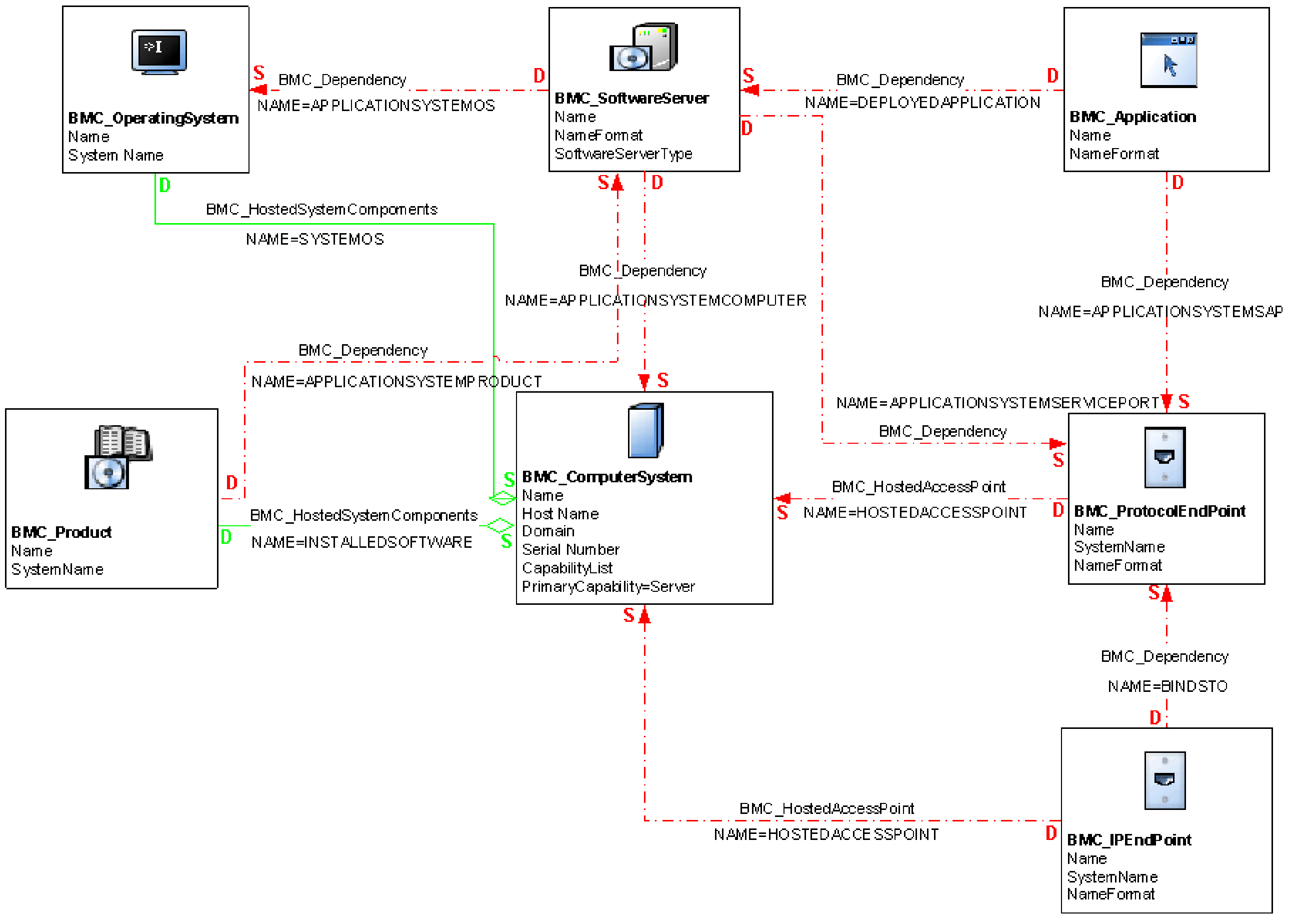
This chapter describes how to model software servers, such as database servers, web servers, DNS servers, mainframe servers, and directory servers.

The following topics are provided:

Software server characteristics

A software server is a system that provides services to client applications and other servers, runs on top of a physical or virtual system, and is modeled using the `BMC_SoftwareServer` class. Figure 3-1 illustrates a software server model.

Figure 3-1: Illustrative model of software servers



Logical identity of BMC_SoftwareServer

The `BMC_SoftwareServer` class stores information about a server that provides a single service to client applications or other systems. Database servers, web servers, DNS servers, mainframe servers, and directory servers can be represented by this class.

Table 3-1 details the key attributes used in the `BMC_SoftwareServer` class. When modeling software servers, you identify the unique server type by specifying its name in the `SoftwareServerType` attribute.

For example, for database servers, set the SoftwareServerType attribute to DatabaseServer.

Table 3-1: Key attributes for BMC_SoftwareServer

Attribute	Usage
Name	Identifies the name of the software server.
NameFormat	Sets the heuristic used to generate the Name value. Set this attribute to name the installation directory of the server.
SoftwareServerType	Specifies the type of software server (for example, DatabaseServer).

Additional attributes for BMC_SoftwareServer

Table 3-2 describes additional attributes for BMC_SoftwareServer.

Table 3-2: Additional attributes for BMC_SoftwareServer

Attribute	Description
ShortDescription	A caption of the software server.
TokenId	The unique identifier populated by BMC Discovery products and used by the Reconciliation Engine (of BMC Atrium CMDB) to identify instances.

Relationships for software servers

Table 3-3 describes the relationships for software servers.

Table 3-3: Relationships for software servers

Relationship	Relationship class	Value of Name attribute
Computer system hosting the software server (mandatory).	BMC_Dependency	APPLICATIONSYSTEMCOMPUTER
Communication endpoint (one relationship per endpoint) that the software server is listening on.	BMC_Dependency	APPLICATIONSYSTEMSERVICEENDPOINT
Operating system running the application. This dependency is actually on the OS that is running the server (as opposed to the computer).	BMC_Dependency	APPLICATIONSYSTEMOS
Installed software of which this software server is an instance (optional).	BMC_Dependency	APPLICATIONSYSTEMPRODUCT

Modeling database servers

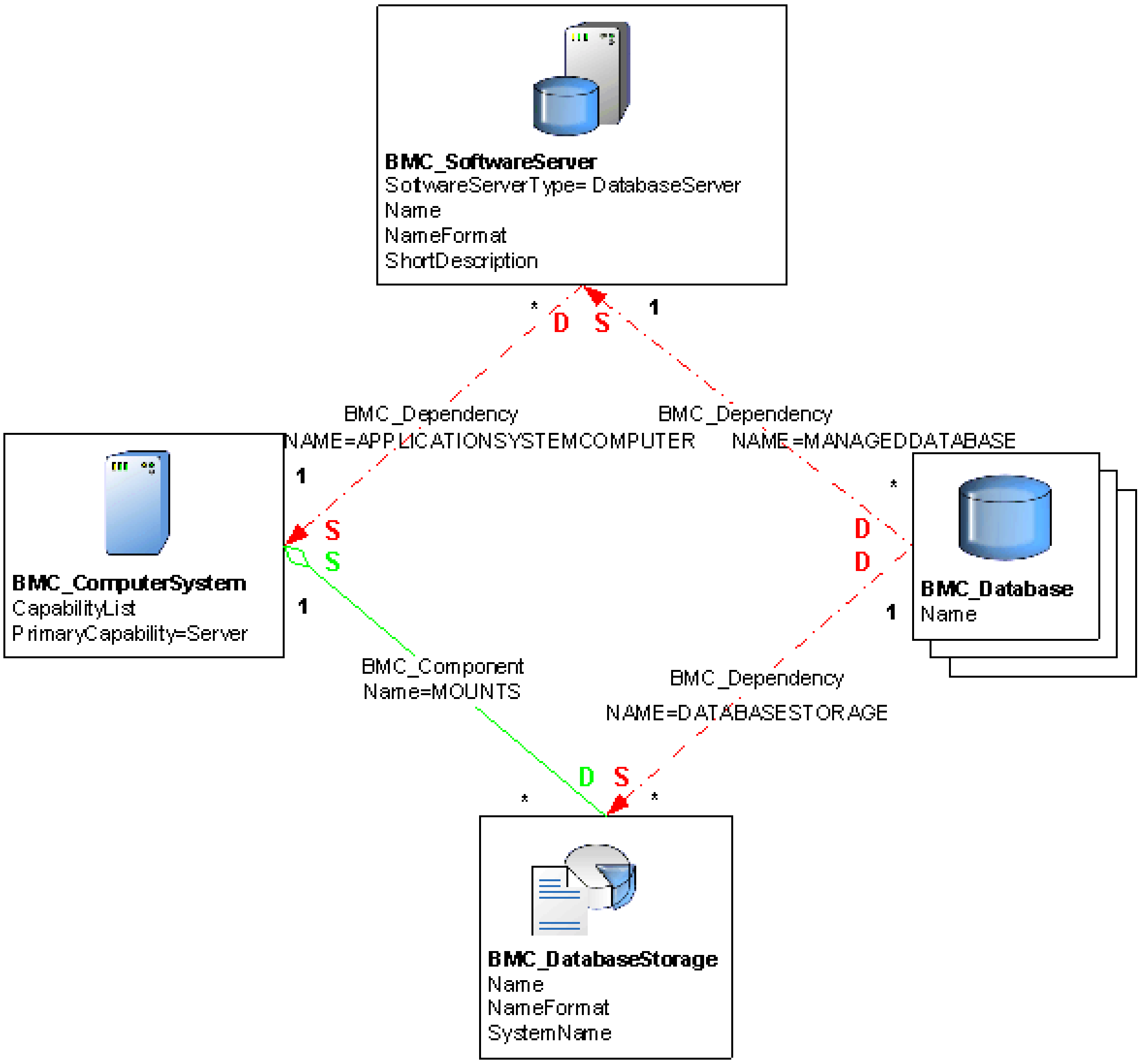
A database server is a form of software server that, like all software servers, must be uniquely named in the context of the CDM. A database server is modeled as an instance of the `BMC_SoftwareServer` class (derived from the `BMC_ApplicationSystem` class) and is identified by its `Name` attribute.

IMPORTANT

The key attribute for this class is `SoftwareServerType`, which must be set to DatabaseServer.

Figure 3-2 illustrates how to model a database server.

Figure 3-2: Illustrative model of a database server environment



Modeling databases

A database is a collection of interrelated data that is treated as a unit and that is organized into one or more schemas. Databases are dependent on software servers and, therefore, are dependent on database servers.

Logical identity of BMC_DataBase

A database is modeled as an instance of the `BMC_DataBase` class (derived from the `BMC_LogicalEntity` class) and is identified by its `Name` attribute.

Table 3-4 details the description and syntax for the `Name` and `NameFormat` attributes used in the `BMC_DataBase` class.

Table 3-4: Key attributes for BMC_DataBase

Attribute	Usage
`Name`	Identifies the database.
`NameFormat`	As multiple valid naming conventions can be used in specific contexts, the `NameFormat` attribute must be set with a value indicating the heuristic used to generate the `Name` value. For example, in some cases, a computer system will be identified by an external DNS Name (a name configured in a DNS Server). In other cases, a static IP Address will be used. In any case, the values for `NameFormat` should be: ■ HostName.IP: The name must be a valid IP Address, decimal bytes delimited with dots ('.') ■ HostName.DNS: The name must a fully qualified host name, a host name and a domain name delimited with dots (the domain name can also consist of multiple components delimited with dots).

The `BMC_DataBase` class defines the properties that are common across database models and vendor implementations for the database entity that is represented by the unit of interrelated data. Create an instance of this class for each managed database. You can use this class to specify the software that belongs to the database, perform system-wide database management operations (such as stopping all the databases that were created by the system for maintenance purposes), or view runtime statistics for the database.

To represent database storage areas, use the `BMC_DataBaseStorage` class. The key to a `BMC_DataBase` instance in an enterprise environment is its `Name` attribute. For more information about database storage, see “Modeling database storage entities” on page 60.

Additional attributes for BMC_DataBase

Although databases are primarily defined by the Name attribute, Table 3-5 provide additional information about an instance of BMC_DataBase:

Table 3-5: Additional information for BMC_DataBase

Attribute	Description
ShortDescription	A short description of the database.
TokenId	A unique identifier populated by BMC Discovery products and used by the Reconciliation Engine (of BMC Atrium CMDB) to identify instances.

Modeling an Oracle Listener

The Oracle Listener manages network communications for one or more database instances. An Oracle Listener is modeled as an instance of the BMC_SoftwareServer class (derived from BMC_ApplicationSystem) and is identified by both its Name attribute (set to Oracle Listener) and SoftwareServerType attribute (set to Other).

Table 3-6 details the attributes used to model Oracle Listeners.

Table 3-6: Attributes used to model Oracle Listeners

Attribute	Usage
Name	Identifies the Oracle Listener.
NameFormat	Defines the heuristic used to generate the Name value.
SoftwareServerType	Defines the type of server. This value must be set to Other.
ShortDescription	A caption of the database.
TokenId	Specifies the unique identifier populated by BMC Discovery products and used by the Reconciliation Engine to identify instances.

Relationships for database servers and databases

The BMC_Dependency class is a generic association used to establish dependency relationships between instances in the BMC Atrium CMDB. This association allows you to establish dependency relationships between endpoints, including the roles of the endpoints.

Table 3-7 describes the relationships for database servers and databases.

Table 3-7: Relationships for database servers and databases

Relationship	Relationship class	Value of Name attribute
Computer system (source) and database server (destination)	`BMC_Dependency`	APPLICATIONSYSTEMCOMPUTER
Computer system (source) and Oracle Listener (destination)	`BMC_Dependency`	APPLICATIONSYSTEMCOMPUTER
Oracle Listener (source) and database server (destination)	`BMC_Dependency`	DEPENDENCY
Installed software of which this software server is an instance (optional).	`BMC_Dependency`	APPLICATIONSYSTEMPRODUCT
Database server (source) and database (destination)	`BMC_Dependency`	MANAGEDDATABASE
Computer system (source) and the file system (destination)	`BMC_HostedSystemComponents`	HOSTEDSYSTEMCOMPONENT
Database storage (source) and database (destination)	`BMC_Component`	DATABASEDATASTORAGE

Modeling database storage entities

Database storage entities are an extension of file system CIs in a database environment and are modeled using the BMC_DataBaseStorage class.

Logical identity of BMC_DataBaseStorage

The BMC_DataBaseStorage class stores information about a collection of logical storage areas that hold and retain data. You model a database storage CI as an instance of the BMC_DataBaseStorage class (derived from the BMC_FileSystem class) and identify the instance by its Name and SystemName attributes. The BMC_DataBaseStorage class extends a file system CI and uses its inherited associations to represent the internal structure of the database.

Table 3-8 details the attributes used to model database storage CIs.

Table 3-8: Key attributes for model database storage CIs

Attribute	Usage
Name	Identifies the storage area (tablespace) name.
NameFormat	Specifies the heuristic used to generate the Name value. Set it with the storage name used to generate the Name value.
SystemName	Specifies the name of the parent BMC_Database CI. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes for BMC_DataBaseStorage

Table 3-9 describes attributes that provide additional information about an instance of BMC_DataBaseStorage.

Table 3-9: Additional information for BMC_DataBaseStorage

Attribute	Description
IsSystemArea	The owner of the storage area.
ShortDescription	A caption of the database.
TokenId	A unique identifier populated by BMC Discovery products and used by the Reconciliation Engine to identify instances

Chapter
4
Modeling Microsoft Exchange

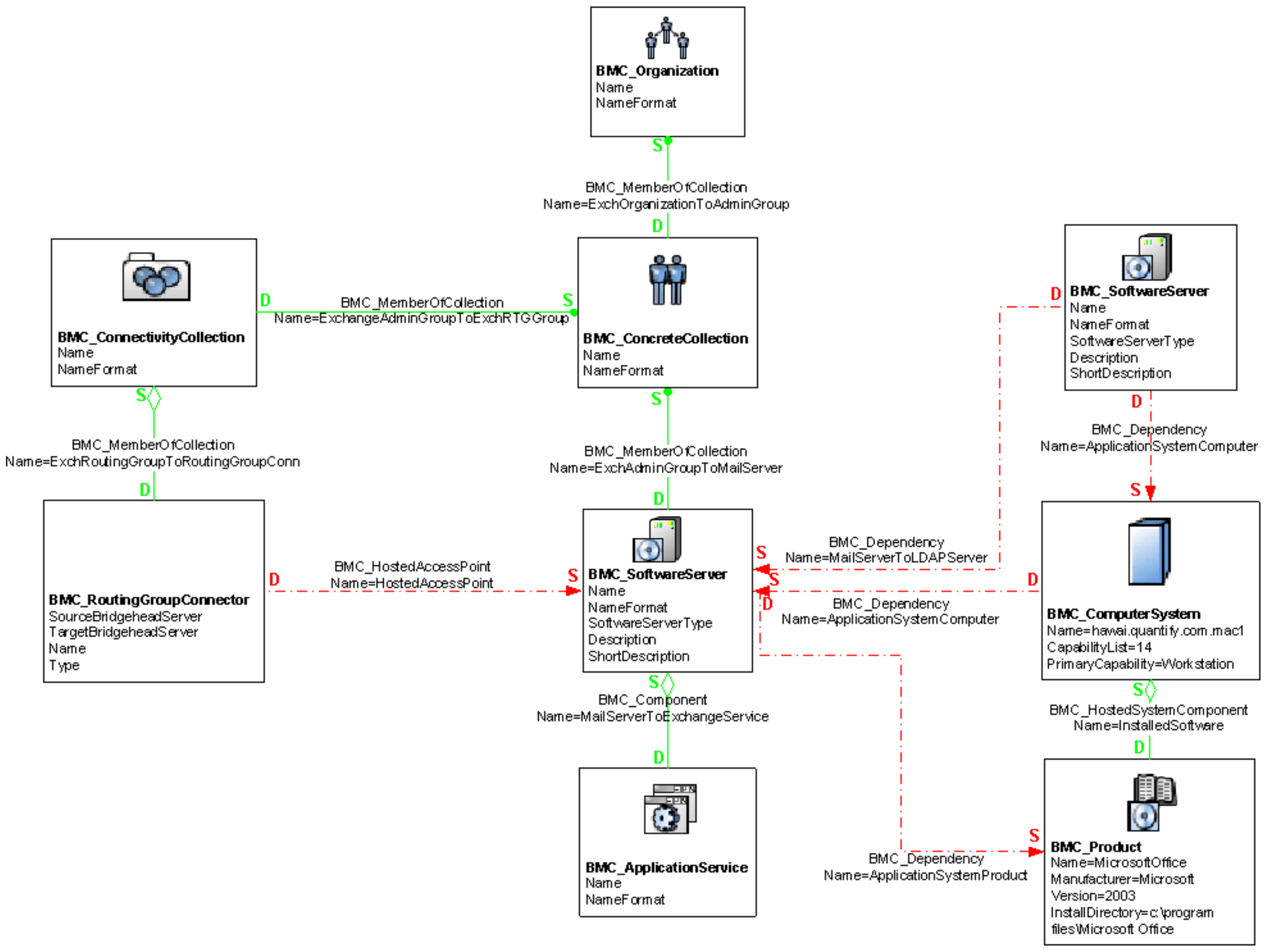
This chapter describes how to use the CDM to model Microsoft Exchange business entities, such as organizations, servers, services, software, routing groups, and connectors.

The following topics are provided:

Microsoft Exchange business entity characteristics

Exchange objects are contained in an organization, as illustrated in Figure 4-1.

Figure 4-1: Illustrative model of an Exchange environment



In the diagram, an Exchange server organization is represented by the container `BMC_Organization`, which includes the objects defined in Table 4-1:

Table 4-1: Exchange environment objects (Sheet 1 of 2)

Object	Function and class used to model
Exchange server organization administrative groups	Collect Exchange servers that are grouped to manage permissions. You model these groups using the `BMC_ConcreteCollection` class.
Exchange servers	Provide a single service to client applications or other systems. You model Exchange servers using the `BMC_Software` class.

Table 4-1: Exchange environment objects (Sheet 2 of 2)

Object	Function and class used to model
Exchange server services	Represent MS Exchange SA Service, MS Exchange MTA Service and MS Exchange IS Service. You model services using the `BMC_BusinessService` class.
Exchange server software	Represents the installed version of Exchange server. You model Exchange server software using the `BMC_Product` class.
Routing groups	Logically organize Exchange servers. You model routing groups using the `BMC_ConnectivityCollection` class.
Routing group connectors	Communicate between routing groups. You model connectors using the `BMC_RoutingGroupConnector` class.

Modeling an Exchange server organization

The `BMC_Organization` class represents the Exchange server organization and is the container that includes all administrative groups deployed within it.

Logical identity of BMC_Organization

An Exchange server organization is modeled as an instance of `BMC_Organization` and is identified by its `Name` attribute. The `NameFormat` attribute should be set to the name of the organization.

Table 4-2: Key attributes for BMC_Organization

Attribute	Usage
`Name`	Identifies the organization.
`NameFormat`	Specifies the name of the organization in the format MSEXCHANGE:ObjectGUID. ObjectGUID is a unique identifier. As multiple valid Naming Conventions may exist and be used according to specific contexts, the `NameFormat` attribute must be set with a value indicating the heuristic used to generate the `Name` value. For example, in some cases, you identify a computer system by an external DNS name (a name configured in a DNS Server). In other cases, you will use a static IP address.

Additional attributes for BMC_Organization

Table 4-3 describes additional attributes that you can use to model an Exchange organization.

Table 4-3: Additional attributes for BMC_Organization

Attribute	Description
`Description`	The description for the organization.
`ShortDescription`	A caption for the organization.

Relationships for an Exchange organization

Table 4-4 describes relationships for an Exchange organization.

Table 4-4: Relationships for an Exchange organization

Relationship	Relationship class	Value of Name attribute
Exchange organization and administrative groups	`BMC_MemberOfCollection`	EXCHORGANIZATIONTOADMINGROUP
Administrative groups and Exchange server	`BMC_MemberOfCollection`	EXCHADMINGROUPTOMAILSERVER
Administrative groups and routing groups	`BMC_MemberOfCollection`	EXCHADMINGROUPTOEXCHRTGGROUP

Modeling Exchange server organization administrative groups

An Exchange server organization administrative group is a collection of Exchange servers that is grouped together for the purposes of managing permissions.

Logical identity of BMC_ConcreteCollection

An Exchange server organization administrative group is modeled as an instance of `BMC_Collection`. This class stores a generic and instantiable collection, such as a pool of hosts available for running jobs. It is defined as a subclass of `BMC_Collection` and is identified by its `Name` attribute. The `NameFormat` attribute is set to HostName.OrganizationName:Name of the administrative group.

Table 4-5: Key attributes for BMC_ConcreteCollection

Attribute	Usage
`Name`	Identifies the organization administrative group.
`NameFormat`	Specifies the heuristic used to generate the `Name` value. Specify the name of the organization administrative group in the format MSEXCHANGE:ObjectGUID. ObjectGUID is a unique identifier.

Additional attributes for BMC_ConcreteCollection

Table 4-6 describes additional attributes that you can use to model an Exchange organization administrative group.

Table 4-6: Additional attributes for BMC_ConcreteCollection

Attribute	Description
`Description`	The description for the organization administrative group.
`ShortDescription`	A caption for the organization administrative group.

Modeling Exchange servers

An Exchange server is an extension to the `BMC_SoftwareServer` class that stores information about a server and provides a single service to client applications or other systems.

Logical identity of BMC_SoftwareServer

An Exchange server is modeled as an instance of the class `BMC_SoftwareServer` (derived from `BMC_ApplicationSystem`) and is identified by its `Name` attribute. The `NameFormat` attribute is set to HostName.DNS:Manufacturer.

Additional attributes for BMC_SoftwareServer

Table 4-7 describes additional attributes that you can use for modeling Exchange servers.

Table 4-7: Additional attributes for BMC_SoftwareServer

Attribute	Description
`Description`	The description for the Exchange server.
`ManufacturerName`	The manufacturer name for the Exchange server. Always set this attribute to Microsoft Corporation.
`Model`	The model name of the Exchange server. Always set this attribute to Microsoft Exchange Server.
`ShortDescription`	A caption for the Exchange server.
`SoftwareServerType`	The type of Exchange server. Set this attribute to MailServer for Microsoft Exchange servers or LDAPServer for an LDAP server.

Relationships for an Exchange server

Table 4-8 describes relationships for an Exchange server.

Table 4-8: Relationships for an Exchange server

Relationship	Relationship class	Value of Name attribute
Exchange server and its services whose class is `BMC_BusinessService`	`BMC_Component`	MAILSERVERTOEXCHANGESERVICE
Exchange server and connector	`BMC_HostedAccessPoint`	HOSTEDACCESSPOINT
Exchange server and product	`BMC_HostedSystemComponents`	INSTALLEDSOFTWARE
Mail server and LDAP server	`BMC_Dependency`	MAILSERVERTOLDAPSERVER
Computer system to mail server	`BMC_Dependency`	APPLICATIONSYSTEMCOMPUTER
Computer system and LDAP server	`BMC_Dependency`	APPLICATIONSYSTEMCOMPUTER
Mail server and product	`BMC_Dependency`	APPLICATIONSYSTEMPRODUCT

Modeling Exchange server services

Services describe what is delivered by an organization, process or system and cannot be performed or executed. Exchange server services run on Exchange servers and include message transfer agents, information stores, and system attendants. Application services can be used to capture modules of an application, or services that are deployed on a server.

Logical identity of BMC_BusinessService

Exchange server services are captured as an instance of `BMC_BusinessService` and are identified by the `ServiceType` attribute. The `BMC_BusinessService` class stores information about business or technical services. For MS Exchange, this class is used to represent MS Exchange SA Service, MS Exchange MTA Service and MS Exchange IS Service.

Table 4-9: Key attributes for BMC_BusinessService

Attribute	Usage
`ServiceType`	Identifies the type of service that the instance of `BMC_BusinessService` represents. Supported values are Unknown (10), BusinessService (20), and TechnicalService (30). The default value is Unknown.
`NameFormat`	Specifies the heuristic used to generate the `Name` value. Specify the name of the service in the format MSEXCHANGE:ObjectGUID. ObjectGUID is a unique identifier.

Modeling Exchange server software

This section details how to model installed versions of Microsoft Exchange software.

Logical identify of BMC_Product

Exchange server software is modeled using the `BMC_Product` class, and is identified by the `Name` attribute. Set the `NameFormat` attribute to InstallDirectory:Model:Version.

Additional attributes for BMC_Product

The following table describes additional attributes that you can use for modeling Exchange server installations.

Table 4-10: Additional attributes for BMC_Product

Attribute	Description
`Description`	The description for the product.
`InstallLocation`	The location where the Exchange software is installed.
`ManufacturerName`	The manufacturer name for the Exchange Server installation. Always set this attribute to Microsoft Corporation.
`Model`	The model name of the Exchange server installation. Always set this attribute to Microsoft Exchange Server.
`ShortDescription`	A caption for the product.
`VersionNumber`	The version number of the Exchange server.

Modeling routing groups

Routing groups are used to logically organize Exchange servers. They represent a set of protocol endpoints of the same type that can communicate with each other. They can also group related systems, users, or other base elements.

Logical identity of BMC_ConnectivityCollection

A routing group is modeled as an instance of `BMC_ConnectivityCollection` (derived from `BMC_Collection`) and is identified by the `Name` attribute. The `NameFormat` attribute is set to MSEXCHANGE:ObjectUID.

Additional attributes for BMC_ConnectivityCollection

The following table describes additional attributes for BMC_ConnectivityCollection.

Table 4-11: Additional attributes for BMC_ConnectivityCollection

Attribute	Description
Description	The description for the routing group.
ShortDescription	A caption for the routing group.

Modeling routing group connectors

Routing group connectors are used for communication between routing groups. Each routing group has a BridgeHead Exchange server that communicates with other routing groups through the connectors.

Logical identity BMC_RoutingGroupConnector

You model a routing group connector as an instance of the class BMC_RoutingGroupConnector (derived from BMC_AccessPoint) and identify the class by its Name attribute.

The name of the connector is set with the NameFormat attribute as SystemName:Name.

Additional attributes for BMC_RoutingGroupConnector

The following table describes additional attributes of BMC_RoutingGroupConnector for modeling connectors.

Table 4-12: Additional attributes for BMC_RoutingGroupConnector

Attribute	Description
SourceBridgeHeadServer	The source BridgeHead server.
TargetBridgeHeadServer	The target BridgeHead server.

Relationships for routing groups and connectors

Routing groups and connectors have one relationship: a BMC_MemberOfCollection relationship with the Name attribute set to EXCHROUTINGGROUPTOROUTINGGROUPCONN.

Chapter

5 Modeling storage entities

This chapter details how to model storage entities and devices.

The following topics are provided:

Characteristics of storage entities and devices

Storage entities and devices might include business CIs such as tape drives, disk drives, and storage subsystems. The following sections detail how to model these types of entities and devices.

Modeling tape drives

A tape drive is modeled as an instance of `BMC_TapeDrive` (derived from `BMC_Media`).

Logical identity of BMC_TapeDrive

Table 5-1 describes key attributes of a tape drive.

Table 5-1: Key attributes of a tape drive

Attribute	Usage
`Name`	Identifies the instance. The unique name is not necessarily human-readable.
`NameFormat`	Specifies the heuristic used to generate the `Name` value.
`SystemName`	Specifies the name of the system in which the component resides. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes for BMC_TapeDrive

Table 5-2 describes additional attributes of a tape drive.

Table 5-2: Additional attributes of a tape drive

Attribute	Description
`ComponentAliases`	The array of strings indicating the unique identifier of the SIM.
`Description`	The description of the instance.
`ManufacturerName`	The name of the vendor.
`MediaType` (not inherited)	The type of media. Its value is always Removable Media.
`Model`	The tape drive model.
`ShortDescription`	A caption of the instance.
`TokenID`	A unique identifier populated by BMC Discovery products and used by the Reconciliation Engine (of BMC Atrium CMDB) to identify instances.

Tape drive instance

Table 5-3 illustrates an example instance.

Table 5-3: Example of a tape drive instance

Attribute	Value
`ComponentAlias`	003590.B1A.IBM.13.000000044832.0080
`Description`	003590.B1A.IBM.13.000000044832.0080
`ManufacturerName`	IBM®
`MediaType`	Removable Media
`Model`	3590-1
`Name`	003590.B1A.IBM.13.000000044832.0080
`NameFormat`	Mainframe
`ShortDescription`	003590.B1A.IBM.13.000000044832.0080
`SystemName`	003590.B1A.IBM.13.000000044832
`TokenID`	003590.B1A.IBM.13.000000044832.0080

Modeling a DASD

A DASD is modeled as an instance of `BMC_DiskDrive` (derived from `BMC_Media`).

Logical identity of BMC_DiskDrive

Table 5-4 describes key attributes of a DASD.

Table 5-4: Key attributes of DASD

Attribute	Usage
`Name`	Identifies the instance of the disk drive. Use this attribute to specify the unique name of the instance; not necessarily human-readable.
`NameFormat`	Specifies the heuristic used to generate the `Name` value.
`SystemName`	Specifies the name of the system in which the component resides. This attribute is automatically populated from the related CI when a weak relationship is created between the computer system and the operating system.

Additional attributes for BMC_DiskDrive

Table 5-5 describes additional attributes of a DASD.

Table 5-5: Additional attributes of DASD

Attribute	Description
ComponentAliases	The array of strings indicating the unique identifier of the SIM.
Description	The description of the instance.
ManufacturerName	The name of the mainframe vendor.
MediaType (inherited from BMC_Media)	The type of media. Its value is always Fixed Hard Disk.
Model	The DASD model.
SerialNumber	The manufacturer-allocated number used to identify the instance.
ShortDescription	A caption of the instance.
TokenID	A unique identifier populated by the BMC Discovery products and used by the Reconciliation Engine (of BMC Atrium CMDB) to identify instances.

DASD instance

Table 5-6 illustrates an example DASD instance.

Table 5-6: Attributes of a DASD instance

Attribute	Value
Name	002105.000.IBM.13.000000025559.0B46
NameFormat	Mainframe
SystemName	002105.000.IBM.13.000000025559
ComponentAlias	002105.000.IBM.13.000000025559.0B46
Description	002105.000.IBM.13.000000025559.0B46
ManufacturerName	IBM
MediaType	Fixed Hard Disk
Model	3390
SerialNumber	ADR071
ShortDescription	ADR071
TokenID	002105.000.IBM.13.000000025559.0B46

Modeling storage subsystems

A storage subsystem is modeled as an instance of BMC_StorageSubsystem (derived from BMC_ComputerSystem).

Logical identity of BMC_StorageSubsystem

Table 5-7 describes key attributes of a storage subsystem.

Table 5-7: Key attributes of a storage subsystem

Attribute	Usage
`Name`	Identifies the storage subsystem. Use this attribute to specify the unique name of the instance; not necessarily human-readable.
`NameFormat`	Specifies the heuristic used to generate the `Name` value.

Additional attributes of BMC_StorageSubsystem

Table 5-8 describes additional attributes of a storage subsystem.

Table 5-8: Additional attributes of a storage subsystem

Attribute	Description
`ComponentAliases`	The array of strings indicating the unique identifier of the SIM.
`Description`	The description of the instance.
`ManufacturerName`	The name of the vendor.
`Model`	The mainframe model.
`ShortDescription`	A caption of the instance.
`TokenID`	A unique identifier populated by BMC Discovery products and used by the Reconciliation Engine (of BMC Atrium CMDB) to identify instances.

Example subsystem instances

These tables illustrate instances of storage subsystems.

Tape drive subsystem instance

Attribute	Value
`CapabilityList`	StorageSubsystem
`ComponentAlias`	003590.B1A.IBM.13.000000044832
`Description`	003590.B1A.IBM.13.000000044832
`ManufacturerName`	IBM
`Model`	TapeSubsystem
`Name`	003590.B1A.IBM.13.000000044832
`NameFormat`	Mainframe
`PrimaryCapability`	StorageSubsystem
`ShortDescription`	003590.B1A.IBM.13.000000044832
`TokenID`	003590.B1A.IBM.13.000000044832

DASD subsystem instance

Attribute	Value
CapabilityList	StorageSubsystem
ComponentAlias	002105.000.IBM.13.000000025559
Description	002105.000.IBM.13.000000025559
ManufacturerName	IBM
Model	DASDSubsystem
Name	002105.000.IBM.13.000000025559
NameFormat	Mainframe
PrimaryCapability	StorageSubsystem
ShortDescription	002105.000.IBM.13.000000025559
TokenID	002105.000.IBM.13.000000025559

Relationships for a storage subsystem

Table 5-9 describes the relationships for a storage subsystem.

Table 5-9: Relationships for a storage subsystem

Relationship	CI class	Relationship class	Value of Name attribute
Storage subsystem and an operating system	BMC_OperatingSystem	BMC_Dependency	STORAGESUBSYSTEMOS
Storage subsystem and a DASD	BMC_DiskDrive	BMC_HostedSystemComponents	STORAGESUBSYSTEMDASD
Storage subsystem and a tape drive	BMC_TapeDrive	BMC_HostedSystemComponents	STORAGESUBSYSTEMTAPE

Chapter

6 Modeling network topology

This chapter details how to model network topology, including components on a subnet, network interfaces, and LAN and WAN networks.

The following topics are provided:

Network topology characteristics

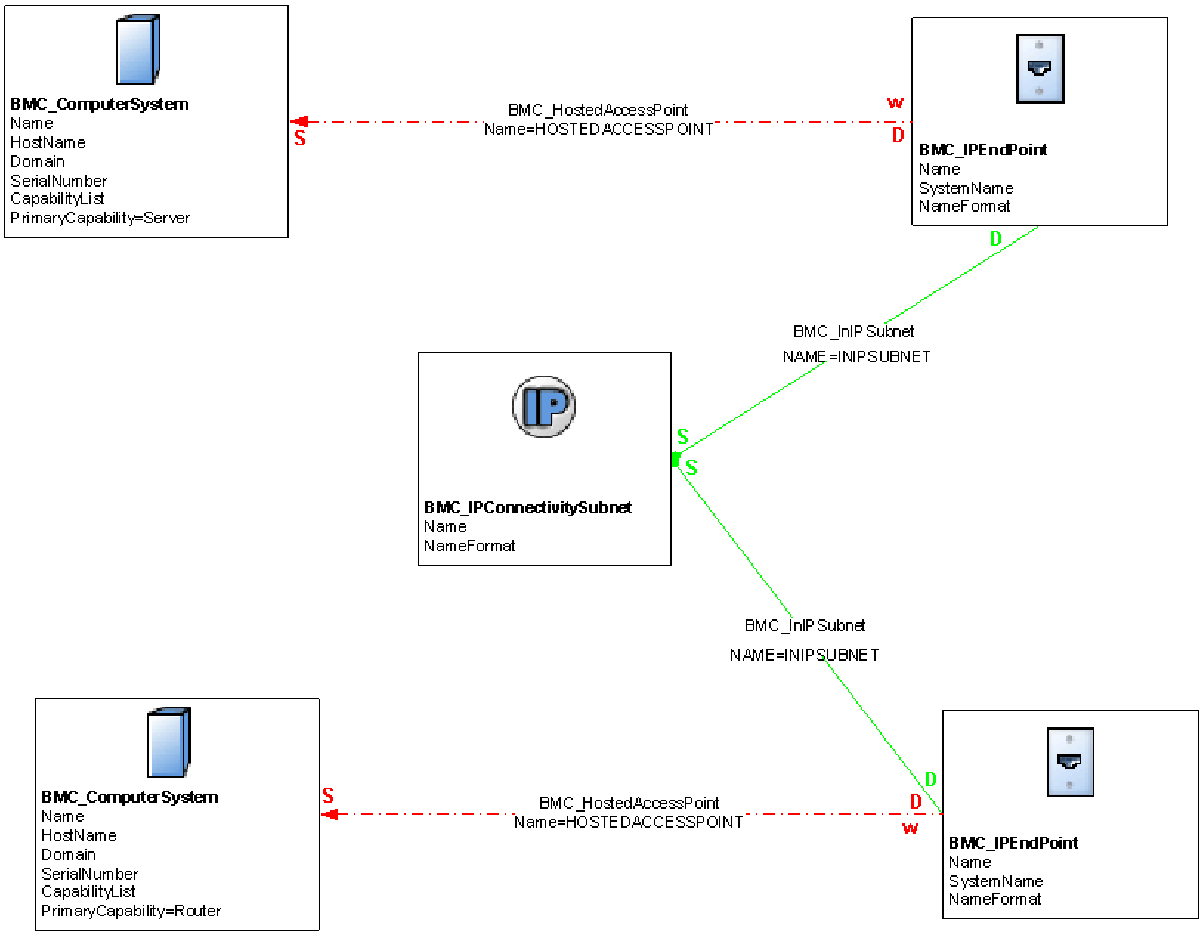
Topologies are based on the `BMC_ConnectivityCollection` class, which are collections of `BMC_ProtocolEndpoint` (communication points from which data may be sent or received) of the same type and which can communicate with each other. Logical groupings of these connectivity collections enable users to define the scope of LAN and WAN networks.

Modeling L3 topology and IP connectivity

A `BMC_IPConnectivitySubnet` instance represents a group of related `BMC_IPEndpoint` instances that can communicate with each other as members of a subnet and describes the characteristics of the subnet.

Figure 6-1 illustrates a server and a router that belong to the same subnet.

Figure 6-1: Illustrative model of components on a subnet



Logical identity of BMC_IPConnectivitySubnet

Table 6-1 describes key attributes of BMC_IPConnectivitySubnet.

Table 6-1: Key attributes for BMC_IPConnectivitySubnet

Attribute	Usage
Name	Identifies the IP address of the entire subnet, formatted according to the appropriate convention as defined in the AddressType attribute. When AddressType is 1 (IPV4), the Name must be built by concatenating the SubnetNumber and SubnetMask separated by a /.
NameFormat	Specify the heuristic used to generate the Name value, which must be set to IP.

Additional attributes for BMC_IPConnectivitySubnet

Table 6-2 describes attributes that provide additional information about BMC_IPConnectivitySubnet.

Table 6-2: Additional attributes for BMC_IPConnectivitySubnet

Attribute	Description
AddressType	An enumeration that describes the format of the Name and SubnetNumber properties in IPConnectivitySubnet: ■ 0—Unknown ■ 1—IPv4 ■ 2—IPv6
PrefixLength	A prefix length for IPv6 addresses in the IP subnet (AddressType property is 2).
SubnetMask	The mask for the starting address of the IPv6 IP subnet (AddressType is 1).
SubnetNumber	The IP address of the entire subnet; must be equal to the Name attribute.

Relationships for components on a subnet

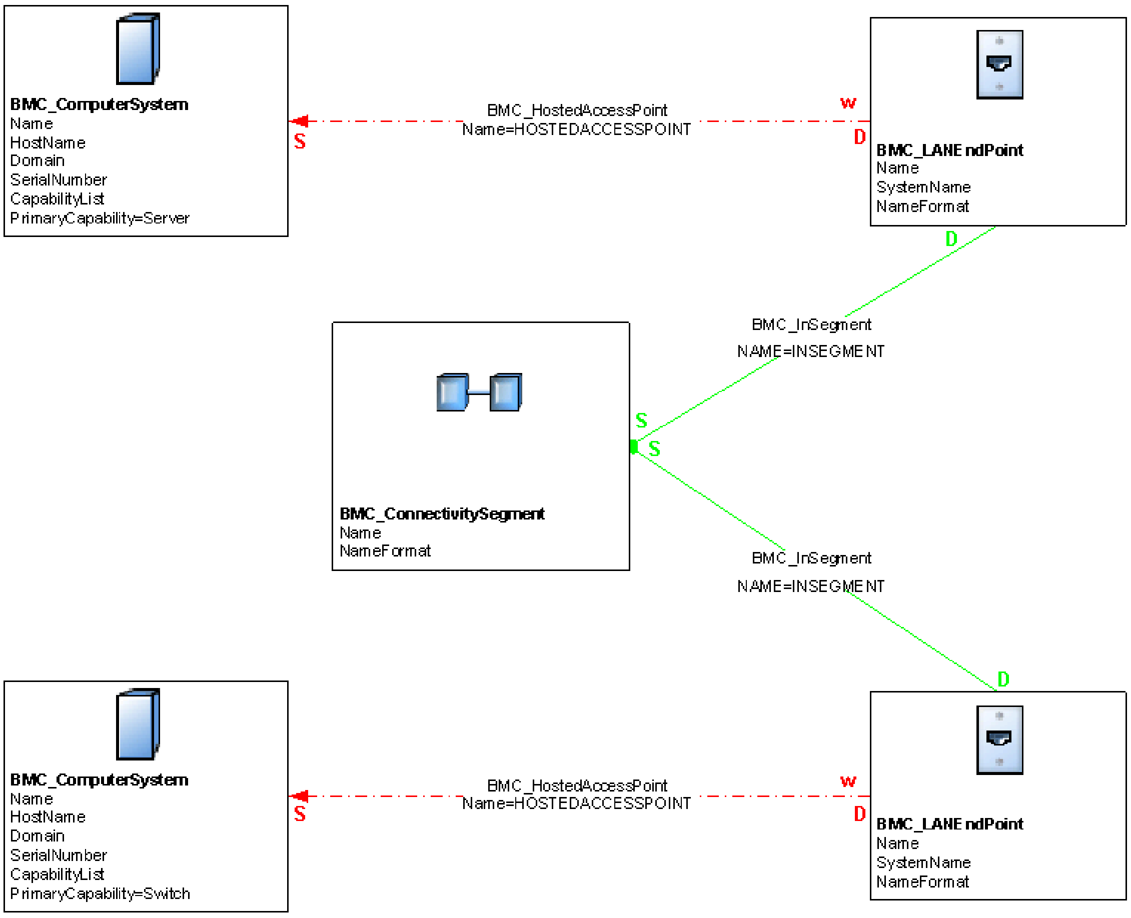
BMC_IPEndpoint instances are associated to the BMC_IPConnectivitySubnet to which they belong through the BMC_InIPSubnet relationship.

Modeling L2 topology and physical connectivity

A `BMC_ConnectivitySegment` instance represents a group of related instances of `BMC_LANEndpoint` of a particular type (such as Ethernet, token ring, or fiber channel) that can intercommunicate without the assistance of bridging or routing services. They are sometimes referred to as members of the same collision domain. The class describes the characteristics of the group, or segment.

Figure 2-2 illustrates a server and a switch, with the server having one NIC directly connected to a network interface of the switch.

Figure 6-2: Illustrative model of a network interface



Logical identity of BMC_ConnectivitySegment

Table 6-3 describes key attributes of `BMC_ConnectivitySegment`.

Table 6-3: Key attributes of BMC_ConnectivitySegment

Attribute	Usage
`Name`	Identifies the connectivity segment, which uses the following information and generates a hash code as resulting value: ■ The list of physical addresses that belong to the segment. ■ The name of the LAN instance to which the segment belongs.
`NameFormat`	Specifies the heuristic used to generate the `Name` value, which must be set to OID.

Additional attributes for BMC_ConnectivitySegment

Table 6-4 describes attributes that provide additional information about `BMC_ConnectivitySegment`.

Table 6-4: Additional attributes for BMC_ConnectivitySegment

Attribute	Description
`ConnectivityType`	An enumeration that describes the type of technology used: ■ 0—Unknown ■ 1—Other ■ 2—Ethernet ■ 3—Token Ring ■ 4—FDDI ■ 5—Fiber Channel
`Count`	The current number of endpoints connected to this segment. When this value equals 2 it indicates a direct connection between the two network ports interconnected by means of the segment.

Relationships for network interfaces

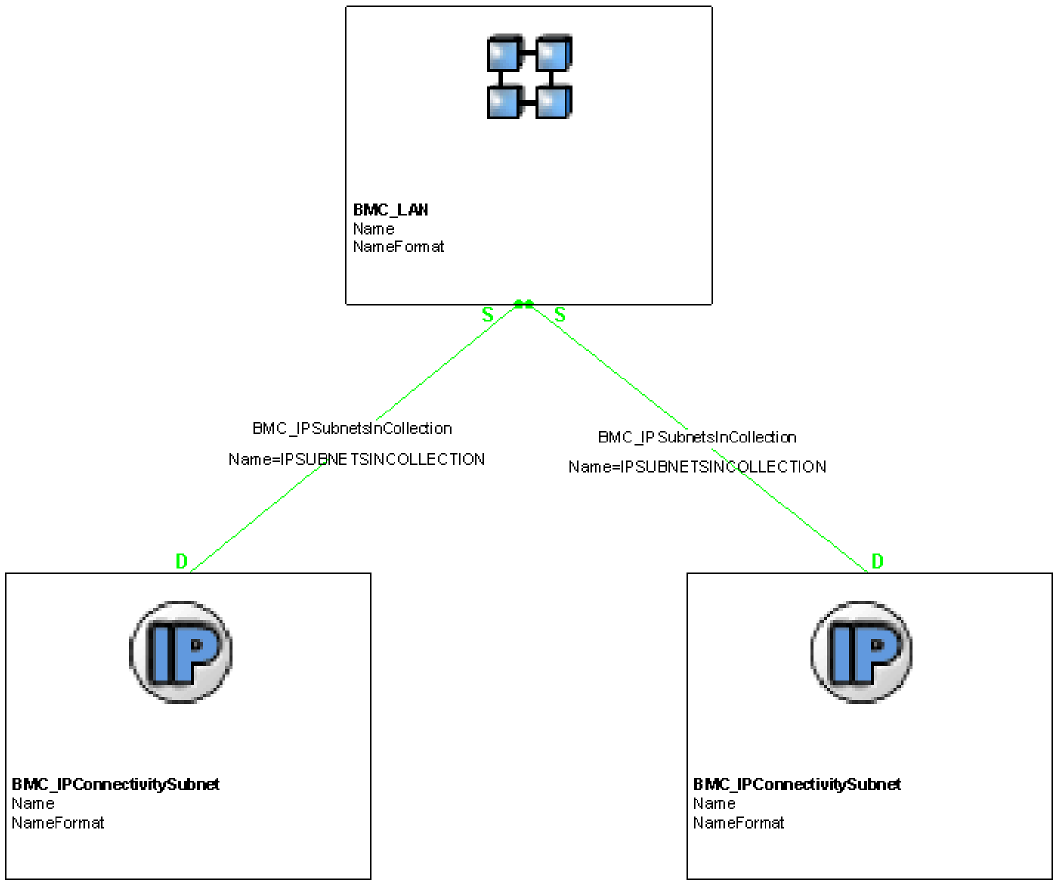
Instances of `BMC_LANEndpoint` are associated to the `BMC_ConnectivitySegment` to which they belong through the `BMC_InSegment` relationship.

Modeling network topology and LAN and WAN networks

LAN and WAN networks do not have a well-known identifier, such as an IP address or mask for an IP subnet. These networks are characterized by the list of machines that can intercommunicate at the physical level without crossing the boundaries of gateways. This description includes the infrastructure network devices (switches, hubs) that enable these machines to communicate. In the CDM, LANs and WANs are captured by entities that aggregate that list of IP subnets.

Figure 6-3 illustrates a LAN that aggregates IP subnets.

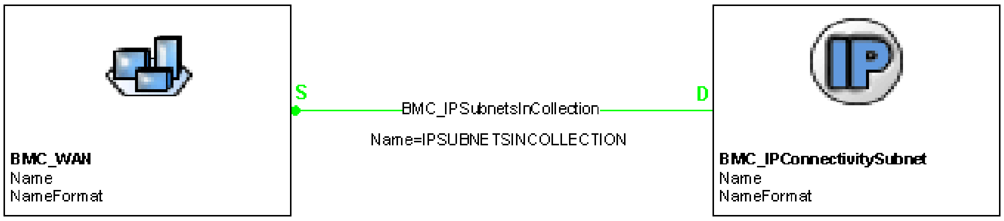
Figure 6-3: Illustrative model of a LAN


BMC_LAN
Name
NameFormat
S
S
BMC_IPSubnetsInCollection
Name=IPSUBNETSINCOLLECTION
BMC_IP SubnetsInCollection
Name=IPSUBNETSINCOLLECTION
D
D
BMC_IPConnectivitySubnet
Name
NameFormat
BMC_IPConnectivitySubnet
Name
NameFormat

Illustrative model of a WAN

Figure 6-4 illustrates an example of a WAN that aggregates IP subnets.

Figure 6-4: Illustrative model of a WAN



Logical identity of BMC_LAN

Table 6-5 describes key attributes of BMC_LAN.

Table 6-5: Key attributes of BMC_LAN

Attribute	Usage
Name	Identifies the LAN, computed from the list of IP subnets that make up the LAN. The name for the LAN is the lexicographically lower value of the names of IP subnets.
NameFormat	Specifies the heuristic used to generate the Name value, which must be set to OID.

Logical identity of BMC_WAN

Table 6-6 describes key attributes of a BMC_WAN.

Table 6-6: Key attributes of BMC_WAN

Attribute	Usage
Name	Identifies the WAN, computed from the list of IP subnets that make up the WAN. The name for the WAN is the lexicographically lower value of the names of IP subnets.
NameFormat	Specifies the heuristic used to generate the Name value, which must be set to OID.
WANType	Specifies the enumeration that describes the type of technology used: ■ 0—Unknown ■ 1—Other ■ 2—ATM ■ 3—Frame relay

Appendix

A Summary of changes to the Common Data Model

This appendix lists all the change to the Common Data Model in the 7.6.00 release.

The following topics are provided:

- Added classes and attributes (page 84)
- Moved attributes (page 84)
- Hidden attributes (page 84)

Added classes and attributes

The following classes and attributes have been added in BMC Atrium CMDB 7.6.00.

Table A-1: New classes and attributes

New class	Existing class	New attribute
BMC_Geneology	Not applicable	No attributes
Not applicable	BMC_VirtualSystemSettingData	ActualDecommissionDate
Not applicable	BMC_VirtualSystemSettingData	ActualProvisionDate
Not applicable	BMC_VirtualSystemSettingData	ProposedDecommissionDate

Moved attributes

In BMC Atrium CMDB 7.6.00, the `isVirtual` attribute was moved from the `BMC_ComputerSystem` class to the `BMC_SystemComponent` class to expand the virtualization scope beyond computer systems, enabling you to model potential future virtualizable entities, such as applications.

Hidden attributes

The `Dimensions` attribute in the `BMC_ComputerSystem` class is hidden in the Console view but can be seen in the Class Manager.

Glossary

abstract class
A *class* that has *attribute*s but of which no instances can be created. An abstract class exists for the purpose of creating an organizational layer without a database join. See also *data replication*.

account
An entity or party whose data is represented in BMC Atrium Core, and to whom specific levels of permission can be granted. Specifying instance permission by account enables BMC Atrium Core to support *multitenancy*.

activity
An individual reconciliation task that can be grouped together in a defined sequence to form a *reconciliation job*. You can run an activity only as part of a job, never by itself. See also *Comparison activity*, *Copy Dataset activity*, *Delete Dataset activity*, *Execute Job activity*, *Identification activity*, *Merge activity*, *Purge Dataset activity*, *Rename Dataset activity*.

Adapter Development Kit
The Adapter Development Kit is a software development kit that lets you build your own adapters to interact with *BMC Atrium Integration Engine*. The Adapter Development Kit interface defines a set of C++ objects that are used by the *AIE service* to manage data exchanges that use these adapters.

AIE Definitions Admin
A *BMC Atrium Integration Engine* application *role*. Members can view, create, and modify *data mapping*s and *data exchange*s, and manage the configuration and connection settings. See also *AIE User*.

AIE service
The AIE service obtains the defined *data exchange* from the Data Exchange application and completes the transfer of data by communicating with the adapter specified in the data exchange definition. The AIE service can connect to a BMC Remedy AR System server. It runs as a client to the BMC Remedy AR System server using the BMC Remedy AR System application programming interface (API).

AIE User
A *BMC Atrium Integration Engine* application *role*. Members can view *data mapping*s, *data exchange*s, and configuration and connection settings. See also *AIE Definitions Admin*.

Atrium Explorer
A component of BMC Atrium CMDB that graphically displays the relationships between *CI*s. It can also be embedded in other BMC Remedy AR System-based applications.

attribute
A property or characteristic of a *class*, such as the IP address of a computer system. An attribute equates to a column on a database table or a field on a BMC Remedy AR System form.

attribute permission
Permission to view or change the value in the attribute for any instance, assuming valid *instance permissions*.

attribute substitution
A method of data federation in context that uses placeholders to represent *attribute*s from a linked *class*. Launching the link triggers the respective *attribute* values to be substituted for the placeholders.

audit
: A logging of attribute values and other information for purposes of tracking the history of changes to *instance* data. An audit is triggered when the value of one or more specified *attribute*s changes or when the instance is created or deleted.

base class
: A *class* that has no *superclass*.

BMC Atrium Core Console
: The main user interface of BMC Atrium CMDB, accessible from both web and BMC Remedy User clients. The BMC Atrium Core Console replaced the CMDB Console.

BMC Atrium Integration Engine
: A product that enables you to transfer large amounts of data between third-party data sources and both the BMC Remedy AR System and BMC Atrium CMDB.

BMC Atrium Product Catalog
: A BMC Remedy AR System application that is part of the BMC Atrium Core solution and provides data for normalization and discovery, including storage of product signatures, and tracks and manages products by categorization, life cycle, development status, approval status, and other attributes.

bulk functions
: A set of functions you can use to create, update, and delete multiple instances in a single call. These functions can manipulate instances of different classes in a single operation.

Business Service Management (BSM)
: The concept of prioritizing IT efforts to support the overall goals of the business.

cardinality
: The number of members a *relationship class* can have on each side. Cardinality can be one to one, one to many, many to one, or many to many.

cascading delete
: To automatically delete, or mark as deleted the destination member of a relationship when the source member is deleted or marked.

Categories, Types, and Items (CTI)
: A method formerly used for categorizing assets in BMC Remedy Asset Management. Category, Type, and Item are each an attribute on the `BMC_BaseElement` class, so you can use CTIs in BMC Atrium CMDB.

categorization class
: A *class* that does not have its own BMC Remedy AR System form, but stores its instance data in the form of its *superclass*, preventing the need for a database join.

CDM
: See *Common Data Model (CDM)*.

child
: See *destination*.

CI
: See *configuration item (CI)*.

CI class
: A *class* that defines a type of *configuration item (CI)*, such as a computer system or software application.

CIM
: See *Common Information Model (CIM)*.

class
: Metadata in BMC Atrium CMDB that defines a type of object, usually a *configuration item (CI)* or *relationship*. Either of these types of class can store data as a *regular class*, *categorization class*, *abstract class*, or abstract class with *data replication*. You can apply the *final class* and *singleton class* options to it as well.

Class Manager
: A component of BMC Atrium CMDB where you can view, create, modify, and delete the *class*es and *attribute*s that make up the data model, as well as view a list of subclasses for each *class*.

class permissions
: Permission to view instances of a class in the BMC Atrium CMDB interface or access them with BMC Remedy AR System workflow.

CMDB
: See *Configuration Management Database (CMDB)*.

CMDB Console
See *BMC Atrium Core Console*.

CMDB Console Admin
An application *role*. Members can perform searches from the *BMC Atrium Core Console*, view, create, and modify *federation* definitions, and perform *BMC Atrium Core Console* administrative tasks.

CMDB Console User
An application *role*. Members can perform searches from the *BMC Atrium Core Console* and view *federation* definitions.

CMDB Data Change
An application *role*. Members can view, create, and modify *instance*s if they have *row-level security*.

CMDB Data Change All
An application *role*. Members can view, create, and modify *instance*s independent of *row-level security*.

CMDB Data View
An application *role*. Members can view *instance*s if they have *row-level security*.

CMDB Definitions Admin
An application *role*. Members can view, create, modify, and delete *class*es.

CMDB Definitions Viewer
An application *role*. Members can view *class* definitions.

CMDB Extended Data
Related data or CI attributes linked to or from BMC Atrium CMDB.

CMDB RE Definitions Admin
An application *role*. Members can view, create, modify, and delete *reconciliation definition*s and can start and cancel *job*s.

CMDB RE Manual Identification
An application *role*. Members can identify *instance*s manually.

CMDB RE User
An application *role*. Members can view *reconciliation definition*s and can start and cancel *job*s.

cmdbdriver
A utility that executes BMC Atrium CMDB C API functions from a command line, prompting for parameters.

Common Data Model (CDM)
The object-oriented, hierarchical set of *class*es in BMC Atrium CMDB used to represent types of *CI*s and relationships. The CDM is based on industry standards such as the *Common Information Model (CIM)* and Microsoft Windows Management Instrumentation.

Common Information Model (CIM)
A definition of management information developed by the *Distributed Management Task Force (DMTF)* that facilitates the exchange of management information between systems.

Comparison activity
A *Reconciliation Engine* activity that compares identified instances between two *dataset*s, either producing a report that shows the differences or executing workflow.

configuration data
Data about your IT environment, consisting of *CI*s and *relationship*s.

configuration item (CI)
A physical, logical, or conceptual entity that is part of your IT environment and has configurable *attribute*s. Examples include computer systems, buildings, employees, software, and business services. One of the two types of *class*es in BMC Atrium CMDB. See also *relationship*.

Configuration Management Database (CMDB)
A database that stores information about your IT configuration, including both *CI*s and *relationship*s.

consumer
An application that works with data in BMC Atrium CMDB. It might view the data or modify it. See also *provider*.

Copy Dataset activity
A *Reconciliation Engine* activity that copies instances from one *dataset* to another.

CTI
See *Categories, Types, and Items (CTI)*.

data exchange
A BMC Atrium Integration Engine integration object that you execute to transfer data. A data exchange specifies the source and destination in a data transfer, the adapter to be used for the transfer, and the connection parameters to connect to the external data store. A *data mapping* is associated with a data exchange to transfer data.

data mapping
A BMC Atrium Integration Engine integration object that defines the data to be transferred. A data mapping defines the source and destination of a data transfer, the *primary key*, and other fields and attributes. You can use a data mapping by associating it with a *data exchange*.

data replication
An option for *abstract class*es. With this option, the instances of all subclasses are replicated to a single form to allow you to search the abstract class as though it had data. Only the attributes inherited from the abstract class are replicated.

dataset
A logical group of data in BMC Atrium CMDB. A dataset can represent data from a particular source, a snapshot from a particular date, or other purpose. The dataset used by BMC Software products for reconciled production data is named BMC Asset. See also *overlay dataset*.

Dataset Merge Precedence
A pairing of a *dataset* with a *Precedence group*. Each *Merge activity* references a collection of these, called a Dataset Merge Precedence set.

defined dataset
One of a pair of dataset IDs that is specified when executing a *job* with dynamic dataset substitution. The job is executed with the *working dataset* in place of the defined dataset.

Definitive Hardware Library (DHL)
The Definitive Hardware Library (DHL) is a subset, or filter, of the Product Catalog that represents hardware products that are marked as approved for use in an organization.

Definitive Media Library (DML)
A repository where approved software configurations are stored. Installed instances of the software can be checked against the DML for compliance with licenses and policies. From the 7.5.00 release of BMC Atrium Core, Definitive Software Library (DSL) has been renamed to Definitive Media Library.

Definitive Software Library (DSL)
See *Definitive Media Library (DML)*.

Delete Dataset activity
A *Reconciliation Engine* activity that deletes instances from one or more *dataset*s without removing the *dataset* itself. See also *cascading delete*, *hard delete*, *soft delete*.

destination
The *CI class* defined as Class 2 in a *relationship class*, or an instance of that CI class as a member of such a *relationship*. Also known as the child member or weak member.

discovery
The act of scanning your environment for *configuration data*.

discovery application
An application that scans your environment for *configuration data* and can act as a *provider* to BMC Atrium CMDB.

Distributed Management Task Force (DMTF)
An organization appointed to facilitate the exchange of management information by promoting the initiation of industry standards and interoperability.

DHL
See *Definitive Hardware Library (DHL)*.

DML
See *Definitive Media Library (DML)*.

DMTF
See *Distributed Management Task Force (DMTF)*.

DSL
See *Definitive Media Library (DML)*.

Enterprise Integration Engine
See *BMC Atrium Integration Engine*.

event
A particular type of change to the *instance*s of specified *class*es. You can *publish* an event so that any instance of it is written to the `CMDB:Events` form. You can receive notification each time an instance of the event occurs by polling the form.

Exclusion rule
A rule that specifies an *attribute* to be excluded from participation in a Comparison activity.

Execute Job activity
A *Reconciliation Engine* activity that executes a job.

extension
A logical set of *class*es and *attribute*s, usually in its own *namespace*, that is not part of the *Common Data Model (CDM)*.

extension loader
The `cmdbExtLoader` program, which is used for installing data model extensions and importing other BMC Atrium CMDB data and metadata.

federated data
Data linked from *CI*s in BMC Atrium CMDB but stored externally. Federated data might represent more *attribute*s of the CIs or related information such as change requests on the CIs.

federated interface
An instance of the `BMC_FederatedInterface` *class* that specifies how to access a particular type of *federated data*. See also *federated link*.

federated link
The connection between a *class* or *CI* and a *federated interface*.

federated product
A product that holds *federated data*. It can be linked to more than one *federated interface*.

federation
The act of linking *CI*s in BMC Atrium CMDB to external data.

Federation Manager
A component of BMC Atrium CMDB that you can use to manage *federated data*. From the Federation Manager, you can view, create, and modify *federated product*s, *federated interface*s, and *federated link*s.

filter
A set of criteria for restricting the information displayed by the *Atrium Explorer*. This is different from a BMC Remedy AR System filter.

final class
A *class* that cannot have *subclass*es.

foreign key substitution
A method of federation that assigns a key from the *federated product* to each linked *CI*. Foreign key substitution is useful when no *attribute*s that also exist in BMC Atrium CMDB are stored in the federated product.

graph walk
The act of searching for *CI*s and relationships in BMC Atrium CMDB.

graph walk functions
A set of specific functions that are used to search for *CI*s and relationships in BMC Atrium CMDB. Use these functions when you want to search for CIs regardless of their class or relationship.

group
A set of a particular type of *reconciliation definition* that is referenced by an *activity*. See also *Identification group*, *Precedence group*, *Qualification group*, *Workflow Execution group*.

GUID
A globally unique identifier, automatically generated by the BMC Remedy AR System server. GUIDs are used for instance IDs, reconciliation IDs, and other cases where a unique value must be generated without human interaction.

hard delete
The act of removing an *instance* from BMC Atrium CMDB. See also *soft delete*.

Identification activity
A *Reconciliation Engine* activity that matches instances from two or more *dataset*s and assigns them the same identity, meaning that they represent the same real-life object.

Identification group
A set of *Identification rule*s that collectively identify *instance*s from a particular *dataset* against other datasets. Each dataset that participates in an *Identification activity* is paired with one Identification group.

Identification rule
A *rule* used when identifying *instance*s between datasets. When two instances match the qualification for the rule, they are assigned the same *reconciliation ID*.

identity
See *reconciliation ID*.

incident
Defined by *ITIL®* as any event that is not part of the standard operation of a service and that causes, or might cause, an interruption to, or a reduction in, the quality of that service.

incremental merge
A Merge activity that only processes instances created or modified since the activity was last run, saving otherwise useless processing time. Setting Force Attribute Merge to No makes a Merge activity incremental.

instance
An actual incarnation of a particular *class*, represented as a record in BMC Atrium CMDB. Both *CI*s and *relationship*s are instances of their respective *class*es.

instance ID
A *GUID* that BMC Atrium CMDB applies to each instance to uniquely identify it.

instance permissions
The right to view or modify a specific instance. These permissions are called *row-level security* and *write security*, respectively.

instantiate
To create an instance of a class.

ITIL®
The IT Infrastructure Library® (ITIL) is an internationally accepted set of best practices developed by the British government for management of IT services.

job
A group of one or more *reconciliation* activities executed in sequence. You cannot run an *activity* by itself; only as part of a job. You can start a job manually, with a schedule, with an *Execute Job activity*, with workflow, or with an API program.

key query
Limits the records transferred in a data transfer in the BMC Atrium Integration Engine. See also *row-level query*.

Merge activity
A *Reconciliation Engine* activity that merges two or more *dataset*s into a single complete and correct *dataset* based on *precedence value*s that favor the strengths of each dataset.

metadata
Definitions that describe the data stored in BMC Atrium CMDB. Metadata includes *class*es in the data model and special classes to define things such as *dataset*s and *federation* objects.

multitenancy
The separation of data and access so that a single BMC Atrium CMDB can contain the data of multiple parties, but each party can access only their own data. See also *account*, *role*.

namespace
A logical set of *class*es and *attribute*s in the data model, usually related to a specific *consumer* or *provider*. The *Common Data Model (CDM)* uses the `BMC.CORE` namespace.

normalize
To ensure that data of the same type follows the same text formatting conventions. This helps reconciliation by making it more likely for instances to match in an *Identification activity*.

Normalization Engine
A BMC Atrium CMDB application that provides a centralized, customizable, and uniform way to overcome consistency problems by normalizing attributes for software and hardware products.

orphan
An instance that has been physically deleted from source datasets but still exists in the target dataset into which they merge.

overlay dataset
A *dataset* that provides a layer in which to make changes that are pending approval. API queries to the dataset seamlessly return its modified instances along with unmodified instances from the underlying regular dataset.

parent
See *source*.

Precedence group
The definition of an overall *precedence value* for a *dataset*. It can optionally contain precedence values for specific classes and attributes within the dataset.

precedence value
A method of assigning weight to specific *dataset*s, *class*es, and *attribute*s in a *Merge activity*. Attribute precedence values override class precedence values, which override dataset precedence values.

primary key
A primary key uniquely identifies a row of data. In BMC Atrium Integration Engine you must specify the attributes of a CI class and the corresponding fields in the external data store to create the primary key. After a data transfer, the primary key is the link that matches a record in the external data store with its counterpart in BMC Atrium CMDB.

product categorization
Divides CIs into groups. Using the three-tier structure of product categorization, you can create successively smaller, more tightly defined groups. You can create groups of CIs in Tier 1 followed by smaller groups in Tier 2 and Tier 3.

production dataset
The dataset that serves as the single source of reference for your organization and from which you make business decisions. It acts as the target dataset in most Merge activities.

provider
An application, often a discovery application, that loads bulk data into BMC Atrium CMDB. See also *consumer*.

provisioning
The process of providing access to resources, such as printers, telephones, and so on, and to information, such as permissions, databases, and so on.

publish
To make an *event* available so that instances of it can be written to the `CMDB:Events` form.

Purge Dataset activity
A *Reconciliation Engine* activity that removes instances that have been marked as deleted from one or more *dataset*s.

Qualification
A Boolean statement that is evaluated to determine whether an instance should be included in an *activity*.

Qualification group
A set of *Qualification*s that can be used in various types of *activity*. An *instance* that meets one or more Qualifications in the group is included in the activity.

reconciliation
The process of managing data in multiple *dataset*s using the *Reconciliation Engine*. The main activities of reconciliation are identifying, comparing, and merging datasets, though the Reconciliation Engine performs other activities as well.

reconciliation definition
An entity defined in the *Reconciliation Manager* such as a *job*, *activity*, or group.

Reconciliation Engine
The component of BMC Atrium CMDB that reconciles data from different *dataset*s.

reconciliation ID
A *GUID* that the *Reconciliation Engine* assigns to *instance*s in different *dataset*s that represent the same real-life object.

Reconciliation Manager
The component of BMC Atrium CMDB that you can use to manage *reconciliation definition*s.

regular class
A *class* that stores its instance data in its own BMC Remedy AR System form. See also *abstract class*, *categorization class*.

related information
Information about a *CI* that does not qualify as attributes of the CI, and should therefore not be stored in a *Configuration Management Database (CMDB)*.

relationship
A connection between two *CI*s such as a dependency or membership. It is an instance of a *relationship class*. See also *configuration item (CI)*.

relationship class
A *class* that defines a type of relationship between *CI*s, such as a dependency or membership.

relationship filter
See *filter*.

relationship key
A relationship key uniquely identifies a row of data in a relationship mapping. In BMC Atrium Integration Engine you must specify the attributes of a primary CI class and a secondary CI class to create the relationship key. After a data transfer, the relationship key is the link that matches a record in the external data store with its counterpart in BMC Atrium CMDB.

Rename Dataset activity
A *reconciliation activity* that renames a *dataset* without changing its ID, preserving references to the dataset from any *reconciliation definition*s.

role
A designation that grants permissions to more than one BMC Remedy AR System group.

row-level query
Limits the transfer of data on a row-by-row basis in the BMC Atrium Integration Engine. See also *key query*.

row-level security
The permission required to view a specific *instance*. See also *write security*.

rule
One or more criteria that, when met, cause an action. The types of rules used in BMC Atrium Core are *Exclusion rule*, *Identification rule*, and *Workflow Execution rule*.

ruleset
A group of *rule*s.

service level agreement
A contract between a service provider and a purchaser that defines the level of service.

singleton class
An optional *class* characteristic that restricts the class to holding only one instance.

snapshot
A set of data that represents a configuration at a certain point in time, usually stored in its own *dataset*. There can be multiple snapshots of a given configuration.

soft delete
The act of marking an *instance* as deleted from BMC Atrium CMDB by setting the `MarkAsDeleted` attribute to `Yes`. See also *hard delete*.

source
The *CI class* defined as Class 1 in a *relationship class*, or an instance of that CI class as a member of such a *relationship*. Also known as the parent member or strong member.

subclass
A *class* that is derived from another *class*, which is called its *superclass*. The subclass inherits all the *attribute*s of its superclass and any superclasses above it in the hierarchy, and can also participate in relationships defined for all superclasses.

superclass
A *class* from which other classes, called *subclass*es, are derived.

synchronization
The automatic process of creating BMC Remedy AR System forms and workflow to represent a class that has just been created or modified. The class is not available until synchronization completes.

text normalization
See *normalize*.

weak reference
See *weak relationship*.

weak relationship
An optional characteristic for *relationship class*es, signifying that the members of a relationship form a composite object that can be reconciled as one. The *destination* member is considered the weak member of a weak relationship, existing as part of the *source* member.

Windows Management Instrumentation (WMI)
The Microsoft application of the Web-Based Enterprise Management initiative for an industry standard for accessing management information.

WMI
See *Windows Management Instrumentation (WMI)*.

workflow
BMC Remedy AR System objects such as active links, escalations, and filters that perform actions against data.

Workflow Execution group
A set of *Workflow Execution rule*s. Each *Comparison activity* can optionally reference one Workflow Execution group.

Workflow Execution rule
A *rule* used when comparing *instance*s between datasets. When a compared instance matches the qualification for the rule, specified BMC Remedy AR System workflow is executed against the instance or the instance against which it is compared.

working dataset
One of a pair of dataset IDs that is specified when executing a *job* with dynamic dataset substitution. The job is executed with the working dataset in place of the *defined dataset*.

write security
The permission required along with *row-level security* to modify or delete a specific *instance*. See also *row-level security*.

Index

103619